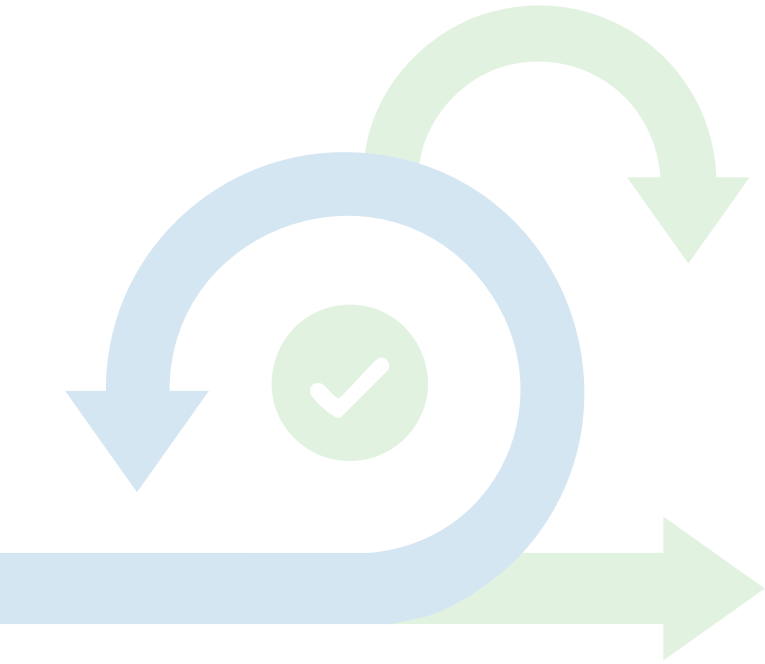




A Guide to the

SCRUM BODY OF KNOWLEDGE

(SBOK® Guide)

Fifth Edition

**A Comprehensive Guide to Implementing
and Scaling Scrum, with Practical Examples**

(Includes insights into how Artificial Intelligence can enhance Scrum processes)



A Guide to the

SCRUM BODY OF KNOWLEDGE

(SBOK® Guide)

Fifth Edition

A Comprehensive Guide to Implementing and Scaling Scrum,
with Practical Examples

(Includes insights into how Artificial Intelligence can enhance Scrum processes)

© 2025 SCRUMstudy™, a brand of VMEdU, Inc. All rights reserved.

Library of Congress Cataloging-in-Publication Data

A Guide to the Scrum Body of Knowledge (SBOK® Guide) – Fourth edition

Includes bibliographical references and index.

ISBN: 978-098992520-4

1. Scrum Framework. I. SCRUMstudy™. II. *SBOK® Guide*

2013950625

ISBN: 978-0-9899252-0-4

Published by:

SCRUMstudy™, a brand of VMEdU, Inc.

12725 W. Indian School Road, Suite F-112

Avondale, Arizona 85392 USA

Email: support@scrumstudy.com, sbok@scrumstudy.com

Website: www.scrumstudy.com

“SBOK,” the SCRUMstudy logo, “SFC,” “SDC,” “SMC,” “SAMC,” “SPOC,” “SSMC,” “SSPOC,” and “ESMC” are trademarks of SCRUMstudy™ (a brand of VMEdU, Inc.) For a comprehensive list of SCRUMstudy™ marks, contact the SCRUMstudy™ Legal Department.

A Guide to the Scrum Body of Knowledge (SBOK® Guide) is provided for educational purposes. SCRUMstudy™ or VMEdU, Inc. does not warrant that it is suitable for any other purpose and makes no expressed or implied warranty of any kind and assumes no responsibility for errors and omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

SCRUMstudy™ welcomes corrections and comments on its books. Please feel free to send comments on typographical, formatting, or other errors. You can make a copy of the relevant page of the book, mark the error, and send it to the above address or send an email to support@scrumstudy.com.

No part of this work may be reproduced or transmitted in any form or by any means, electronic, manual, photocopying, recording, or by any information storage and retrieval system, without prior written permission of the publisher.

10 9 8 7 6 5 4 3 2

PREFACE

A Guide to the Scrum Body of Knowledge (SBOK® Guide) provides guidelines for the successful implementation of Scrum—the most popular Agile product development and project delivery approach. Scrum, as defined in the *SBOK® Guide*, is a framework which is applicable to portfolios, programs, or projects of any size or complexity; and may be applied effectively in *any* industry to create a product, service, or any other result.

The *SBOK® Guide* draws from the combined knowledge and insight gained from thousands of projects across a variety of organizations and industries. This Fifth Edition adds to the collective contributions of experts in Scrum and project delivery. In particular, feedback from the global Scrum community played a key role in identifying improvements and additions to the *SBOK® Guide*. Its development has truly been a collaborative effort from a large number of experts and practitioners in a variety of disciplines.

Unlike other bodies of knowledge or books related to Scrum, the *SBOK® Guide* is completely free and supported on [Scrumstudy.com](https://www.scrumstudy.com), offering free certifications, webinars, videos, and study guides—ideal for any professional seeking a fundamental understanding of Business Analysis or exploring a career in the field. Additionally, the *SBOK® Guide* focuses on real-life issues and practical problems faced by Scrum practitioners and how these can be addressed using modern tools and Artificial Intelligence (AI).

The *SBOK® Guide* is well-organized and enjoyable to read. Unlike most Scrum books—which tend to be either overly simplistic or excessively detailed, requiring readers to go through the entire content to grasp key concepts—the *SBOK® Guide* follows the 80-20 rule: 80% of the core concepts can be understood by reading just 20% of the book. The remaining content can be consulted as needed when dealing with more complex Scrum tasks.

The *SBOK® Guide* is intended for use as a reference and knowledge guide by both experienced Scrum and other product or service development practitioners, as well as by people with no prior experience or knowledge of Scrum or any other project delivery method.

Although the *SBOK® Guide* is a comprehensive guide and framework for delivering projects using Scrum, its contents are organized for easy reference, regardless of the reader's prior knowledge on the subject. I hope each reader will learn from and enjoy it as much as the many authors and reviewers learned from and enjoyed the process of collating the collective knowledge and wisdom contained within it.



Tridibesh Satpathy,

Lead Author, *SBOK® Guide*

TABLE OF CONTENTS

1.	INTRODUCTION.....	1
1.1	Overview of Scrum.....	2
1.1.1	Brief History of Scrum.....	3
1.2	Why Use Scrum?.....	4
1.2.1	Scalability of Scrum.....	5
1.3	Purpose of the <i>SBOK® Guide</i>	6
1.4	SBOK® vs. other Scrum Books or Bodies of Knowledge.....	7
1.4.1	Free download of SBOK®.....	7
1.4.2	Free certifications, webinars, videos, study guides.....	7
1.4.3	Practical, Contemporary, and supports AI.....	7
1.4.4	Practice Scrum in Real Life with Vabro.ai.....	7
1.4.5	80-20 Rule: (20% of SBOK® is Sufficient).....	7
1.4.6	Easily organized and Fun to read.....	8
1.4.7	Applicable to Organizations of all Sizes and Industries.....	8
1.4.8	Aligns with other Frameworks including Kanban, OKRs, DevOps, Business Analysis.....	8
1.4.9	Advanced Certifications for Senior Scrum Professionals.....	8
1.4.10	Available in 6 languages.....	9
1.4.11	2,500+ Partners in 50+ Countries, 300,000+ LinkedIn Group.....	9
1.5	Framework of the <i>SBOK® Guide</i>	10
1.5.1	How to Use the <i>SBOK® Guide</i> ?.....	11
1.5.2	Scrum Principles.....	11
1.5.3	Scrum Aspects.....	13
1.5.4	Scrum Processes.....	16
1.5.5	Scrum for Large Projects.....	21
1.5.6	Scrum for the Enterprise.....	21
1.6	Scrum vs. Traditional Project Management.....	22
2.	PRINCIPLES.....	23
2.1	Introduction.....	23
2.1.1	Roles Guide.....	24
2.2	Empirical Process Control.....	25
2.2.1	Transparency.....	25

TABLE OF CONTENTS

2.2.2	Inspection.....	27
2.2.3	Adaptation	29
2.3	Self-organization	31
2.3.1	Benefits of Self-organization	31
2.4	Collaboration.....	34
2.4.1	Benefits of Collaboration in Scrum Projects.....	34
2.4.2	Importance of Colocation in Collaboration	37
2.4.3	Collaboration in Distributed Teams	37
2.4.4	AI-powered Scrum Project Tool	37
2.5	Value-based Prioritization.....	42
2.6	Time-boxing	44
2.6.1	Scrum Time-boxes.....	44
2.7	Iterative Development	47
2.8	Scrum vs. Traditional Project Management.....	49
3.	ORGANIZATION	51
3.1	Introduction.....	51
3.1.1	Roles Guide.....	52
3.2	Scrum Project Roles	53
3.2.1	Core Roles	54
3.2.2	Non-core Roles.....	55
3.3	Product Owner.....	57
3.3.1	Voice of the Customer (VOC)	58
3.4	Scrum Master	59
3.5	Scrum Team	60
3.5.1	Personnel Selection.....	61
3.5.2	Scrum Team Size	61
3.6	Scrum in Large Projects, Programs, and Portfolios.....	62
3.6.1	Making Scrum Work in a Large Project	62
3.6.2	Additional Core Roles in a Large Project.....	63
3.6.3	Making Scrum Work in an Enterprise Environment.....	65
3.6.4	Additional Core Roles in an Enterprise Environment	67
3.6.5	Examples of Projects, Programs, and Portfolios	69
3.7	Summary of Responsibilities	71

3.8	Scrum vs. Traditional Project Management.....	72
3.9	Popular HR Theories and their Relevance to Scrum	73
3.9.1	Tuckman's Model of Group Dynamics.....	73
3.9.2	Conflict Management.....	74
3.9.3	Conflict Management Techniques	74
3.9.4	Leadership Styles	75
3.9.5	Theory X, Theory Y, and Theory Z	77
4.	BUSINESS JUSTIFICATION	79
4.1	Introduction.....	79
4.1.1	Roles Guide.....	80
4.2	Value-driven Delivery	81
4.2.1	Responsibilities of the Product Owner in Business Justification	82
4.2.2	Responsibilities of Other Scrum Roles in Business Justification	83
4.3	Importance of Business Justification.....	84
4.3.1	Factors Used to Determine Business Justification	84
4.3.2	Business Justification and the Project Lifecycle.....	85
4.4	Business Justification Techniques	87
4.4.1	Estimation of Project Value	87
4.4.2	Planning for Value	88
4.4.3	Relative Prioritization Ranking	90
4.4.4	Story Mapping	91
4.5	Continuous Value Justification.....	92
4.5.1	Sprint Burndown or Burnup Chart	92
4.5.2	Cumulative Flow Diagram (CFD).....	93
4.6	Confirm Benefits Realization	95
4.6.1	Prototypes, Simulations, and Demonstrations	95
4.7	Summary of Responsibilities	96
4.8	Scrum vs. Traditional Project Management.....	97
5.	QUALITY.....	99
5.1	Introduction.....	99
5.1.1	Roles Guide.....	99
5.2	Quality Defined	100
5.2.1	Quality and Scope	100

TABLE OF CONTENTS

5.2.2	Quality and Business Value	101
5.3	Acceptance Criteria and the Prioritized Product Backlog.....	102
5.3.1	Writing Acceptance Criteria.....	103
5.3.2	Definition of Ready	103
5.3.3	Definition of Done (or Done Criteria)	104
5.3.4	Minimum Done Criteria	106
5.3.5	Acceptance or Rejection of Prioritized Product Backlog Items	106
5.4	Quality Management in Scrum	108
5.4.1	Quality Planning	108
5.4.2	Quality Assurance and Quality Control.....	110
5.4.3	Plan-Do-Check-Act (PDCA) Cycle	110
5.5	Summary of Responsibilities	112
5.6	Scrum vs. Traditional Project Management.....	113
6.	CHANGE.....	115
6.1	Introduction.....	115
6.1.1	Roles Guide.....	116
6.2	Overview.....	117
6.2.1	Unapproved and Approved Change Requests	117
6.3	Understanding Change in Scrum.....	119
6.3.1	Balancing Flexibility and Stability	119
6.3.2	Incorporating Flexibility	119
6.4	Integrating Change.....	124
6.4.1	Changes to a Sprint.....	124
6.5	Change in Programs and Portfolios	128
6.5.1	In Programs.....	128
6.5.2	In Portfolios	128
6.6	Summary of Responsibilities	130
6.7	Scrum vs. Traditional Project Management.....	131
7.	RISK	133
7.1	Introduction.....	133
7.1.1	Roles Guide.....	134
7.2	What is Risk?.....	135
7.2.1	Difference between Risks and Issues.....	135

TABLE OF CONTENTS

7.2.2	Risk Attitude	136
7.3	Risk Management Procedure	137
7.3.1	Risk Identification	137
7.3.2	Risk Assessment	138
7.3.3	Risk Prioritization	140
7.3.4	Risk Mitigation	141
7.3.5	Risk Communication	142
7.4	Minimizing Risks through Scrum	144
7.5	Risks in Portfolios and Programs	145
7.5.1	In Portfolios	145
7.5.2	In Programs	145
7.6	Summary of Responsibilities	147
7.7	Scrum vs. Traditional Project Management	148
8.	INITIATE	149
8.1	Create Project Vision	153
8.1.1	Inputs	155
8.1.2	Tools	156
8.1.3	Outputs	158
8.2	Identify Scrum Master and Business Stakeholder(s)	162
8.2.1	Inputs	164
8.2.2	Tools	165
8.2.3	Outputs	166
8.3	Form Scrum Team	167
8.3.1	Inputs	169
8.3.2	Tools	170
8.3.3	Outputs	173
8.4	Develop Epic(s)	174
8.4.1	Inputs	175
8.4.2	Tools	178
8.4.3	Outputs	181
8.5	Create Prioritized Product Backlog	182
8.5.1	Inputs	183
8.5.2	Tools	185

TABLE OF CONTENTS

8.5.3	Outputs	189
8.6	Conduct Release Planning	191
8.6.1	Inputs	192
8.6.2	Tools	194
8.6.3	Outputs	196
8.7	Initiate Phase Data Flow Diagram	198
9.	PLAN AND ESTIMATE	199
9.1	Create User Stories	203
9.1.1	Inputs	204
9.1.2	Tools	206
9.1.3	Outputs	208
9.2	Estimate User Stories	210
9.2.1	Inputs	211
9.2.2	Tools	212
9.2.3	Outputs	216
9.3	Commit User Stories	217
9.3.1	Inputs	218
9.3.2	Tools	219
9.3.3	Outputs	220
9.4	Identify Tasks	222
9.4.1	Inputs	223
9.4.2	Tools	224
9.4.3	Outputs	226
9.5	Estimate Tasks	228
9.5.1	Inputs	229
9.5.2	Tools	230
9.5.3	Outputs	232
9.6	Update Sprint Backlog	233
9.6.1	Inputs	234
9.6.2	Tools	235
9.6.3	Outputs	237
9.7	Plan and Estimate Phase Data Flow Diagram	238
10.	IMPLEMENT	239

TABLE OF CONTENTS

10.1	Create Deliverables.....	243
10.1.1	Inputs	244
10.1.2	Tools	247
10.1.3	Outputs	248
10.2	Conduct Daily Standup.....	250
10.2.1	Inputs	251
10.2.2	Tools	252
10.2.3	Outputs	254
10.3	Refine Prioritized Product Backlog	256
10.3.1	Inputs	257
10.3.2	Tools	259
10.3.3	Outputs	261
10.4	Implement Phase Data Flow Diagram	262
11.	REVIEW AND RETROSPECT.....	263
11.1	Demonstrate and Validate Sprint.....	266
11.1.1	Inputs	268
11.1.2	Tools	269
11.1.3	Outputs	271
11.2	Retrospect Sprint	272
11.2.1	Inputs	273
11.2.2	Tools	274
11.2.3	Outputs	276
11.3	Review and Retrospect Phase Data Flow Diagram.....	278
12.	RELEASE.....	279
12.1	Ship Deliverables	282
12.1.1	Inputs	283
12.1.2	Tools	284
12.1.3	Outputs	286
12.2	Retrospect Release.....	287
12.2.1	Inputs	288
12.2.2	Tools	289
12.2.3	Outputs	290
12.3	Release Phase Data Flow Diagram.....	291

13. SCALING SCRUM FOR LARGE PROJECTS	293
13.1 Impact of Large Projects on Fundamental Scrum Processes	296
13.1.1 Initiate	296
13.1.2 Plan and Estimate	302
13.1.3 Implement	305
13.1.4 Review and Retrospect.....	308
13.1.5 Release.....	310
13.2 Additional Inputs and Outputs for Large Projects	312
13.2.1 Large Project Scrum Organization*	312
13.2.2 Product Owners Collaboration Plan*	312
13.2.3 Scrum Masters/Scrum Teams Collaboration Plan*	312
13.2.4 Shared Resources*	313
13.2.5 Team Specialization*	313
13.2.6 Environment and Environment Schedule*	314
13.2.7 Release Readiness Plan*	314
13.3 Additional Tools for Large Projects	315
13.3.1 Large Project Communications Plan	315
13.3.2 Large Project Resource Planning*	315
13.3.3 Environment Identification*	316
13.3.4 Prioritized Product Backlog Assignment*.....	316
13.3.5 Scrum of Scrums (SoS) Meeting*.....	317
13.3.6 Release Preparation Methods*	317
13.3.7 Release Readiness Sprint.....	318
13.3.8 AI-powered Scrum Project Tool	318
14. SCALING SCRUM FOR THE ENTERPRISE	319
14.1 Impact of Programs or Portfolios to Fundamental Scrum Processes	321
14.1.1 Initiate	321
14.1.2 Plan and Estimate	324
14.1.3 Implement	324
14.1.4 Review and Retrospect.....	325
14.1.5 Release.....	326
14.2 Additional Processes to Scale Scrum for the Enterprise (Program/Portfolio)	328
14.3 Create/Update Program or Portfolio Teams	331

TABLE OF CONTENTS

14.3.1	Inputs	331
14.3.2	Tools	332
14.3.3	Outputs	333
14.4	Create/Update Program or Portfolio Components	334
14.4.1	Inputs	334
14.4.2	Tools	337
14.4.3	Outputs	339
14.5	Review and Update Scrum Guidance Body	341
14.5.1	Inputs	341
14.5.2	Tools	342
14.5.3	Outputs	343
14.6	Create/Refine Prioritized Program or Portfolio Backlog	344
14.6.1	Inputs	344
14.6.2	Tools	347
14.6.3	Outputs	348
14.7	Create/Update Program or Portfolio Releases	350
14.7.1	Inputs	350
14.7.2	Tools	352
14.7.3	Outputs	353
14.8	Retrospect Program or Portfolio Releases	354
14.8.1	Inputs	354
14.8.2	Tools	355
14.8.3	Outputs	356
APPENDIX A. OVERVIEW OF AGILE		357
APPENDIX B. FIFTH EDITION UPDATES		367
REFERENCES		369
GLOSSARY		371
INDEX		405

LIST OF FIGURES

Figure 1-1: Scrum Flow for One Sprint	2
Figure 1-2: SBOK® Guide Framework.....	10
Figure 1-3: Scrum Principles	12
Figure 1-4: Organization in Scrum.....	14
Figure 2-1: Transparency in Scrum	26
Figure 2-2: Transparency Facilitated through Product Backlog (Source: Vabro).....	26
Figure 2-3: Open Product Backlog used for Transparency (Source: Jira)	27
Figure 2-4: Inspection in Scrum.....	27
Figure 2-5: Transparency in Scrum through the use of AI-powered Scrum Project Tool (Source: Vabro).....	28
Figure 2-6: Transparency in Scrum through the use of AI-powered Scrum Project Tool (Source: Jira)	28
Figure 2-7: Adaptation in Scrum.....	29
Figure 2-8: Challenges in Traditional Project Management	30
Figure 2-9: Goals of a Self-organizing Team.....	32
Figure 2-10: Facilitating self-organization via the Scrumboard in an AI-powered Scrum project tool (Source: Vabro).....	33
Figure 2-11: Scrumboard-enabled self-organization in AI-powered Scrum tool (Source: Jira).....	33
Figure 2-12: Benefits of Collaboration in Scrum Projects	35
Figure 2-13: Message Center in an AI-powered Scrum Project Tool (Source: Vabro).....	36
Figure 2-14: Task detail view for enhanced collaboration (Source: Vabro)	36
Figure 2-15: Value-based Prioritization.....	43
Figure 2-16: Value-based prioritization in AI-powered Scrum Project Tool (Source: Vabro).....	43
Figure 2-17: Value-based prioritization in AI-powered Scrum Project Tool (Source: Jira).....	43
Figure 2-18: Time-Box Durations for Scrum Meetings.....	45
Figure 2-19: Sprint Planning with Vabro Genie (Source: Vabro).....	46
Figure 2-20: Time-boxed Sprint Planning (Source: Jira).....	46
Figure 2-21: Scrum vs. Traditional Waterfall	48
Figure 3-1: Organization Setup (Source: Vabro)	53
Figure 3-2: Team Composition (Source: Airtable)	54
Figure 3-3: Scrum Roles—Overview	55
Figure 3-4: Desirable Traits for the Core Scrum Roles.....	61
Figure 3-5: Questions Asked during a Scrum of Scrums Meeting.....	64
Figure 3-6: Create Program Interface (Source: Vabro).....	65
Figure 3-7: Create Portfolio Interface (Source: Vabro)	66
Figure 3-8: Scrum across the Organization for Projects, Programs, and Portfolios	70
Figure 3-9: Tuckman's Stages of Group Development.....	73
Figure 4-1: Delivering Value in Scrum vs. Traditional Projects	82
Figure 4-2: Business Justification and the Project Lifecycle	86
Figure 4-3: Value Stream Mapping	89
Figure 4-4: Kano Analysis	90
Figure 4-5: Story Mapping.....	91

LIST OF FIGURES

Figure 4-6: Sprint Burndown Chart	92
Figure 4-7: Sprint Burnup Chart.....	93
Figure 4-8: Sample Cumulative Flow Diagram (CFD)	94
Figure 5-1: Project Increment Flow Diagram.....	102
Figure 5-2: Sample DoR presented in an AI-powered Scrum Project Tool (Source: Vabro).....	104
Figure 5-3: : Sample DoD presented in an AI-powered Scrum Project Tool (Source: Vabro).....	105
Figure 5-4: User Story Approval Process in an AI-powered Scrum Project Tool (Source: Vabro).....	107
Figure 5-5: PDCA Cycle in Scrum	111
Figure 6-1: Sample Change Approval Process	118
Figure 6-2: Updating Prioritized Product Backlog with Approved Changes	118
Figure 6-3: Scrum Characteristics for Achieving Flexibility	119
Figure 6-4: Motivation of Business Stakeholders for Requesting Changes	120
Figure 6-5: Motivation of Scrum Core Team for Requesting Changes	121
Figure 6-6: Incorporating a Change during a Sprint	124
Figure 6-7: Impact of the Probability of Change on the Length of Sprint.....	125
Figure 6-8: Incorporating Changes in Programs and Portfolios	129
Figure 7-1: Sample Probability Tree	138
Figure 7-2: Sample Pareto Chart.....	139
Figure 7-3: Sample Probability and Impact Matrix.....	140
Figure 7-4: Process for Risk Prioritization.....	141
Figure 7-5: Sample Risk Burndown Chart.....	143
Figure 7-6: Handling Risks in Portfolios and Programs	146
Figure 8-1: Initiate Overview.....	151
Figure 8-2: Initiate Overview (Essentials).....	152
Figure 8-3: Create Project Vision—Inputs, Tools, and Outputs	153
Figure 8-4: Create Project Vision—Data Flow Diagram	154
Figure 8-5: The Gap Analysis Process	157
Figure 8-6: Scrum Project Tools – Vabro.....	160
Figure 8-7: Scrum Project Tools - Jira	160
Figure 8-8: Identify Scrum Master and Business Stakeholder(s)—Inputs, Tools, and Outputs	162
Figure 8-9: Identify Scrum Master and Business Stakeholder(s)—Data Flow Diagram.....	163
Figure 8-10: Form Scrum Team—Inputs, Tools, and Outputs	167
Figure 8-11: Form Scrum Team—Data Flow Diagram	168
Figure 8-12: Creating a Team (Source: Vabro).....	172
Figure 8-13: Team Set-up (Source: Jira)	172
Figure 8-14: Develop Epic(s)—Inputs, Tools, and Outputs.....	174
Figure 8-15: Develop Epic(s)—Data Flow Diagram.....	175
Figure 8-16: Sample Scrum Project Tool – Vabro	180
Figure 8-17: Sample Scrum Project Tool – Jira.....	180
Figure 8-18: Create Prioritized Product Backlog—Inputs, Tools, and Outputs	182
Figure 8-19: Create Prioritized Product Backlog—Data Flow Diagram	183
Figure 8-20: Creation of Prioritized Product Backlog (Source: Vabro)	188
Figure 8-21: Use of Prioritized Product Backlog (Source: Jira).....	188
Figure 8-22: Conduct Release Planning—Inputs, Tools, and Outputs	191

LIST OF FIGURES

Figure 8-23: Conduct Release Planning—Data Flow Diagram	192
Figure 8-24: Release Planning in Scrum (Source: Vabro)	195
Figure 8-25: Release and Configuration Management in Scrum (Source: Jira).....	195
Figure 8-26: Initiate Phase—Data Flow Diagram	198
Figure 9-1: Plan and Estimate Overview.....	201
Figure 9-2: Plan and Estimate Overview (Essentials).....	202
Figure 9-3: Create User Stories—Inputs, Tools, and Outputs.....	203
Figure 9-4: Create User Stories—Data Flow Diagram.....	204
Figure 9-5: Adding User Stories in Prioritized Product Backlog (Source: Vabro).....	207
Figure 9-6: Adding User Stories in Product Backlog (Source: Jira).....	208
Figure 9-7: Estimate User Stories—Inputs, Tools, and Outputs.....	210
Figure 9-8: Estimate User Stories—Data Flow Diagram.....	211
Figure 9-9: User Stories Estimation (Source: Vabro).....	215
Figure 9-10: Estimating User Stories in Scrum (Source: Jira).....	215
Figure 9-11: Commit User Stories—Inputs, Tools, and Outputs	217
Figure 9-12: Commit User Stories—Data Flow Diagram	218
Figure 9-13: Typical Scrumboard (Source: Vabro)	221
Figure 9-14: Identify Tasks—Inputs, Tools, and Outputs.....	222
Figure 9-15: Identify Tasks—Data Flow Diagram.....	223
Figure 9-16: Task Creation in Scrum Project Tool (Source: Vabro)	225
Figure 9-17: Task Creation in Scrum Project Tool (Source: Jira).....	225
Figure 9-18: Scrumboard with Identified Tasks	226
Figure 9-19: Estimate Tasks—Inputs, Tools, and Outputs	228
Figure 9-20: Estimate Tasks—Data Flow Diagram	229
Figure 9-21: Task Estimation in Scrum (Source: Vabro).....	231
Figure 9-22: Use of Scrum Tool for Task Estimation (Source: Jira)	232
Figure 9-23: Update Sprint Backlog—Inputs, Tools, and Outputs.....	233
Figure 9-24: Update Sprint Backlog—Data Flow Diagram.....	234
Figure 9-25: Setting up Sprint Backlog in Scrum (Source: Vabro)	236
Figure 9-26: Use of Sprint Backlog in Scrum Project Tool (Source: Jira).....	237
Figure 9-27: Plan and Estimate Phase—Data Flow Diagram	238
Figure 10-1: Implement Overview.....	241
Figure 10-2: Implement Overview (Essentials).....	242
Figure 10-3: Create Deliverables—Inputs, Tools, and Outputs.....	243
Figure 10-4: Create Deliverables—Data Flow Diagram.....	244
Figure 10-5: Scrumboard with Tasks To Do, In Progress, and Complete	245
Figure 10-6: Conduct Daily Standup—Inputs, Tools, and Outputs.....	250
Figure 10-7: Conduct Daily Standup—Data Flow Diagram.....	251
Figure 10-8: Use of Scrumboard (Source: Vabro)	254
Figure 10-9: Role of Scrumboard in Scrum (Source: Jira).....	254
Figure 10-10: Refine Prioritized Product Backlog—Inputs, Tools, and Outputs.....	256
Figure 10-11: Refine Prioritized Product Backlog—Data Flow Diagram.....	257
Figure 10-12: Refining Product Backlog in Scrum (Source: Vabro)	260
Figure 10-13: Product Backlog Refinement in Scrum (Source: Jira).....	260

LIST OF FIGURES

Figure 10-14: Implement Phase—Data Flow Diagram	262
Figure 11-1: Review and Retrospect Overview	264
Figure 11-2: Review and Retrospect Overview (Essentials)	265
Figure 11-3: Demonstrate and Validate Sprint—Inputs, Tools, and Outputs	266
Figure 11-4: Demonstrate and Validate Sprint—Data Flow Diagram	267
Figure 11-5: User Story Review and Approval (Source: Vabro)	270
Figure 11-6: Retrospect Sprint—Inputs, Tools, and Outputs	272
Figure 11-7: Retrospect Sprint—Data Flow Diagram	273
Figure 11-8: Review and Retrospect Phase—Data Flow Diagram	278
Figure 12-1: Release Overview	280
Figure 12-2: Release Overview (Essentials)	281
Figure 12-3: Ship Deliverables—Inputs, Tools, and Outputs	282
Figure 12-4: Ship Deliverables—Data Flow Diagram	283
Figure 12-5: Release Management in Scrum (Source: Vabro)	285
Figure 12-6: Release Management in Scrum (Source: Jira)	286
Figure 12-7: Retrospect Release—Inputs, Tools, and Outputs	287
Figure 12-8: Retrospect Release—Data Flow Diagram	288
Figure 12-9: Release Phase—Data Flow Diagram	291
Figure 13-1: Setting up a Large Project (Source: Vabro)	295
Figure 13-2: Initiating a Large Project (Source: Vabro)	296
Figure 13-3: Creating a User Story in Plan and Estimate Phase (Source: Vabro)	302
Figure 13-4: Prioritization of User Stories in Implement Phase (Source: Vabro)	305
Figure 14-1: Creating a Release during Initiation (Source: Vabro)	324
Figure 14-2: Program Release Dashboard (Source: Vabro)	326
Figure 14-3: Scaling Scrum for the Enterprise	330
Figure 14-4: Create/Update Program or Portfolio Teams—Inputs, Tools, and Outputs	331
Figure 14-5: Create/Update Program or Portfolio Components—Inputs, Tools, and Outputs	334
Figure 14-6: Company Vision Statement Template (Source: Smartsheet)	335
Figure 14-7: Company Mission Statement Template (Source: Smartsheet)	336
Figure 14-8: Scrum of Scrums of Scrums (SoSoS) Meeting	338
Figure 14-9: Review and Update Scrum Guidance Body—Inputs, Tools, and Outputs	341
Figure 14-10: Create/Refine Prioritized Program or Portfolio Backlog—Inputs, Tools, and Outputs	344
Figure 14-11: Create/Update Program or Portfolio Releases—Inputs, Tools, and Outputs	350
Figure 14-12: Retrospect Program or Portfolio Releases—Inputs, Tools, and Outputs	354

LIST OF TABLES

Table 1-1: Summary of Fundamental Scrum Processes	17
Table 1-2: Scrum Meetings and Processes	20
Table 1-3: Scrum vs. Traditional Project Management	22
Table 2-1: AI Tools Commonly Used in Scrum Projects	41
Table 3-1: Responsibilities of the Product Owner in Scrum Processes	58
Table 3-2: Responsibilities of the Scrum Master in Scrum Processes	59
Table 3-3: Responsibilities of the Scrum Team in Scrum Processes	60
Table 3-4: Summary of Responsibilities Relevant to Organization	71
Table 4-1: Hierarchy for Business Justification Responsibilities	83
Table 4-2: Summary of Responsibilities Relevant to Business Justification	96
Table 5-1: Cascading Done Criteria	106
Table 5-2: Summary of Responsibilities Relevant to Quality	112
Table 6-1: Summary of Responsibilities Relevant to Change	130
Table 7-1: Summary of Responsibilities Relevant to Risk	147
Table 13-1: Impact of Large Projects to Fundamental Scrum Processes—Initiate Phase	301
Table 13-2: Impact of Large Projects to Fundamental Scrum Processes—Plan and Estimate Phase	304
Table 13-3: Impact of Large Projects to Fundamental Scrum Processes—Implement Phase	307
Table 13-4: Impact of Large Projects to Fundamental Scrum Processes—Review and Retrospect Phase	309
Table 13-5: Impact of Large Projects to Fundamental Scrum Processes—Release Phase	311
Table 14-1: Impact of a Program or Portfolio to Fundamental Scrum Processes—Initiate Phase	323
Table 14-2: Impact of a Program or Portfolio to Fundamental Scrum Processes—Implement Phase	325
Table 14-3: Impact of a Program or Portfolio to Fundamental Scrum Processes—Review & Retrospect Phase	325
Table 14-4: Impact of a Program or Portfolio to Fundamental Scrum Processes—Release Phase	327

1. INTRODUCTION

A Guide to the Scrum Body of Knowledge (SBOK® Guide) provides guidelines for the successful implementation of Scrum—the most popular Agile project delivery and product development approach. It provides a comprehensive framework that includes the principles, aspects, and processes of Scrum.

Scrum, as defined in the *SBOK® Guide*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

This first chapter describes the purpose and framework of the *SBOK® Guide* and introduces the key concepts of Scrum. It contains a summary of Scrum principles, Scrum aspects and Scrum processes. Chapter 2 expands on the six Scrum principles which are the foundation on which the Scrum framework is based. Chapters 3 through 7 elaborate on the five Scrum aspects that must be addressed throughout any project: organization, business justification, quality, change, and risk. Chapters 8 through 12 cover the 19 Scrum processes involved in carrying out a Scrum project. These processes are part of the five Scrum phases: Initiate; Plan and Estimate; Implement; Review and Retrospect; and Release. These phases describe in detail the associated inputs and outputs of each process, as well as the various tools that may be used in each process. Some inputs, tools, and outputs are mandatory and are indicated as such; others are optional and can be used depending on the specific project, organizational requirements, and/or guidelines set forth by the organization’s Scrum Guidance Body (SGB). Chapters 13 and 14 provide guidance on scaling Scrum for large projects and at the enterprise level (which involves programs and portfolios).

This chapter is divided into the following sections:

1.1 Overview of Scrum

1.2 Why Use Scrum?

1.3 Purpose of the *SBOK® Guide*

1.4 Framework of the *SBOK® Guide*

1.5 Scrum vs. Traditional Project Management

1.1 Overview of Scrum

A Scrum project involves a collaborative effort to create a new product, service, or other result as defined in the Project Vision Statement. Projects are impacted by constraints of time, cost, scope, quality, resources, organizational capabilities, and other limitations that make it difficult to plan, execute, manage, and succeed. However, successful implementation of the results of a finished project provides significant business benefits to an organization. It is therefore important for organizations to select and practice an appropriate project management approach.

Scrum is one of the most popular Agile frameworks. It is an adaptive, iterative, fast, flexible, and effective framework designed to deliver significant value quickly and throughout a project. Although the Scrum framework as defined in the *SBOK® Guide* is primarily used to deliver projects and create products, Scrum may also be used to manage the continuous maintenance of products and services, to track issues, and to manage changes.

Scrum ensures transparency in communication and creates an environment of collective accountability and continuous progress. The Scrum framework, as defined in the *SBOK® Guide*, is structured in such a way that it supports product and service development in all types of industries and in any type of project, irrespective of its complexity.

A key strength of Scrum lies in its use of cross-functional, self-organized, and empowered teams who divide their work into short, concentrated work cycles called Sprints.

Figure 1-1 depicts the Scrum Flow for one Sprint. It outlines the sequence of Scrum events, roles, and artifacts from Product Backlog to product increment, reflecting the iterative nature of Scrum.

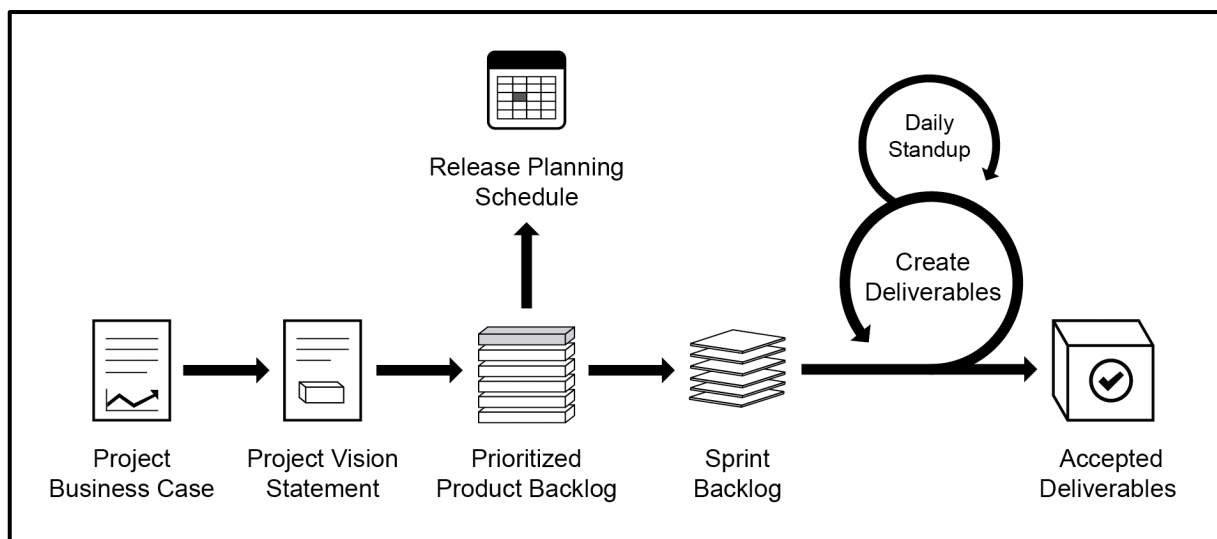


Figure 1-1: Scrum Flow for One Sprint

The Scrum cycle begins with a Stakeholder Meeting, during which the Project Vision is created. The Product Owner then develops a Prioritized Product Backlog which contains a prioritized list of business and project requirements written in the form of User Stories.

Each Sprint begins with a Sprint Planning Meeting during which high priority User Stories are considered for inclusion in the Sprint. A Sprint lasts between one and four weeks and involves the Scrum Team working to create potentially shippable deliverables or product increments. During the Sprint, short, highly focused Daily Standup Meetings are conducted where team members discuss daily progress. The Product Owner can assess completed deliverables during the Sprint and can accept the deliverables that meet the predefined Acceptance Criteria. Toward the end of the Sprint, a Sprint Review Meeting is held during which the Product Owner and relevant business stakeholders are provided a demonstration of the deliverables. The Sprint cycle ends with a Retrospect Sprint Meeting where the team members discuss ways they can improve the way they work and their performance as they move forward into the subsequent Sprint.

1.1.1 Brief History of Scrum

In the mid 80's, Hirotaka Takeuchi and Ikujiro Nonaka defined a flexible and all-inclusive product development strategy where the development team works as a unit to reach a common goal. They described an innovative approach to product development that they called a holistic or “rugby” approach, “where a team tries to go the distance as a unit, passing the ball back and forth.” The authors based their approach on manufacturing case studies from various industries. Takeuchi and Nonaka proposed that product development should not be like a sequential relay race but rather should be analogous to the game of rugby where the team works together, passing the ball back and forth as they move as a unit down the field. The rugby concept of a “Scrum” (where a group of players form together to restart the game) was introduced in this article to describe the authors' proposal that product development should involve “moving the Scrum downfield.”

Since then, several Scrum practitioners, experts, and authors have continued to refine the Scrum conceptualization and framework based on best practices. A major milestone in the Scrum journey was the creation of the original version of the *SBOK® Guide* in 2013. Over time, the *SBOK® Guide* has been continuously improved based on reviews and feedback provided by several thousand Scrum and Agile practitioners, including 5000+ faculty in 110+ countries who teach Scrum and Agile practices using the *SBOK® Guide* framework. This fourth edition is the product of further refinement that ensure the *SBOK® Guide* continues to remain valid and relevant in an ever-changing world.

The *SBOK® Guide* is now the industry standard for companies and practitioners interested in implementing Scrum or Agile practices. In recent years, Scrum has increased in popularity and is now the preferred project development approach for many organizations globally. To facilitate its application across multinational settings, the *SBOK® Guide* has been translated into multiple languages including Spanish, Portuguese, French, Italian, Arabic, Chinese, and Japanese. For more information about accessing translated versions, please visit www.scrumstudy.com.

1.2 Why Use Scrum?

Some of the key benefits of using Scrum in any project are:

1. **Adaptability**—Empirical process control and iterative delivery make projects adaptable and open to incorporating change.
2. **Transparency**—All information radiators like a Scrumboard and Sprint Burndown Chart are shared, leading to an open work environment.
3. **Continuous Feedback**—Continuous feedback is provided through the *Conduct Daily Standup* and *Demonstrate and Validate Sprint* processes.
4. **Continuous Improvement**—The deliverables are improved progressively Sprint by Sprint, through the *Refine Prioritized Product Backlog* process.
5. **Continuous Delivery of Value**—Iterative processes enable the continuous delivery of value through the *Ship Deliverables* process as frequently as the customer requires.
6. **Sustainable Pace**—Scrum processes are designed such that the people involved can work at a sustainable pace that they can, in theory, continue indefinitely.
7. **Early Delivery of High Value**—The *Create Prioritized Product Backlog* process ensures that the highest value requirements of the customer are satisfied first.
8. **Efficient Development Process**—Time-boxing and minimizing non-essential work leads to higher efficiency levels.
9. **Motivation**—The *Conduct Daily Standup* and *Retrospect Sprint* processes lead to greater levels of motivation among employees.
10. **Faster Problem Resolution**—Collaboration and colocation of cross-functional teams lead to faster problem solving.
11. **Effective Deliverables**—The *Create Prioritized Product Backlog* process and regular reviews after creating deliverables ensures effective deliverables to the customer.
12. **Customer Centric**—Emphasis on business value and having a collaborative approach to engage business stakeholders ensures a customer-oriented framework.
13. **High Trust Environment**—*Conduct Daily Standup* and *Retrospect Sprint* processes promote transparency and collaboration, leading to a high-trust work environment ensuring low friction among employees.
14. **Collective Ownership**—The *Commit User Stories* process allows team members to take ownership of the project and their work, leading to better quality.
15. **High Velocity**—A collaborative framework enables highly skilled cross-functional teams to achieve their full potential and high velocity.
16. **Innovative Environment**—The *Retrospect Sprint* and *Retrospect Release* processes create an environment of introspection, learning, and adaptability leading to an innovative and creative work environment.

1.2.1 Scalability of Scrum

To be effective, Scrum Teams should ideally have six to ten members. This practice may be the reason for the misconception that the Scrum framework can only be applied to small projects. However, the framework can easily be scaled for effective use in large projects, programs, and portfolios. In situations where the Scrum Team size exceeds ten people, multiple Scrum Teams can be formed to work on the project. The logical approach of the guidelines and principles in this framework can be used to manage projects of any size, spanning geographies and organizations. Large projects may have multiple Scrum Teams working in parallel making it necessary to synchronize and facilitate the flow of information and enhance communication. Large or complex projects are often implemented as part of a program or portfolio.

Details on Scaling Scrum for Large Projects are provided in chapter 13. Guidance on Scaling Scrum for the Enterprise is covered in chapter 14.

1.3 Purpose of the *SBOK® Guide*

The Scrum framework has proven to be the preferred project delivery framework to consistently deliver high business value and improve Returns on Investment. Scrum's focus on value-driven delivery helps Scrum Teams deliver results as early in the project as possible.

The *SBOK® Guide* was developed as a means to create a necessary guide for organizations and project management practitioners who want to implement Scrum, as well as those already doing so who want to make needed improvements to their processes. It is based on experience drawn from thousands of projects across a variety of organizations and industries. The contributions of many Scrum experts and project management practitioners have been considered in its development.

The *SBOK® Guide* is especially valuable:

- to Scrum Core Team members including:
 - Product Owners who want to fully understand the Scrum framework and particularly the customer or stakeholder-related concerns involving business justification, quality, change, and risk aspects associated with Scrum projects;
 - Scrum Masters who want to learn their specific role in overseeing the application of the Scrum framework to Scrum projects;
 - Scrum Team members who want to better understand Scrum processes and the associated used to create the project's product or service.
- as a comprehensive guide for all Scrum practitioners working on Scrum projects in any organization or industry.
- as a reference source for anyone interacting with the Scrum Core Team, including but not limited to the Portfolio Product Owner, Portfolio Scrum Master, Program Product Owner, Program Scrum Master, Scrum Guidance Body, and business stakeholders (i.e., sponsors, customers, and users);
- as a handbook for any person who has no prior experience or knowledge of Scrum framework but wants to learn more about the subject.

The content of the *SBOK® Guide* is also helpful for individuals preparing to write the following SCRUMstudy™ certification exams:

- Scrum Developer with AI Certified (SDC®)
- Scrum Master with AI Certified (SMC®)
- Scaled Scrum Master with AI Certified (SSMC™)
- SCRUMstudy Agile Master with AI Certified (SAMC™)
- Scrum Product Owner with AI Certified (SPOC®)
- Scaled Scrum Product Owner with AI Certified (SSPOC™)
- Expert Scrum Master with AI Certified (ESMC™)

1.4 SBOK® vs. other Scrum Books or Bodies of Knowledge

1.4.1 Free download of SBOK®

To support growth and standardization in Scrum frameworks and knowledge, Scrumstudy.com has made the Scrum Reference Guide (SBOK®) available for free on its website. Any Scrum practitioner or anyone interested in the field of Scrum can easily download and use the guide as a convenient reference for personal study or professional work.

Not sure about the quality of SBOK®? Just download it and see for yourself—*it's free!*

1.4.2 Free certifications, webinars, videos, study guides

The SBOK® is supported on Scrumstudy.com with free certifications, webinars, videos, and study guides—ideal for any professional seeking a fundamental understanding of Scrum or exploring a career in the field. The free certification will also give you a head start and recognition in the Scrum field.

Please visit Scrumstudy.com for more details.

1.4.3 Practical, Contemporary, and supports AI

Most Scrum books and Bodies of Knowledge are overly theoretical, lengthy, and lack practical examples or explanations of contemporary concepts—such as how Artificial Intelligence (AI) or automation tools can be used by Scrum practitioners to work more effectively. However, SBOK® is written with a focus on real-life issues and practical problems faced by Scrum practitioners, and how these can be addressed using modern tools and AI.

1.4.4 Practice Scrum in Real Life with Vabro.ai

Scrumstudy has partnered with Vabro.ai—an AI-enabled SaaS platform—to practically demonstrate how the theoretical concepts taught in SBOK® are applied using an AI-enabled tool.

For more details, please visit Scrumstudy.com.

1.4.5 80-20 Rule: (20% of SBOK® is Sufficient)

Most Scrum books are overly comprehensive and expect readers to go through the entire content to understand key concepts. However, SBOK® follows the 80-20 rule—meaning 80% of the core concepts can be understood by reading just 20% of the book. The remaining content can be referred to as needed when dealing with more complex Scrum tasks.

All five process chapters—*Initiate, Plan and Estimate, Implement, Review and Retrospect*, and *Release*—include mandatory inputs, tools, and outputs, and are ideal for those seeking a high-level understanding of Scrum. These essential sections make up only 20% of the book's content. The more detailed concepts are designed for experienced practitioners and can be easily accessed as needed while performing specific Scrum activities.

1.4.6 Easily organized and Fun to read

Unlike other Scrum books or Bodies of Knowledge (BOKs), the SBOK® is well-organized and engaging to read. Chapter 2 discusses the six principles of Scrum, while Chapters 3 to 7 focus on the five aspects of Scrum: Organization, Business Justification, Quality, Change, and Risk. The actual work performed by Scrum practitioners is structured into five chapters covering the Scrum processes—Initiate, Plan and Estimate, Implement, Review and Retrospect, and Release. These chapters include nineteen processes that provide step-by-step guidance to help Scrum practitioners carry out their work effectively.

1.4.7 Applicable to Organizations of all Sizes and Industries

The concepts in the Scrum Reference Guide (SBOK®) are applicable to the following:

- Scrum initiatives in any industry
- Products, services, or any other outcomes delivered to stakeholders
- Scrum initiatives of any size or complexity

The Scrum concepts in SBOK® can be effectively applied to initiatives across any industry—from small efforts or teams with as few as six team members to large, complex initiatives involving thousands of team members across multiple teams.

1.4.8 Aligns with other Frameworks including Kanban, OKRs, DevOps, Business Analysis

Unlike other Scrum books, which typically do not show how the Scrum function can interact with other popular frameworks such as Kanban, OKRs, DevOps and Business Analysis. SBOK® includes an Appendix to illustrate how the Scrum concepts align with other leading frameworks used by professionals in the industry.

1.4.9 Advanced Certifications for Senior Scrum Professionals

Multiple advanced certifications are available for senior Scrum professionals, based on the Scrum Reference Guide (SBOK®).

For more details, please visit [Scrumstudy.com](https://www.scrumstudy.com)

1.4.10 Available in 6 languages

The Scrum Reference Guide (SBOK®), along with its courses and certifications, is available in six languages: English, French, German, Italian, Spanish, and Portuguese.

For more details, please visit [Scrumstudy.com](https://www.scrumstudy.com)

1.4.11 2,500+ Partners in 50+ Countries, 300,000+ LinkedIn Group

Scrumstudy.com has partnered with VMEdU Inc.—a reputable professional learning and certification company that also collaborates with well-known brands such as KanbanStudy.com, OKRStudy.com, BALearning.com, 6SigmaStudy.com, and more. VMEdU has over 2,500 Authorized Training Partners in 50+ countries, a LinkedIn group with more than 300,000 members, and more than 2,000,000 certified students. This positions us as a global leader in the professional education field.

For more details, please visit [VMEdU.com](https://www.vmedu.com) or [Scrumstudy.com](https://www.scrumstudy.com)

1.5 Framework of the SBOK® Guide

The SBOK® Guide is broadly divided into the following three areas:

1. **Principles** covered in chapter 2, expand on the six principles which form the foundation on which Scrum is based—Empirical Process Control, Self-organization, Collaboration, Value-based Prioritization, Time-boxing, and Iterative Development.
2. **Aspects** covered in chapters 3 through 7, describe the five aspects that are important considerations for all Scrum projects—Organization, Business Justification, Quality, Change, and Risk.
3. **Processes** covered in chapters 8 through 12 include the nineteen fundamental Scrum processes and their associated inputs, tools, and outputs. Chapter 13 covers the additional inputs, tools, and outputs needed for scaling Scrum for large projects, whereas chapter 14 covers the additional processes needed for scaling Scrum for the enterprise.

Figure 1-2 illustrates the SBOK® Guide Framework, showing the structured relationship between Scrum principles, aspects, and processes, offering a comprehensive framework for applying Scrum.

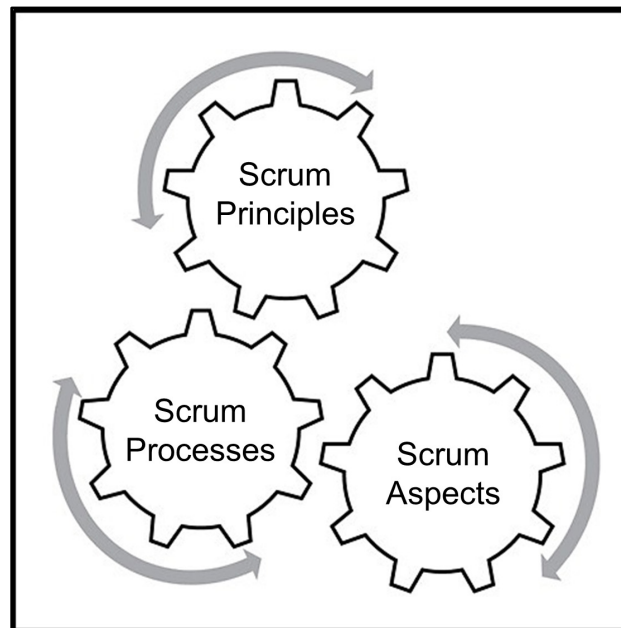


Figure 1-2: SBOK® Guide Framework

1.5.1 How to Use the *SBOK® Guide*?

The *SBOK® Guide* can be used as a reference and knowledge guide by both experienced Scrum and other product and service development practitioners, as well as by people with no prior experience or knowledge of Scrum or project management approach. The contents are organized for easy reference by the three Scrum Core Team roles: Product Owner, Scrum Master, and Scrum Team.

The chapters covering the six Scrum principles (chapter 2) and five Scrum aspects (chapter 3 through 7) include a Roles Guide. This guide provides direction regarding the relevance of each section in the chapter to the Scrum Core Team roles.

In order to facilitate the best application of the Scrum framework, the *SBOK® Guide* has clearly differentiated mandatory inputs, tools, and outputs, from non-mandatory or optional ones. Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, while others with no asterisks are optional. It is recommended that those being introduced to Scrum focus primarily on the mandatory inputs, tools, and outputs; while more experienced practitioners should read the entire process chapters to benefit from the optional best practice inputs, tools, and outputs suggested.

Scrum is a framework that is not meant to be prescriptive, which means there is room for flexibility in its application. All the fundamental Scrum processes detailed in the *SBOK® Guide* (chapters 8 through 12) are required for every Scrum project but should be applied based on the specific needs of the organization, project, product, and/or team. Additional inputs, tools, and outputs would apply when Scaling Scrum for Large Projects (chapter 13) and additional processes would apply when Scaling Scrum for the Enterprise (chapter 14).

1.5.2 Scrum Principles

Scrum principles are the core guidelines for applying the Scrum framework and should mandatorily be used in all Scrum projects. The six Scrum principles presented in chapter 2 are:

1. Empirical Process Control
2. Self-organization
3. Collaboration
4. Value-based Prioritization
5. Time-boxing
6. Iterative Development

Figure 1-3 represents the key Scrum principles such as empirical process control, self-organization, collaboration, and iterative development, forming the foundational philosophy of Scrum.

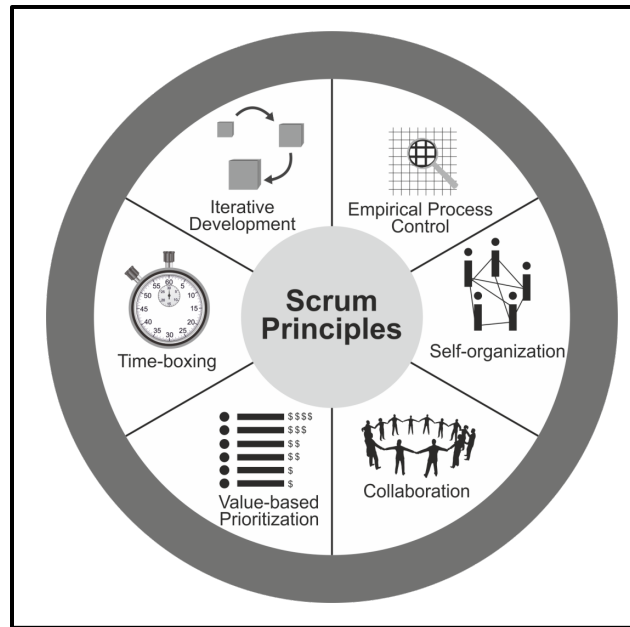


Figure 1-3: Scrum Principles

Scrum principles can be applied to any type of project in any organization and must be adhered to in order to ensure effective implementation of the Scrum framework. Scrum Principles are non-negotiable and must be applied as specified in the *SBOK® Guide*. Keeping the principles intact and using them appropriately instills confidence in the Scrum framework with regard to attaining the objectives of the project. The Scrum aspects and processes, however, can be modified to meet the requirements of the project or the organization.

1. **Empirical Process Control**—This principle emphasizes the core philosophy of Scrum based on the three main ideas of transparency, inspection, and adaptation. Empirical process control aids learning through experimentation, especially when the problem is not well defined or when there are no clear solutions.
2. **Self-organization**—This principle focuses on today's workers, who deliver significantly greater value when self-organized, and this results in better team buy-in and shared ownership; and an innovative and creative environment which is more conducive for growth.
3. **Collaboration**—This principle focuses on the three core dimensions related to collaborative work: awareness, articulation, and appropriation. It also advocates project delivery as a shared value-creation process with teams working and interacting together, as well as with the customer and other business stakeholders, to deliver the greatest value.
4. **Value-based Prioritization**—This principle highlights the focus of Scrum to deliver maximum business value, from early in the project and continuing throughout.
5. **Time-boxing**—This principle describes how time is considered a limiting constraint in Scrum and used to help effectively manage project planning and execution. Time-boxed elements in Scrum include Sprints, Daily Standup Meetings, Sprint Planning Meetings, Sprint Review Meetings, and Retrospect Sprint Meetings.

6. **Iterative Development**—This principle defines iterative development and emphasizes how to better manage changes and build products that satisfy customer needs. It also delineates the Product Owner's and organization's responsibilities related to iterative development.

1.5.3 Scrum Aspects

The Scrum aspects must be addressed and managed throughout a Scrum project. The five Scrum aspects presented in chapter 3 through 7 are:

1.5.3.1 Organization

Understanding defined roles and responsibilities in a Scrum project is important for ensuring the successful implementation of Scrum. Scrum roles fall into two broad categories:

1. **Core Roles**—Core roles are those roles which are mandatorily required for producing the project's product or service. Individuals who are assigned core roles are fully committed to the project and are responsible for the success of each sprint and that of the project as a whole.

Core roles comprise the Scrum Core Team members, which include:

- The **Product Owner** is the person responsible for achieving maximum business value for the project. He or she is also responsible for articulating customer requirements and maintaining business justification for the project. The Product Owner represents the Voice of the Customer.
- The **Scrum Master** is a facilitator who ensures that the Scrum Team is provided with an environment conducive to completing the project successfully. The Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved in the project; clears impediments for the team; and ensures that Scrum processes are being followed.
- The **Scrum Team** is the group or team of people who are responsible for understanding the requirements specified by the Product Owner and creating the deliverables of the project.

2. **Non-core Roles**—Non-core roles are those roles that are not mandatorily required for the Scrum project. They may include team members who are interested in the project but have no formal role in the project team. These individuals may interface with the team but may not be responsible for the success of the project. The non-core roles should be considered in any Scrum project.

Non-core roles include the following:

- **Business Stakeholder(s)**, which is a collective term that includes customers, users, and sponsors who frequently interface with the Scrum Core Team and also influence the project throughout the project's development. Most importantly, it is for the business stakeholders that the project produces collaborative benefits.

Business stakeholders are a subset of all stakeholders in a Scrum project - stakeholders include all individuals and groups affected by the Scrum project, both within and outside the organization (e.g., all core and non-core roles, vendors, internal groups, experts, and so on).

- **Scrum Guidance Body (SGB)** is an optional role, which generally consists of a set of documents and/or a group of experts who are typically involved with defining objectives related to quality, government regulations, security, and other key organizational parameters. The SGB guides the work carried out by the Product Owner, Scrum Master, and Scrum Team.
- **Vendors**, include external individuals or organizations that provide products and/or services that are not within the core competencies of the project organization.

Figure 1-4 shows the organizational structure in Scrum, highlighting roles like the Product Owner, Scrum Master, and Scrum Team within a flat, collaborative hierarchy.

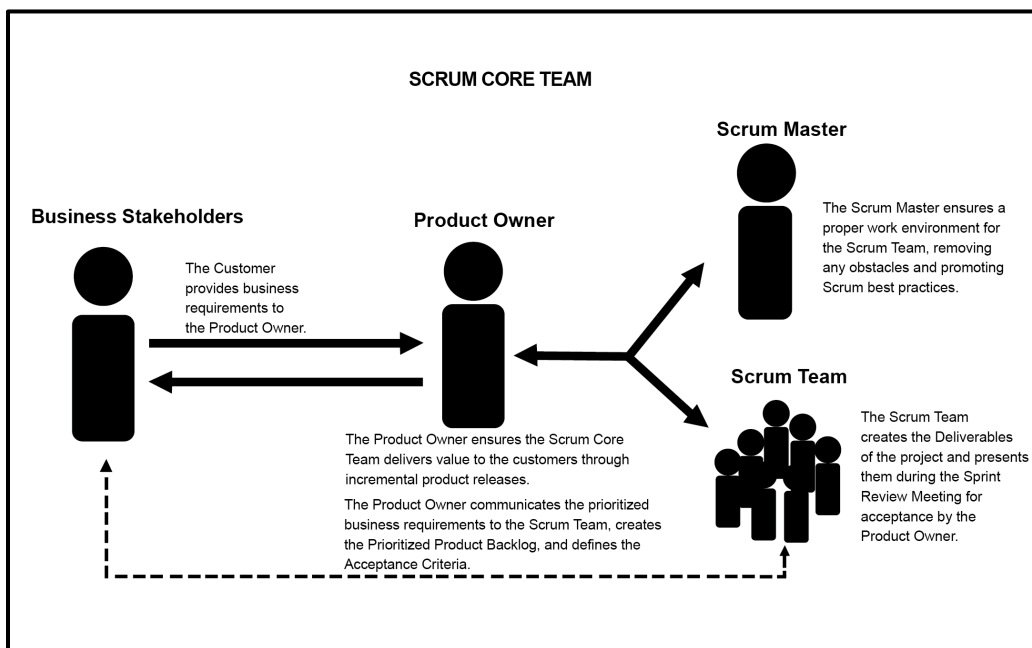


Figure 1-4: Organization in Scrum

The Organization aspect of Scrum also addresses the team structure requirements to implement Scrum in large projects, programs, and portfolios.

1.5.3.2 Business Justification

It is important for an organization to perform a proper business assessment prior to starting any project. This helps key decision makers understand the business need for a change or for a new product or service, the justification for moving forward with a project, and its viability.

Business justification in Scrum is based on the concept of value-driven delivery. One of the key characteristics of any project is the uncertainty of results or outcomes. It is impossible to guarantee project success at completion, irrespective of the size or complexity of a project. Considering this uncertainty of achieving success, Scrum attempts to start delivering results as early in the project as possible.

This speedy delivery of results, and thereby value, provides an opportunity for reinvestment and proves the worth of the project to interested business stakeholders.

Scrum's adaptability allows the project's objectives and processes to change if its business justification changes. It is important to note that although the Product Owner is primarily responsible for business justification, other team members contribute significantly.

1.5.3.3 Quality

In Scrum, quality is defined as the ability of the completed product or deliverables to meet the Acceptance Criteria and achieve the business value expected by the customer.

To ensure a project meets quality requirements, Scrum adopts an approach of continuous improvement whereby the team learns from experience and stakeholder engagement to constantly keep the Prioritized Product Backlog updated with any changes in requirements. The Prioritized Product Backlog is simply never complete until the closure or termination of the project. Any changes to the requirements reflect changes in the internal and external business environment and allow the team to continually work and adapt to achieve those requirements.

Since Scrum requires work to be completed in increments during Sprints, this means that errors or defects get noticed earlier through repetitive quality testing, rather than when the final product or service is near completion. Moreover, important quality-related tasks (e.g., development, testing, and documentation) are completed as part of the same Sprint by the same team—this ensures that quality is inherent in any deliverable created as part of a Sprint. Such deliverables from Scrum projects, which are potentially shippable, are referred to as 'Done.' Thus, continuous improvement with repetitive testing optimizes the probability of achieving the expected quality levels in a Scrum project. Constant discussions between the Scrum Core Team and business stakeholders (including customers and users) with actual increments of the product being delivered at the end of every Sprint, ensures that the gap between customer expectations from the project and actual deliverables produced is constantly reduced.

The Scrum Guidance Body may also provide guidelines about quality which may be relevant to all Scrum projects in the organization.

1.5.3.4 Change

Every project, regardless of its method or framework used, is exposed to change. It is imperative that project team members understand that the Scrum development processes are designed to embrace change. Organizations should try to maximize the benefits that arise from change and minimize any negative impacts through diligent change management processes in accordance with the principles of Scrum.

A primary principle of Scrum is its acknowledgement that a) business stakeholders (e.g., customers, users, and sponsors) change their minds about what they want and need throughout a project (sometimes referred to as “requirements churn”) and b) it is very difficult, if not impossible, for business stakeholders to define all requirements during project initiation.

Scrum projects welcome change by using short, iterative Sprints that incorporate customer feedback on each Sprint’s deliverables. This enables the customer to regularly interact with the Scrum Team members, view deliverables as they are ready, and change requirements if needed earlier in the Sprint.

Also, the portfolio or program management teams can respond to Change Requests pertaining to Scrum projects applicable at their level.

1.5.3.5 Risk

Risk is defined as an uncertain event or set of events that can affect the objectives of a project and may contribute to its success or failure. Risks that are likely to have a positive impact on the project are referred to as opportunities, whereas threats are risks that could affect the project in a negative manner. Managing risk must be done proactively, and it is an iterative process that should begin at project initiation and continue throughout the project’s lifecycle. The process of managing risks should follow some standardized steps to ensure that risks are identified, evaluated, and a proper course of action is determined and acted upon accordingly.

Risks should be identified, assessed, and responded to on the basis of two factors—the probability of each risk’s occurrence and the possible impact in the event of such occurrence. Risks with a high probability and impact value (determined by multiplying both factors), should be addressed before those with a lower value. In general, once a risk is identified, it is important to understand the risk with regards to the probable causes and the potential effects if the risk occurs.

1.5.4 Scrum Processes

Scrum processes address the specific activities and flow of a Scrum project. However, Scrum processes are not sequential but are iterative in nature and may overlap with one another. In total there are nineteen fundamental Scrum processes that apply to all projects. These nineteen processes are grouped into five phases as shown in Table 1-1. Details on each phase and its corresponding processes are presented in chapters 8 through 12 of the *SBOK® Guide*.

Chapter	Phase	Fundamental Scrum Processes
8	Initiate	1. Create Project Vision 2. Identify Scrum Master and Business Stakeholder(s) 3. Form Scrum Team 4. Develop Epic(s)

		5. Create Prioritized Product Backlog 6. Conduct Release Planning
9	Plan and Estimate	7. Create User Stories 8. Estimate User Stories 9. Commit User Stories 10. Identify Tasks 11. Estimate Tasks 12. Update Sprint Backlog
10	Implement	13. Create Deliverables 14. Conduct Daily Standup 15. Refine Prioritized Product Backlog
11	Review and Retrospect	16. Demonstrate and Validate Sprint 17. Retrospect Sprint
12	Release	18. Ship Deliverables 19. Retrospect Release

Table 1-1: Summary of Fundamental Scrum Processes

The chapters that focus on each phase describe each process in detail including their associated inputs, tools, and outputs. In each process, some inputs, tools, and outputs are mandatory (those with an asterisk [*] after their names), while others are optional. Whether to include the optional inputs, tools, and/or outputs depends on the particular project, organization, or industry. Inputs, tools, and outputs denoted with an asterisk are considered mandatory or critical to the successful implementation of Scrum in any organization.

For large-scale Scrum projects that require coordination across multiple teams, there are additional inputs, tools, and outputs needed. These are defined in Chapter 13—Scaling Scrum for Large Projects.

Also, there are additional processes defined when implementing Scrum at the enterprise level. These are covered in Chapter 14—Scaling Scrum for the Enterprise.

1.5.4.1 Initiate Phase

The processes relevant to the Initiate phase are as follows:

1. *Create Project Vision*—In this process, the project business case is reviewed to create a Project Vision Statement that will serve as the inspiration and provide focus for the entire project. The Product Owner is identified in this process. The company may also choose to evaluate and select an AI-powered Scrum project tool that can significantly reduce the complexity of setting up the Scrum initiative and increase the productivity of Scrum projects within the organization.

2. *Identify Scrum Master and Business Stakeholder(s)*—In this process, the Scrum Master and business stakeholders are identified using specific selection criteria.
3. *Form Scrum Team*—In this process, Scrum Team members are identified. Normally the Product Owner has the primary responsibility of selecting team members but often does so in collaboration with the Scrum Master.
4. *Develop Epic(s)*—In this process, the Project Vision Statement serves as the basis for developing Epics. User Group Meetings may be held to discuss appropriate Epics.
5. *Create Prioritized Product Backlog*—In this process, Epic(s) are refined, elaborated, and then prioritized to create a Prioritized Product Backlog for the project. The Done Criteria is also established at this point.
6. *Conduct Release Planning*—In this process, the Product Owner with support from the Scrum Team develops a Release Planning Schedule, which is essentially a phased deployment schedule that can be shared with the project's business stakeholders. Length of Sprint is also determined in this process.

1.5.4.2 Plan and Estimate Phase

The processes relevant to the Plan and Estimate phase are as follows:

7. *Create User Stories*—In this process, User Stories and their related Acceptance Criteria are created by the Product Owner and incorporated into the Prioritized Product Backlog. User Stories are designed to ensure that the customer's requirements are clearly depicted and can be fully understood by all business stakeholders.
8. *Estimate User Stories*— In this process, the Scrum Team, supported by the Scrum Master, estimates the User Stories and identifies the effort required to develop the functionality described in each User Story.
9. *Commit User Stories*—In this process, the Scrum Team commits to deliver Product Owner-approved User Stories for a Sprint. The results of this process are the committed User Stories and the Sprint Backlog.
10. *Identify Tasks*—In this process, the committed User Stories are broken down into specific tasks and compiled into a Task List.
11. *Estimate Tasks*—In this optional process, the Scrum Core Team estimates the effort required to accomplish each task in the Task List.
12. *Update Sprint Backlog*—In this process, the Scrum Core Team updates the Sprint Backlog with further details about the tasks as part of the Sprint Planning Meeting.

1.5.4.3 Implement Phase

The processes relevant to the Implement phase are as follows:

13. *Create Deliverables*—In this process, the Scrum Team works on the tasks in the Sprint Backlog to create Sprint Deliverables. A Scrumboard is often used to track the work and activities being carried out. Issues or problems being faced by the Scrum Team should be updated in an Impediment Log.
14. *Conduct Daily Standup*—In this process, every day a highly focused, Time-boxed meeting, referred to as the Daily Standup Meeting, is conducted. This is the forum for the Scrum Team to update each other on their individual progress and any impediments they may be facing.
15. *Refine Prioritized Product Backlog*—In this process, the Prioritized Product Backlog is continuously updated and maintained. A Prioritized Product Backlog Review Meeting is held, in which any changes or updates to the backlog are discussed and incorporated into the Prioritized Product Backlog as appropriate.

1.5.4.4 Review and Retrospect Phase

The processes relevant to the Review and Retrospect phase are as follows:

16. *Demonstrate and Validate Sprint*—In this process, the Scrum Team demonstrates the Sprint deliverables to the Product Owner and relevant business stakeholders in a Sprint Review Meeting. The purpose of this meeting is to secure approval and acceptance of the Sprint User Stories by the Product Owner.
17. *Retrospect Sprint*—In this process, the Scrum Master and Scrum Team meet to discuss the lessons learned throughout the Sprint. This information is documented and should be applied to future Sprints. Often, as a result of this discussion, there may be agreed actionable improvements or updated Scrum Guidance Body recommendations.

1.5.4.5 Release Phase

The processes relevant to the Release phase are as follows:

18. *Ship Deliverables*—In this process, all deliverables from the accepted User Stories of previously completed Sprints are delivered or transitioned to the relevant business stakeholders. A formal Working Deliverables Agreement documents the successful completion of the release.
19. *Retrospect Release*—In this process which completes a release, business stakeholders and Scrum Core Team members assemble to reflect on the release and identify, document, and internalize the lessons learned. Often, these lessons lead to the documentation of agreed actionable improvements to be implemented in future projects.

1.5.4.6 Scrum Meetings or Ceremonies

Scrum meetings or ceremonies play a critical part in effective implementation of the Scrum framework and are a key means through which Scrum principles are implemented. The important Scrum meetings and associated processes in which these meetings are conducted are summarized in Table 1-2.

Scrum Meetings	Scrum Processes
Project Vision Meeting	• Create Project Vision
User Group Meeting	• Develop Epic(s) • Create User Stories
Focus Group Meeting	• Develop Epics • Create User Stories
Release Planning Session or Meeting	• Conduct Release Planning
Product Backlog Review Meeting	• Estimate User Stories • Refine Prioritized Product Backlog
Sprint Planning Meeting	• Estimate User Stories • Commit User Stories • Identify Tasks • Estimate Tasks • Update Sprint Backlog
Daily Standup Meeting	• Conduct Daily Standup
Sprint Review Meeting	• Demonstrate and Validate Sprint
Retrospect Sprint Meeting	• Retrospect Sprint
Retrospect Release Meeting	• Retrospect Release

Table 1-2: Scrum Meetings and Processes

1.5.5 Scrum for Large Projects

When dealing with large projects requiring the efforts of multiple (four or more) Scrum Teams with multiple Product Owners and multiple Scrum Masters, the fundamental processes defined in chapters 8 through 12, remain valid, but some additional considerations and updates to inputs, tools, and outputs may be required. This may include additional coordination and synchronization needs. The impacts to the fundamental Scrum processes when scaling Scrum to large projects are described in chapter 13.

The definition of what constitutes a large project usually depends on the organization and/or the complexity of the projects being undertaken. A key criterion for whether a project is considered small versus large is whether the project requires multiple Scrum Masters and/or multiple Product Owners. If the project requires just one Scrum Master and one Product Owner, then these individuals can normally handle any additional communication and synchronization efforts required by the project.

1.5.6 Scrum for the Enterprise

When applying Scrum at an Enterprise level (such as to a program or portfolio), there may be several hundred Scrum Teams, with several thousand people working on multiple projects within programs and/or portfolios throughout the organization. Applying Scrum at a program or portfolio level will have certain impacts on the underlying projects. In general, the Scrum projects should still be executed using the fundamental Scrum processes discussed in chapters 8 through 12 for typical small projects, while incorporating the additional considerations outlined in chapter 13 for large projects (that have multiple Product Owners and/or Scrum Masters).

Some of the challenges that arise at the program or portfolio level are similar to those that come up when applying Scrum to a large project. The synchronization between teams and the overall collaboration are usually the biggest challenges with a large Scrum project—this is also a challenge when applying Scrum at a program or portfolio level. However, the biggest challenges when applying Scrum at the program or portfolio level usually occur on the business side because the business priorities of different projects conflict with each other and may also conflict with the overall goals of the program or portfolio. These priorities and goals must be aligned.

When implementing Scrum at the enterprise level, there are not only additional inputs, tools, and outputs, as in a large Scrum project, there are also specific additional processes that are required to address the additional prioritization, alignment, and coordination efforts. These additional considerations are discussed in chapter 14.

1.6 Scrum vs. Traditional Project Management

Table 1-3 summarizes many of the differences between Scrum and traditional project management models.

	Scrum	Traditional Project Management
Emphasis is on	People	Processes
Documentation	Minimal—only as required	Comprehensive
Process style	Iterative	Linear
Upfront planning	Low	High
Prioritization of requirements	Based on business value and regularly updated	Fixed in the project plan
Quality Assurance	Customer centric	Process centric
Organization	Self-organized	Managed
Management style	Decentralized	Centralized
Change	Updates to Prioritized Product Backlog	Formal Change Management System
Leadership	Supporting	Command and control
Performance measurement	Business value	Plan conformity
Return on Investment (ROI)	Early/throughout project life cycle	End of project life cycle
Customer involvement	High throughout the project	Varies depending on the project lifecycle

Table 1-3: Scrum vs. Traditional Project Management

2. PRINCIPLES

2.1 Introduction

Scrum principles are the foundation on which the Scrum framework is based. The principles of Scrum can be applied to any type of project or organization, and they must be adhered to in order to ensure appropriate application of Scrum. The aspects and processes of Scrum can be modified to meet the requirements of the project, or the organization using it, but Scrum principles are non-negotiable and must be applied as described in the framework presented in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*. Keeping the principles intact and using them appropriately instills confidence in the user of the Scrum framework with regard to attaining the objectives of the project. Principles are considered to be the core guidelines for applying the Scrum framework. *Principles*, as defined in the *SBOK® Guide*, are applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams. This chapter is divided into the following sections:

2.1.1 Roles Guide—This section outlines which section or subsection is most relevant for each of the core Scrum roles of Product Owner, Scrum Master, and Scrum Team.

2.2 Empirical Process Control—This section describes the first principle of Scrum, and the three main ideas of transparency, inspection, and adaptation.

2.3 Self-organization—This section highlights the second principle of Scrum, which focuses on today’s workers, who deliver significantly greater value when self-organized, and this results in better team buy-in and shared ownership; and an innovative and creative environment which is more conducive for growth.

2.4 Collaboration—This section emphasizes the third principle of Scrum where product development is a shared value-creation process that needs all business stakeholders working and interacting together to deliver the greatest value. It also focuses on the core dimensions of collaborative work: awareness, articulation, and appropriation.

2.5 Value-based Prioritization—This section presents the fourth principle of Scrum, which highlights the Scrum framework’s drive to deliver maximum business value in a minimum time span.

2.6 Time-boxing—This section explains the fifth principle of Scrum which treats time as a limiting constraint. It also covers the Sprint, Daily Standup Meeting, and the various other Sprint-related meetings such as the Sprint Planning Meeting, Sprint Review Meeting, and Retrospect Sprint Meeting, all of which are Time-boxed.

2.7 Iterative Development—This section addresses the sixth principle of Scrum which emphasizes that iterative development helps to better manage changes and build products that satisfy customer needs.

2.8 Scrum vs. Traditional Project Management—This section highlights the key differences between the Scrum principles and traditional project management (Waterfall model) principles and explains how Scrum works better in today's fast-changing world.

2.1.1 Roles Guide

All the sections in this chapter are important for all the Scrum Core Team roles—Product Owner, Scrum Master, and Scrum Team. A clear understanding of the Scrum principles by all business stakeholders is essential to make Scrum framework a success in any organization.

2.2 Empirical Process Control

In Scrum, decisions are made based on observation and experimentation rather than on detailed upfront planning. Empirical process control aids learning through experimentation when the problem is not well defined or when there are no clear solutions. Empirical process control relies on the three main ideas of transparency, inspection, and adaptation.

2.2.1 Transparency

Transparency allows all facets of any Scrum process to be observed by anyone. This promotes an easy and transparent flow of information throughout the organization and creates an open work culture. In Scrum, transparency is depicted through the following:

- A Project Vision Statement which can be viewed by all business stakeholders and the Scrum Team
- An open Prioritized Product Backlog with prioritized User Stories that can be viewed by everyone, both within and outside the Scrum Team
- A Sprint Backlog which may be used to list all the tasks, associated with the committed User Stories, to be executed by the Scrum Team in the current Sprint
- A Release Planning Schedule which may be used to coordinate work across multiple Scrum Teams and other business stakeholders
- Clear visibility into the team's progress through the use of a Scrumboard, Burndown Chart, and other information radiators
- Sprint Planning Meetings during which the Scrum Team estimates the effort needed to deliver top priority User Stories and commits to a set of User Stories for completion in the Sprint
- Daily Standup Meetings conducted during the *Conduct Daily Standup* process, in which all team members report what they have done the previous day, what they plan to do today, and any problems preventing them from completing their tasks in the current Sprint
- Sprint Review Meetings conducted during the *Demonstrate and Validate Sprint* process, in which the Scrum Team demonstrates the potentially shippable Sprint Deliverables to the Product Owner and business stakeholders
- Retrospect Sprint Meetings conducted after the Sprint Review Meetings on the final day of the Sprint during which the Scrum Team discusses improvement opportunities for future Sprints
- A Release Planning Meeting or session is conducted to enable the Scrum Team to have an overview of the planned releases and delivery schedule for the product they are developing

Figure 2-1 summarizes the concept of transparency in Scrum, emphasizing open communication, visible artifacts, and accessible progress tracking for all team members.

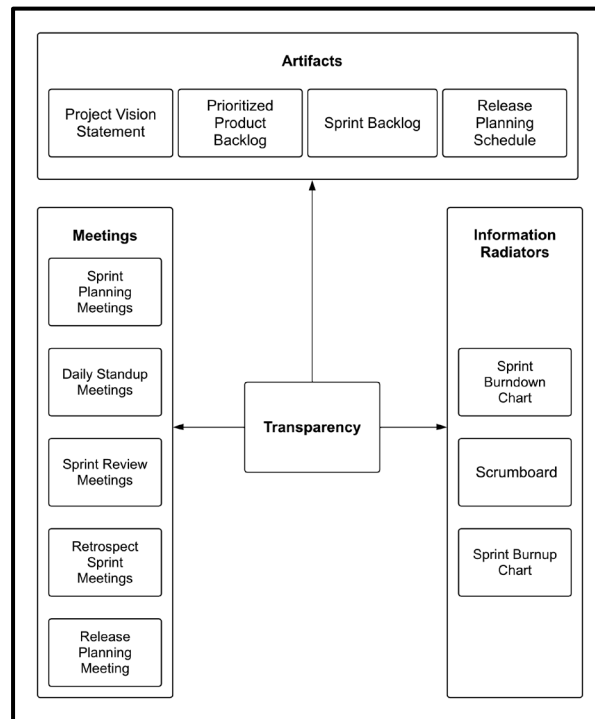


Figure 2-1: Transparency in Scrum

Figure 2-2 is an interface from Vabro, showing a Prioritized Product Backlog. It promotes transparency by allowing team members to access and manage backlog items openly.

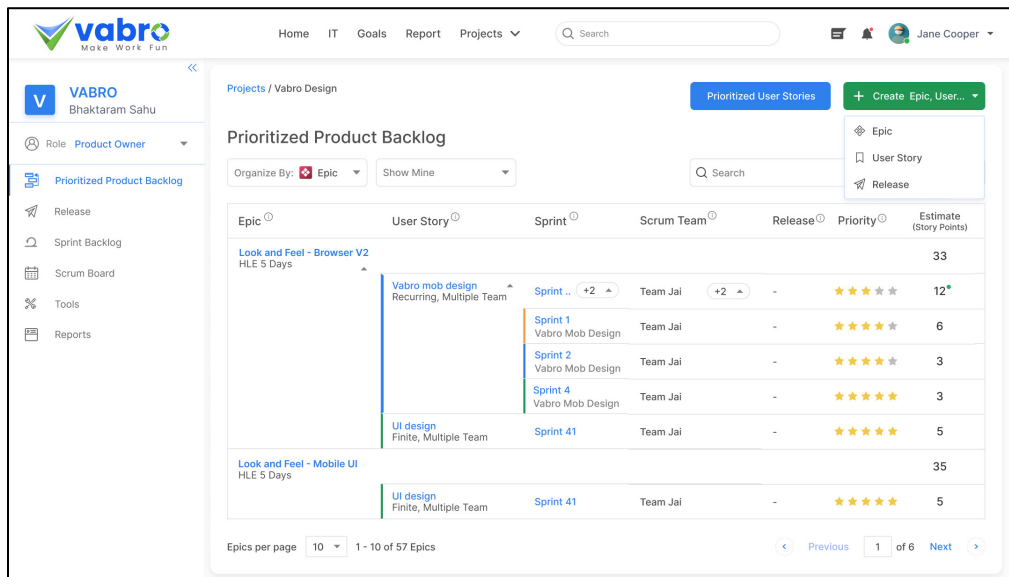


Figure 2-2: Transparency Facilitated through Product Backlog (Source: Vabro)

Figure 2-3 is an interface from Jira, showing an open Product Backlog. It supports Scrum transparency by making priorities and task statuses accessible across the team.

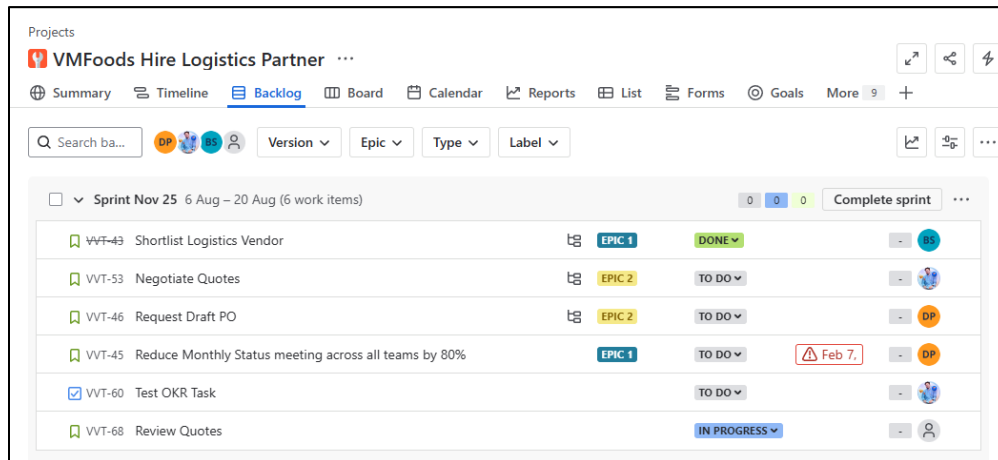


Figure 2-3: Open Product Backlog used for Transparency (Source: Jira)

2.2.2 Inspection

Inspection in Scrum is depicted through the following:

- Use of a common Scrumboard and other information radiators that show the progress of the Scrum Team on completing the tasks in the current Sprint.
- Collection of feedback from the customer and other business stakeholders during the *Develop Epic(s)*, *Create Prioritized Product Backlog*, *Conduct Release Planning*, and *Refine Prioritized Product Backlog* processes.
- Inspection and approval of the deliverables by the Product Owner and the customer in the *Demonstrate and Validate Sprint* process.

Figure 2-4 illustrates the concept of inspection in Scrum, highlighting how regular reviews during Daily Scrums and Sprint Reviews support continuous improvement.

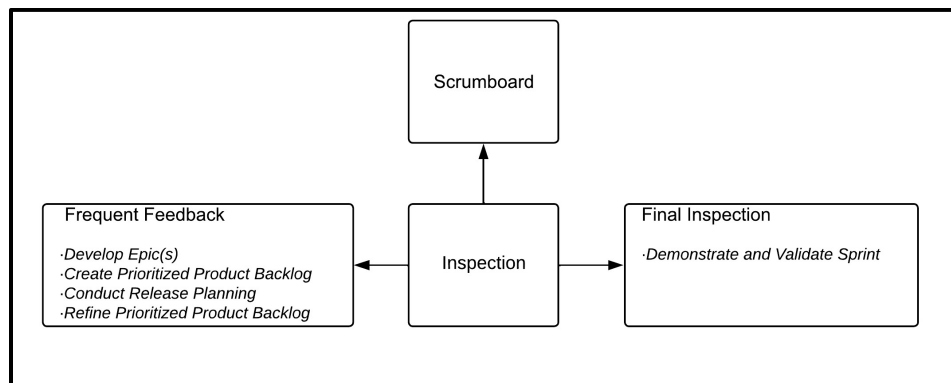


Figure 2-4: Inspection in Scrum

Figure 2-5 is an interface from Vabro, showing how AI-powered tools support transparency through real-time tracking, visual updates, and task visibility.

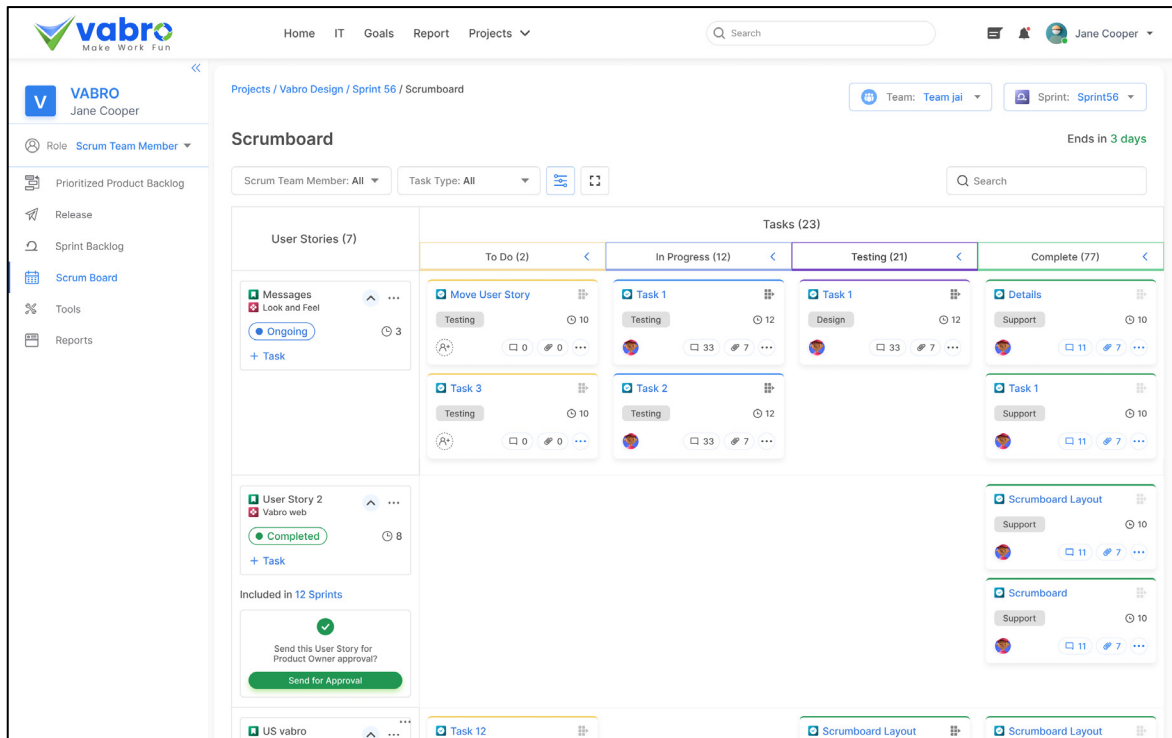


Figure 2-5: Transparency in Scrum through the use of AI-powered Scrum Project Tool (Source: Vabro)

Figure 2-6 is an interface from Jira, depicting how AI tools enable transparency by showing live task statuses, comments, and updates across the project board.

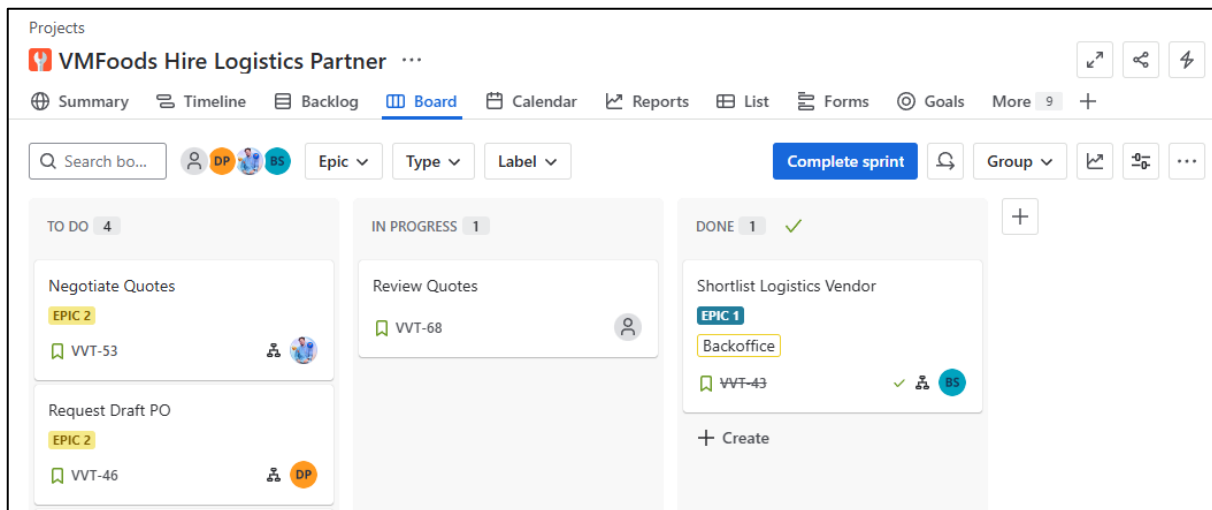


Figure 2-6: Transparency in Scrum through the use of AI-powered Scrum Project Tool (Source: Jira)

2.2.3 Adaptation

Adaptation happens as the Scrum Core Team and business stakeholders learn through transparency and inspection and then adapt by making improvements in the work they are doing. Some examples of opportunities for adaptation in the Scrum framework include:

- In Daily Standup Meetings, Scrum Team members openly discuss impediments to completing their tasks and seek help from other team members. More experienced members in the Scrum Team also mentor those with less experience in knowledge of the project or technology.
- Risk identification is performed and iterated throughout the project. Identified risks become inputs to several Scrum processes including *Create Prioritized Product Backlog*, *Refine Prioritized Product Backlog*, and *Demonstrate and Validate Sprint*.
- Improvements can also result in Change Requests, which are discussed and approved during the *Develop Epic(s)*, *Create Prioritized Product Backlog*, and *Refine Prioritized Product Backlog* processes.
- The Scrum Guidance Body interacts with Scrum Team members during the *Create User Stories*, *Estimate Tasks*, *Create Deliverables*, and *Refine Prioritized Product Backlog* processes to offer guidance and also provide expertise as required.
- In the *Retrospect Sprint* process, agreed actionable improvements are determined based on the outputs from the *Demonstrate and Validate Sprint* process.
- In the Retrospect Release Meeting, participants document lessons learned and perform reviews looking for opportunities to improve processes and address inefficiencies.

Figure 2-7 shows the process of adaptation in Scrum, illustrating how feedback and inspection results lead to real-time process and backlog adjustments.

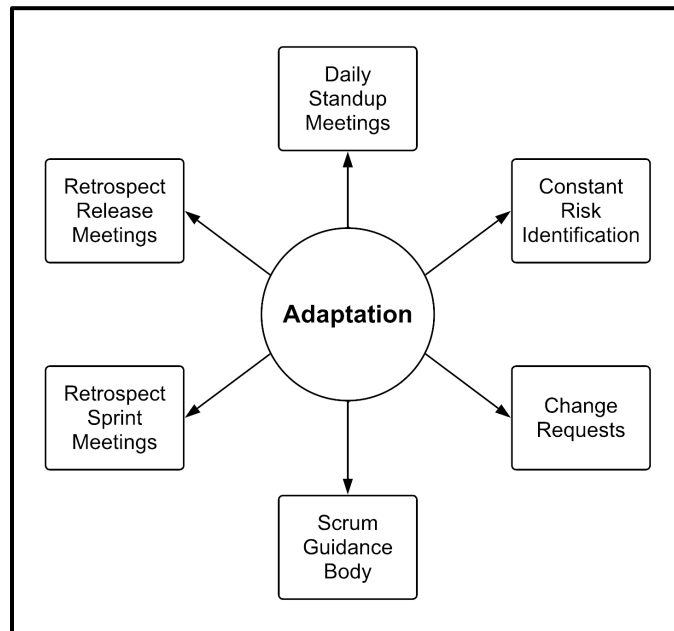


Figure 2-7: Adaptation in Scrum

With other methods, like the traditional Waterfall model, considerable planning needs to be done in advance, and the customer does not review product components until near the end of a phase, or the end of the entire project. This method often presents huge risks to the project's success because it may have more potential for significantly impacting project delivery and customer acceptance. The customer's interpretation and understanding of the finished product may be vastly different from what was actually understood and produced by the team, and this may not be known until extremely late in the project's development.

Figure 2-8 highlights challenges in traditional project management, such as delayed feedback, siloed teams, and rigid structures, contrasting with Scrum's adaptive approach.

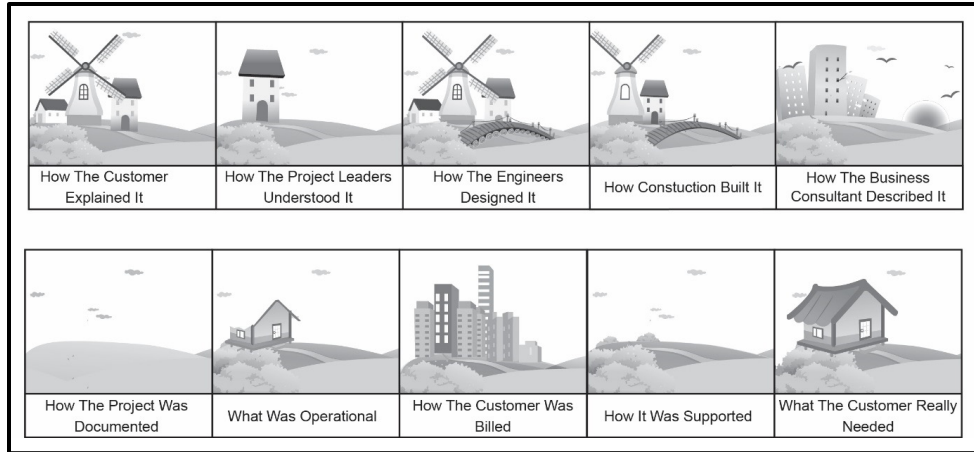


Figure 2-8: Challenges in Traditional Project Management

2.3 Self-organization

Scrum practices embrace the idea that employees are self-motivated and seek to accept greater responsibility. So, employees will deliver greater value when self-organized.

The preferred leadership style in Scrum is “supporting leadership,” which emphasizes achieving results by focusing on the needs of the Scrum Team. See section 3.9.4 for a discussion of various leadership and management styles.

Self-organization does not mean that team members are allowed to act in any manner that they want to. Once the Project Vision is defined in the *Create Project Vision* process, the Product Owner, Scrum Master, and Scrum Team get identified. Also, the Scrum Core Team itself works very closely with relevant business stakeholders for refining requirements better as they go through the *Develop Epic(s)* and *Create User Stories* process. Team expertise is used to assess the inputs needed to execute the planned work of the project. This judgment and expertise are applied to all technical and management aspects of the project during the *Create Deliverables* process.

Although prioritization is primarily done by the Product Owner who represents the Voice of Customer, the self-organized Scrum Team is involved in task breakdown and estimation during the *Identify Tasks* and *Estimate Tasks* processes. During these processes, each team member is responsible for determining what work he or she will be doing. The Scrum Team also helps the Product Owner identify risks and dependencies. During the execution of a Sprint, if team members need any help with completing their tasks, Scrum addresses this through the regular interaction mandatory with the Daily Standup Meetings. The Scrum Team itself interacts with other teams through the Scrum of Scrums (SoS) Meetings and can look for additional guidance as required from the Scrum Guidance Body.

Finally, the Scrum Team and Scrum Master work closely to demonstrate the product increment created during the Sprint in the *Demonstrate and Validate Sprint* process where properly completed deliverables are accepted. Since the deliverables are potentially shippable, (and the Prioritized Product Backlog is prioritized by User Stories in the order of value created by them), the Product Owner and the customer can clearly visualize and articulate the value being created after every Sprint; and Scrum Teams in turn have the satisfaction of seeing their hard work being accepted by the customer and other business stakeholders.

2.3.1 Benefits of Self-organization

Self-organization as an essential principle in Scrum leads to the following:

- Team buy-in and shared ownership
- Motivation, which leads to an enhanced performance level of the team
- Innovative and creative environment conducive to growth
- Selection of the simplest and best approach to satisfy given requirements

The chief goals of self-organizing teams are as follows:

- Understand the Project Vision and why the project delivers value to the organization
- Estimate User Stories during the *Estimate User Stories* process and assign tasks to themselves during the *Update Sprint Backlog* process
- Identify tasks independently during the *Identify Tasks* process
- Apply and leverage their expertise from being a cross-functional team to work on the tasks during the *Create Deliverables* process
- Deliver tangible results which are accepted by the customer and other business stakeholders during the *Demonstrate and Validate Sprint* process
- Resolve individual problems together by addressing them during Daily Standup Meetings
- Clarify any discrepancies or doubts and be open to learning new things
- Upgrade knowledge and skills on a continuous basis through regular interactions within the team
- Maintain stability of team members throughout the duration of the project by not changing members, unless unavoidable

Figure 2-9 illustrates the goals of a self-organizing team, including autonomy, accountability, and collective responsibility for delivering value.

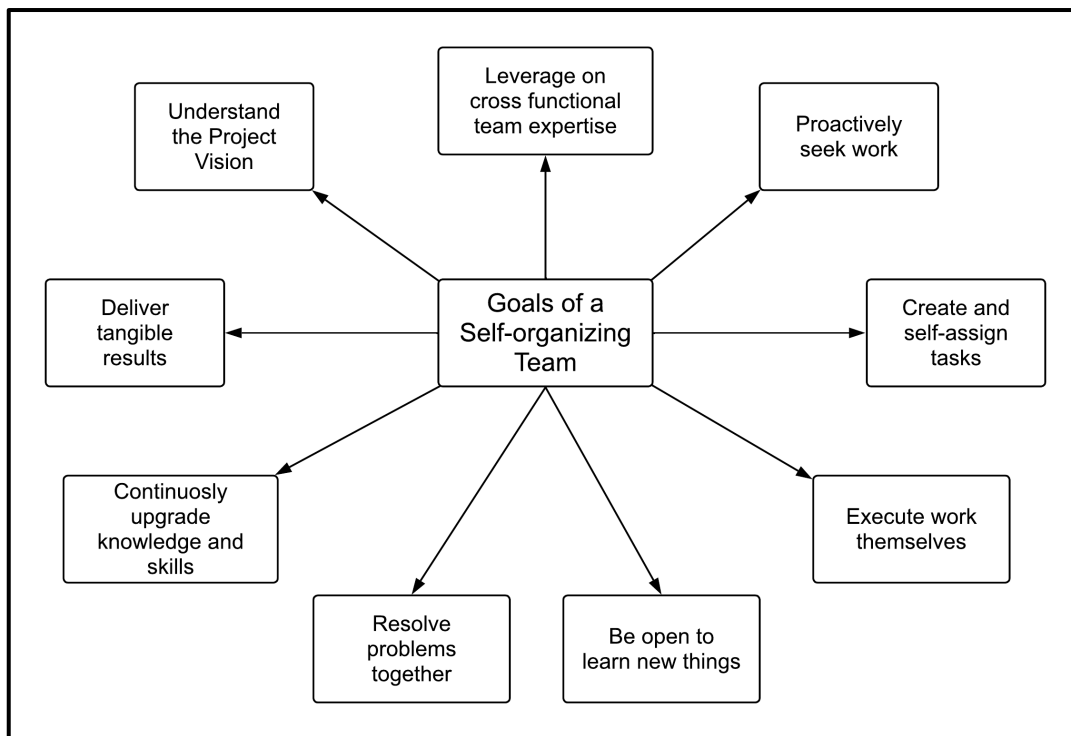


Figure 2-9: Goals of a Self-organizing Team

Figure 2-10 is an interface from Vabro, showing a Scrumboard that facilitates self-organization by allowing task assignment, tracking, and status updates in real time.

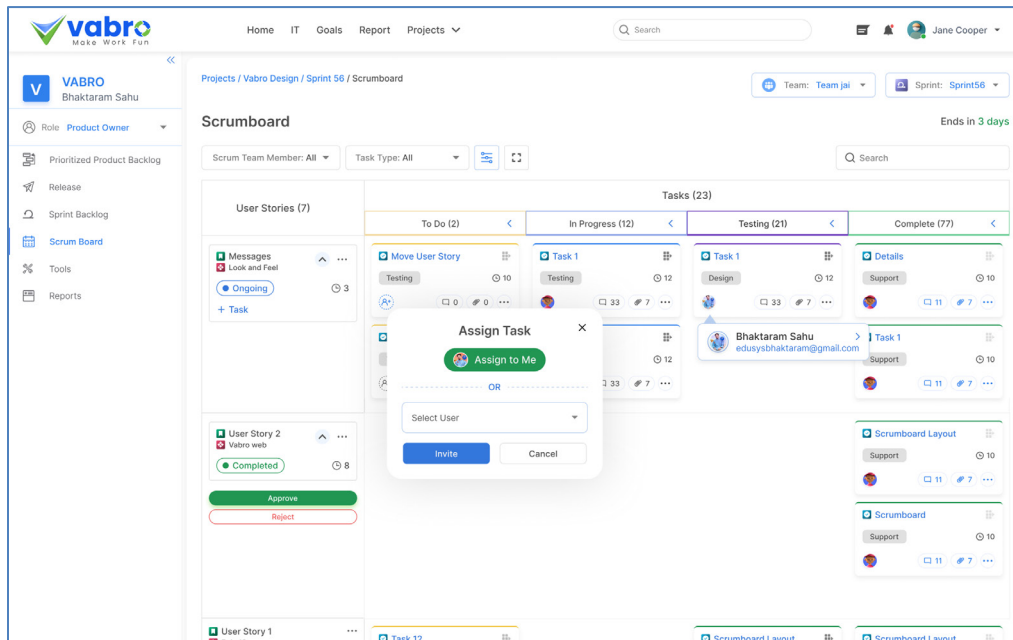


Figure 2-10: Facilitating self-organization via the Scrumboard in an AI-powered Scrum project tool (Source: Vabro)

Figure 2-11 is an interface from Jira, depicting a Scrumboard that enables self-organization through visual task management, promoting autonomy and collaboration.

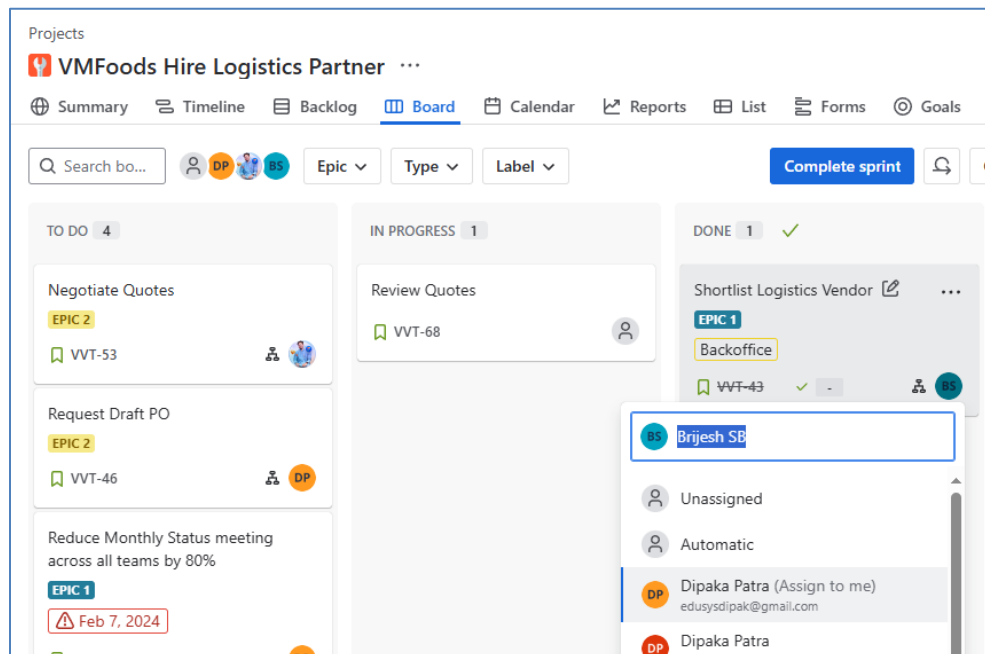


Figure 2-11: Scrumboard-enabled self-organization in AI-powered Scrum tool (Source: Jira)

2.4 Collaboration

Collaboration in Scrum refers to the Scrum Core Team working together and interfacing with the business stakeholders to create and validate the deliverables of the project to meet the goals outlined in the Project Vision. It is important to note the difference between cooperation and collaboration here. Cooperation occurs when the work product consists of the sum of the work efforts of various people on a team. Collaboration occurs when a team works together to play off each other's inputs to produce something greater. To achieve full collaboration, it is important to establish trust between all team members and between the team and the business stakeholders.

The core dimensions of collaborative work are as follows:

- **Awareness**—Individuals working together need to be aware of each other's work.
- **Articulation**—Collaborating individuals must partition work into units, divide the units among team members, and then after the work is done, reintegrate it.
- **Appropriation**—Adapting technology to one's own situation; the technology may be used in a manner completely different than expected by the designers.

2.4.1 Benefits of Collaboration in Scrum Projects

The Agile Manifesto (Fowler & Highsmith, 2001) stresses “customer collaboration over contract negotiation.” Thus, the Scrum framework adopts an approach in which the Scrum Core Team members (Product Owner, Scrum Master, and Scrum Team), collaborate with each other and the business stakeholders to create the deliverables that provide greatest possible value to the customer. This collaboration occurs throughout the project.

Collaboration ensures that the following project benefits are realized:

- **Change Requests are minimized.**
The need for changes due to poorly clarified requirements is minimized. For example, during the *Create Project Vision*, *Develop Epic(s)*, and *Create Prioritized Product Backlog* processes, the Product Owner collaborates with business stakeholders to create the Project Vision, Epic(s), and Prioritized Product Backlog, respectively. This will ensure that there is clarity among Scrum Core Team members regarding the work that is required to complete the project. The Scrum Team collaborates continuously with the Product Owner and business stakeholders through a transparent Prioritized Product Backlog to create the project deliverables. The processes *Conduct Daily Standup*, *Refine Prioritized Product Backlog*, and *Retrospect Sprint* provide scope to the Scrum Core Team members to discuss what has been done and collaborate on what needs to be done. Thus, the number of Change Requests from the customer and rework is minimized.
- **Risks are efficiently identified and mitigated.**
Risks are identified and dealt with efficiently. For example, risks to the project are identified and assessed in the *Develop Epic(s)*, *Create Deliverables*, and *Conduct Daily Standup* processes by the Scrum Core Team members.

The Scrum meeting tools such as the Daily Standup Meeting, Sprint Planning Meeting, Prioritized Product Backlog Review Meeting, and so on provide opportunities to the team to not only identify and assess risks, but also to efficiently implement risk responses (such as risk mitigation) to high-priority risks.

- **Efficiency is increased.**

True potential of the team is realized. For example, the *Conduct Daily Standup* process provides an opportunity for the Scrum Team to collaborate and understand the strengths and weaknesses of its members. If a team member has missed a task deadline, the Scrum Team members align themselves collaboratively to complete the task and meet the targets agreed to for completing the Sprint.

- **Continuous improvement is incorporated.**

Continuous improvement is ensured through lessons learned. For example, the Scrum Team uses the *Retrospect Sprint* process to identify what went well and what did not go well in the previous Sprint. This provides an opportunity for the Scrum Master to work with the team to rework and improve the team for the next scheduled Sprint. This will also ensure that collaboration is even more effective in the next Sprint.

Figure 2-12 illustrates the benefits of collaboration in Scrum projects, including enhanced communication, shared goals, and team-driven problem solving.

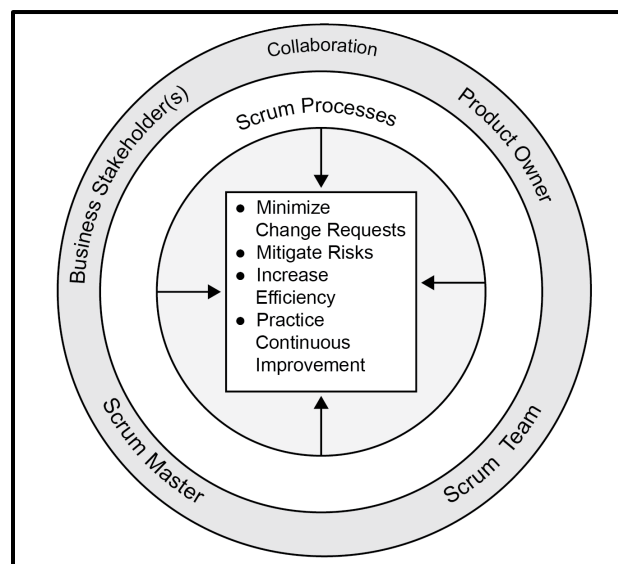


Figure 2-12: Benefits of Collaboration in Scrum Projects

Figure 2-13 is an interface from Vabro, showing a Message Center in an AI-powered Scrum tool. It supports real-time communication and collaboration among team members.

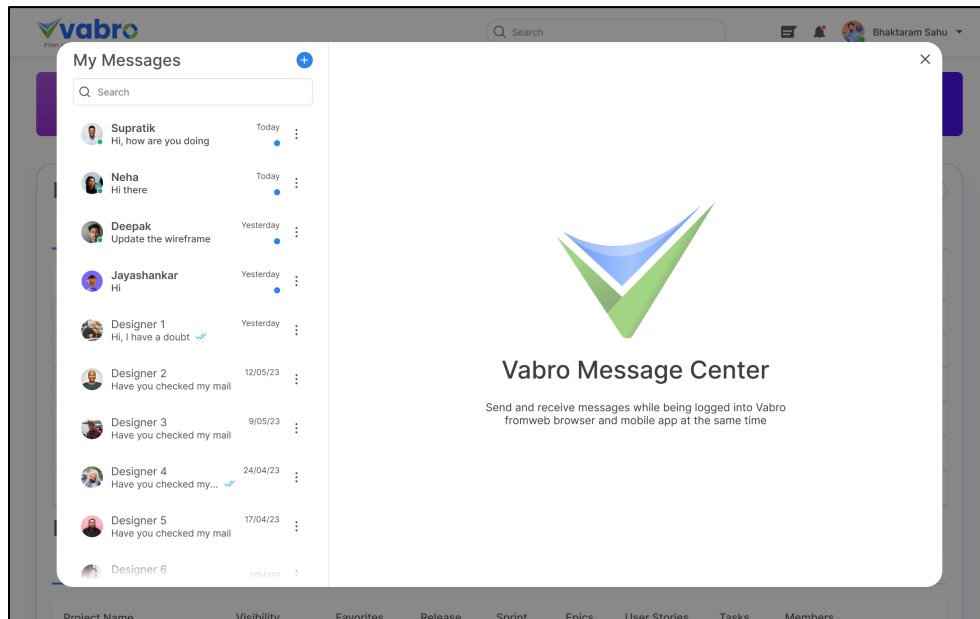


Figure 2-13: Message Center in an AI-powered Scrum Project Tool (Source: Vabro)

Figure 2-14 is an interface from Vabro, showing a task detail view designed for enhanced collaboration, featuring comments, assignments, and task history.

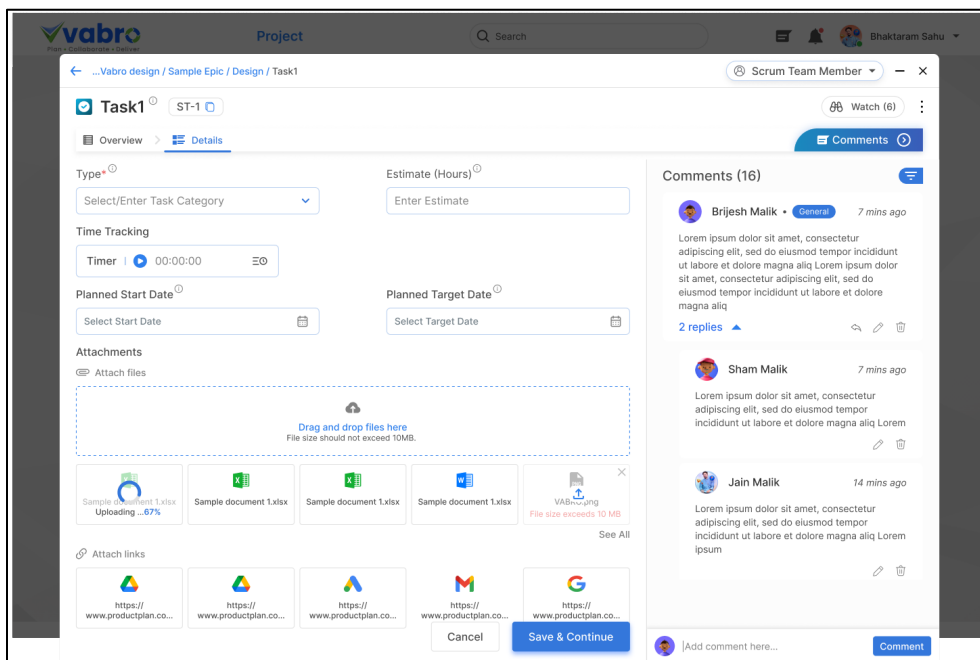


Figure 2-14: Task detail view for enhanced collaboration (Source: Vabro)

2.4.2 Importance of Colocation in Collaboration

For many of the Scrum practices, high-bandwidth communication is required. To enable this, it is preferred that team members are colocated. Preferred modes of communication include face-to-face interactions, Decision Rooms or War Rooms, Scrumboards, wall displays, shared tables, and so on. Colocation allows both formal and informal interaction between team members. This provides the advantage of having team members always at hand for coordination, problem-solving, and learning. Some of the benefits of colocation are the following:

- Questions get answered quickly.
- Problems are fixed on the spot.
- Less friction occurs between interactions.
- Trust is gained and awarded much more quickly.

2.4.3 Collaboration in Distributed Teams

Although colocated teams are preferred, at times the Scrum Team may be distributed. Scrum Team members may be working from multiple physical locations (such as different cities or even different countries) and/or may also be working from home. Even when teams are colocated, team members should have the flexibility to work remotely in case of any extenuating circumstances that could impact the team's ability to work in a colocated environment. In such situations, it may be required to ensure that Scrum Team members are set up to work effectively with the team distributed.

2.4.4 AI-powered Scrum Project Tool

Using a collaborative Artificial Intelligence(AI)-powered Scrum Project Tool is highly recommended to ensure that team members can work productively. Companies have reported up to a 70% increase in productivity and a 60% increase in return on investment by using a suitable AI-powered Scrum project tool.

The tool should ideally provide the capability to:

- Efficiently define all Scrum roles and provide messaging/collaboration functionality for all team members to interact with each other.
- Create and work through important Scrum artifacts, such as the Prioritized Product Backlog, Sprint Backlog, Scrumboard, etc.
- Provide the workflow to work through different Scrum processes involved in the Initiate, Planning, Implementing, Retrospect, and Release phases.
- Scale to organization or enterprise levels (if Scrum is being implemented within a large organization/enterprise).

- Schedule Scrum-related meetings, such as Release Planning Meetings, Daily Standup Meetings, Sprint Planning Meetings, Sprint Review Meetings, Retrospective Sprint Meetings, Product Backlog Review Meetings, and so on. However, the actual meetings may be conducted in a separate video conferencing tool.
- Allow Scrum Team members to easily communicate with each other online, either one-to-one or through distributed groups and/or discussion forums (since effective colocated Scrum Teams communicate with each other regularly); (Note however, that unlike with colocated teams, team members in distributed teams need to understand that other team members may not be available at the same time for instant communication.)
- Capture lessons learned (from retrospectives, etc.) with appropriate reports generated on the fly.
- Incorporate automation so that any templates or guidance from the Scrum Guidance Body are available to all Scrum Teams throughout the organization (for example, the Definition of Ready or the Definition of Done).
- Allow the SGB to assess Scrum-related behavior (such as maximum number of team members, Sprint durations, and so on).
- Clone from similar projects, Epics, and User Stories—this will allow Scrum Team members to spend less time creating unnecessary or duplicate documentation and learn from experiences from similar completed work. (This is especially beneficial when Scrum Teams are using similar implementation processes to create identical categories of products, for example, an advertising firm creating print advertisements for different clients; a construction firm creating drawings for similar road construction activities, and so on).

2.4.4.1 Using an AI-powered Scrum Project Tool for Distributed Teams:

Benefits of using an AI-powered Scrum Project Tool for distributed teams include the following:

- Facilitate Scrum Team members to work at any time, from any place
- Automation of reports, chats, calendar, workflows, and so on
- Enforcement of standard guidelines across the organization by automating SGB recommendations
- Increased efficiency due to a decrease in time spent creating repetitive or unnecessary documentation when cloning from similar projects (e.g., cloning Epics and User Stories)
- Working with a more diverse team (at times working from different countries) often incorporates local perspectives and experiences
- Less logistical challenges as compared to challenges that may arise from ensuring that all team members are working from one location. This can also save time and costs for expenses related to travel, expensive work locations, and so on

It is important for distributed teams to pay special attention to the principles of Scrum to ensure that they are followed. Emphasis should be on enabling a transparent and collaborative work environment of trust.

2.4.4.2 How AI Enhances Scrum Projects — Detailed by Component

1. Product Backlog Management

2

Challenges: Breaking down projects, epics, or user stories; prioritizing items; maintaining clarity; and aligning with business goals.

AI Solutions:

- **AI-Powered Breakdown:** Tools such as Vabro Genie AI, Jira plugins, etc., can break down projects into Epics, User Stories, and Tasks — and even provide sample estimates, priorities, categories, and other features associated with Product Backlog artifacts. This can be based on experience from similar projects, either within the company or from other organizations. Project teams can save up to 70% of their effort in defining projects, Epics, User Stories, and Tasks by using a suitable AI-enabled Scrum tool.
- **AI-Powered Prioritization:** Tools like Aha! or Jira plugins can analyze business value, cost, and customer feedback to auto-prioritize backlog items.
- **Intelligent Classification:** AI-powered Scrum Tools can tag, and group backlog items based on type (bug, feature, enhancement).
- **NLP for User Story Generation:** AI can convert emails, support tickets, or user feedback into structured backlog items.
- **Duplicate Detection:** AI can automatically detect and merge duplicate user stories.
- **Backlog Refining Assistant:** AI chatbots or assistants (like Vabro Genie AI, ChatGPT etc.) can suggest improvements, clarify acceptance criteria, or identify missing details.

2. Sprint Planning

Challenges: Accurate estimation and scope definition.

AI Solutions:

- **Effort Estimation:** AI models trained on historical sprint data can predict the time/effort needed for new User Stories.
- **Team Capacity Analysis:** AI powered reports can help evaluate historical velocity, availability, and skills to recommend realistic sprint goals.
- **Sprint Scope Optimization:** AI can help suggest the best combination of stories for the sprint based on effort and business value.

3. Daily Scrum (Standup Meetings)

Challenges: Coordination in remote/distributed teams, unproductive standups.

AI Solutions:

- **Automated Status Updates:** Bots can summarize progress and issues from tools like Vabro, Jira, Git, Slack, etc.
- **Speech-to-Text + Sentiment Analysis:** AI tools such as Microsoft Copilot, IBM Watsonx, Claude etc. can transcribe daily standups and assess team morale.
- **Proactive Issue Alerts:** AI can detect blockers or risks from discussions and automatically notify stakeholders.

4. Sprint Review

Challenges: Time-consuming preparation, misalignment on delivered value.

AI Solutions:

- **Demo Preparation Assistance:** AI can generate appropriate reports, demo scripts or summaries based on work completed e.g., Vabro Genie AI reports.
- **Stakeholder Feedback Analysis:** NLP models can analyze and categorize feedback from stakeholders to update the product backlog.
- **Progress Visualization:** AI can auto-generate visual reports and dashboards showing sprint achievements.

5. Sprint Retrospective

Challenges: Vague feedback, lack of actionable insights.

AI Solutions:

- **Sentiment & Topic Analysis:** Tools like Retrium with AI can analyze feedback tone and identify dominant themes.
- **Pattern Recognition:** AI can detect recurring issues (e.g., scope creep, delays) across multiple sprints by use of powerful AI-powered reports.
- **Action Item Suggestion:** AI can propose concrete improvements based on previous sprint performance and common best practices.

AI Tools Commonly Used in Scrum Projects

AI Tool	Use Case
Vabro.ai	Comprehensive AI-powered Scrum Tool for the entire Scrum project
ChatGPT	User story refinement, retrospective insights, meeting summaries
Jira AI plugins	Smart backlog grooming, risk detection, and reporting
ClickUp AI	Task summarization, goal tracking, and content generation
Retrium + AI	Enhanced retrospectives with pattern/sentiment analysis
Codacy/DeepCode	AI-assisted code reviews and quality tracking
Forecast.ai	Resource management and effort estimation

Table 2-1: AI Tools Commonly Used in Scrum Projects

Benefits of Using AI in Scrum

- Up to 70% increase in productivity.
- Easier breakdowns and replication of projects, Epics, User Stories or Tasks.
- Efficient management of all Scrum Roles.
- Comprehensive reporting and data analytics.
- Higher accuracy in estimation and risk prediction.
- Faster decision-making with real-time data analysis.
- Increased team productivity via automation and reduced manual effort.
- Improved stakeholder satisfaction through smarter prioritization and faster delivery.
- Stronger retrospectives by detecting patterns that humans may miss.

Challenges and Considerations

- Data Privacy: AI tools must comply with data governance standards (GDPR, etc.).
- Bias in AI: Training data can bias AI recommendations — human oversight is crucial.
- Team Adoption: AI should assist, not replace, human decision-making in Scrum.

2.5 Value-based Prioritization

The Scrum framework is driven by the goal of delivering maximum business value in a minimum time span. One of the most effective tools for delivering the greatest value in the shortest amount of time is prioritization. Prioritization can be defined as determination of the order and separation of what must be done now, from what needs to be done later. The concept of prioritization is not new to project management. The traditional Waterfall model of project management proposes using multiple task prioritization tools. From the Project Manager's point of view, prioritization is integral because certain tasks must be accomplished first to expedite the development process and achieve the project goals. Some of the traditional techniques of task prioritization include setting deadlines for delegated tasks and using prioritization matrices.

Scrum, however, uses Value-based Prioritization as one of the core principles that drives the structure and functionality of the entire Scrum framework—it helps projects benefit through adaptability and iterative development of the product or service. More significantly, Scrum aims at delivering a valuable product or service to the customer on an early and continuous basis. The Product Owner does prioritization when he or she prioritizes User Stories in the Prioritized Product Backlog. The Prioritized Product Backlog contains a list of all the requirements needed to bring the project to fruition.

Once the Product Owner has received business requirements from the customer, the business requirements are written in the form of Epics and User Stories (a specific format for capturing requirements). The Product Owner works with the customer and other business stakeholders to determine which business requirements provide maximum business value. Sometimes, a customer may insist that all User Stories are high priority. Even a list of high-priority User Stories needs to have relative priorities assigned. The Product Owner must understand what the customer wants and values in order to arrange the User Stories into a list from highest to lowest priority. This list is called the Prioritized Product Backlog and should contain all the requirements for the project. Prioritizing a backlog involves determining the criticality of each User Story. High business value requirements are identified and moved to the top of the Prioritized Product Backlog. The processes in which the principle of Value-based Prioritization is put into practice are *Create Prioritized Product Backlog* and *Refine Prioritized Product Backlog*.

Simultaneously, the Product Owner must work with the Scrum Team to understand the project risks and uncertainty as they may have negative consequences associated with them. This should be considered while prioritizing User Stories on a value-based approach (refer to the Risk chapter for more information). The Scrum Team also alerts the Product Owner of any dependencies that arise out of implementation. These dependencies must be considered during prioritization. Prioritization may be based on a subjective estimate of the projected business value or profitability, or it can be based on results and analysis of the market using tools including, but not limited to, customer interviews, surveys, and financial models and analytical techniques. The Product Owner has to translate the inputs and needs of the project business stakeholders to create the Prioritized Product Backlog. Hence, while prioritizing the User Stories in the Prioritized Product Backlog, the following three factors are considered (see Figure 2-15):

1. Value
2. Risk or uncertainty
3. Dependencies

Thus, prioritization results in deliverables that satisfy the requirements of the customer with the objective of delivering the maximum business value in the least amount of time.

Figure 2-15 illustrates the concept of value-based prioritization, emphasizing the alignment of backlog items with business value to maximize ROI.

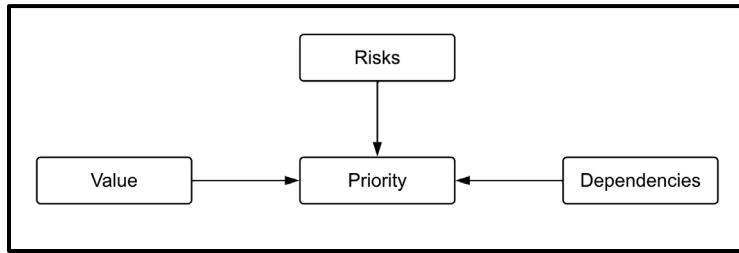


Figure 2-15: Value-based Prioritization

Figure 2-16 is an interface from Vabro, showing value-based prioritization where tasks are ranked by importance and business impact in an AI-powered Scrum tool.

User Story	Epic	Release	User Story Priority	Estimate (Story Points)	Status	Scrum Team	DoR
Sprint 57 Front End Bugs	Bugs	-	★★★★★	25	Scheduled	Dev Team +4	Yes 5 Star
ATP Portal Implementation	Payments	-	★★★★★	-	Not Scheduled	-	Yes
Identify & Fix Bugs Implement Improvements	Bugs	-	★★★★★	20	Not Scheduled	Dev Team +1	Yes
Club Forms and Tasks in Kanban Board	Product Roadmap Features	-	★★★★★	-	Not Scheduled	-	No
Exporting Vabro Data in Excel/CSV	Product Roadmap Features	-	★★★★★	-	Not Scheduled	-	No
API to Push Posts/Blogs/Articles from Kanban Board	Product Roadmap Features	-	★★★★★	-	Not Scheduled	-	No
Creating Recurring Tasks/TG in Kanban Board	Product Roadmap Features	-	★★★★★	-	Not Scheduled	-	No
Translation of Vabro Content (ChatGPT API)	Product Roadmap Features	-	★★★★★	-	Not Scheduled	-	No
Update New Fields/Keywords to Resource Files	Product Roadmap Features	-	★★★★★	304	Ongoing	Dev Team +5	Yes
Using AI to handle First Level Customer Support	Product Roadmap Features	-	★★★★★	80	Ongoing	Dev Team +1	Yes

Figure 2-16: Value-based prioritization in AI-powered Scrum Project Tool (Source: Vabro)

Figure 2-17 is an interface from Jira, showing value-based prioritization with numeric values assigned to tasks for prioritization based on business needs.

The screenshot shows the Jira task details for 'VVT-53 Negotiate Quotes'. The task is in the 'TO DO' state with a 'Medium' priority. It is blocked by 'VVT-46 Request Draft PO'. The interface includes sections for 'Linked work items', 'Confluence content', and 'Activity'.

Figure 2-17: Value-based prioritization in AI-powered Scrum Project Tool (Source: Jira)

2.6 Time-boxing

Scrum treats time as one of the most important constraints in managing a project. To address the constraint of time, Scrum introduces a concept called 'Time-boxing' which proposes fixing a certain amount of time for each process and activity in a Scrum project. This ensures that Scrum Team members do not take up too much or too little work for a particular period of time and do not expend their time and energy on work for which they have little clarity. Some of the advantages of Time-boxing are as follows:

- Efficient development process
- Less overheads
- High velocity for teams
- More focused teams
- Well-prepared team members

Time-boxing can be utilized in many Scrum processes, for example, in the *Conduct Daily Standup* process; the duration of the Daily Standup Meeting is Time-boxed. At times, Time-boxing may be used to avoid excessive improvement of an item (i.e., gold-plating). Time-boxing is a critical practice in Scrum and should be applied with care. Arbitrary Time-boxing can lead to de-motivation of the team and may have the consequence of creating an apprehensive environment, so it should be used appropriately.

2.6.1 Scrum Time-boxes

- **Sprint**—A Sprint is a Time-boxed iteration of one to four weeks in duration during which the Scrum Master guides, facilitates, and shields the Scrum Team from both internal and external impediments during the *Create Deliverables* process. This aids in avoiding vision creep that could affect the Sprint goal. During this time, the team works to convert the requirements in the Prioritized Product Backlog into shippable product functionalities. To achieve maximum benefits from a Scrum project and to provide maximum flexibility for change, the length of a Sprint should be as short as possible. At the same time, Sprint must be long enough for the team to be able to create a working or shippable product deliverable which can be reviewed and approved by the Product Owner.
- **Sprint Planning Meeting**—This meeting is conducted at the start of each Sprint as part of the *Commit User Stories*, *Identify Tasks*, *Estimate Tasks*, and *Update Sprint Backlog* processes. It is Time-boxed to two hours for each week of Sprint duration. For example, for a one-month/four-week Sprint, the Time-box for a Sprint Planning Meeting should be eight hours. The Sprint Planning Meeting satisfies the following objectives:
 1. **Objective Definition**—During the first part of the meeting, the Product Owner explains the highest priority User Stories or requirements in the Prioritized Product Backlog to the Scrum Team. The Scrum Team in collaboration with the Product Owner then commits to the User Stories, which define the Sprint goal.
 2. **Task Identification and Estimation**—The Scrum Team then decides how to complete the selected Prioritized Product Backlog items to fulfill the Sprint goal. The committed User Stories and related effort-estimated tasks (if available) are included in the Sprint Backlog to be tracked.

- **Daily Standup Meeting**—The Daily Standup Meeting is a short daily meeting, Time-boxed to 15 minutes. The team members get together to report the progress of the project by answering the following three questions:

1. What have I done since the last meeting?
2. What do I plan to do before the next meeting?
3. What impediments or obstacles (if any) am I currently facing?

This meeting is conducted by the team as part of the *Conduct Daily Standup* process.

- **Sprint Review Meeting**—The Sprint Review Meeting is Time-boxed to one hour for each week of the Sprint duration. For example, for a four-week Sprint, the Time-box for the Sprint Review Meeting should be four hours. During the Sprint Review Meeting that is conducted in the *Demonstrate and Validate Sprint* process, the Scrum Team presents the deliverables of the current Sprint to the Product Owner. The Product Owner reviews the product (or product increment) against the agreed Acceptance Criteria and either accepts or rejects the completed User Stories.

- **Retrospect Sprint Meeting**—The Retrospect Sprint Meeting is Time-boxed to one hour for each week of the Sprint duration. For example, for a four-week Sprint, the Time-box for the Retrospect Sprint Meeting should be four hours. This meeting is conducted as part of the *Retrospect Sprint* process. During this meeting, the Scrum Team gets together to review and reflect on the current Sprint in terms of the processes followed, tools employed, collaboration and communication mechanisms, and other aspects relevant to the project. The team discusses what went well during the previous Sprint and what did not go well, the goal being to learn and make improvements in the Sprints to follow. Some improvement opportunities or best practices from this meeting could also be updated as part of the Scrum Guidance Body documents.

Figure 2-18 depicts time-box durations for Scrum meetings, such as Daily Scrums, Sprint Planning, Reviews, and Retrospectives, helping teams manage time effectively.

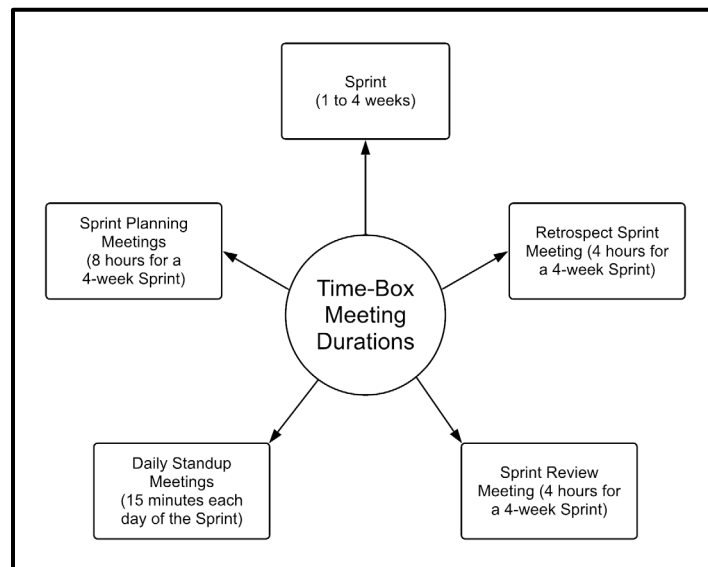


Figure 2-18: Time-Box Durations for Scrum Meetings

Figure 2-19 is an interface from Vabro, showing Sprint Planning using Vabro Genie, allowing teams to allocate time-boxes, define tasks, and set priorities.

Figure 2-19: Sprint Planning with Vabro Genie (Source: Vabro)

Figure 2-20 is an interface from Jira, illustrating time-boxed Sprint Planning with fixed-duration planning sessions and clear task breakdowns.

Figure 2-20: Time-boxed Sprint Planning (Source: Jira)

2.7 Iterative Development

The Scrum framework is driven by the goal of delivering maximum business value in a minimum time span. To achieve this, Scrum application involves the iterative development of deliverables.

In most complex projects, the customer may not be able to define very concrete requirements or is not confident of what the end product may look like. The iterative model is more flexible in ensuring that any change requested by the customer can be included as part of the project. User Stories may have to be written constantly throughout the duration of the project. In the initial stages of writing, most User Stories are high-level functionalities. These User Stories are known as Epic(s). An Epic is usually too large for teams to complete in a single Sprint. Therefore, they are broken down into smaller User Stories.

Each complex aspect of the project is broken down through progressive elaboration during the *Refine Prioritized Product Backlog* process. The *Create User Stories* and the *Estimate User Stories* and *Commit User Stories* processes are used to add new requirements to the Prioritized Product Backlog. The Product Owner's task is to ensure increased ROI by focusing on value and its continuous delivery with each Sprint. The Product Owner should have a particularly good understanding of the project's business justification and the value the project is supposed to deliver as he/she drafts the Prioritized Product Backlog and thereby decides what deliverables and hence value is delivered in each Sprint. Then the *Identify Tasks*, *Estimate Tasks*, and *Update Sprint Backlog* processes produce the Sprint Backlog which the team uses to create the deliverables.

In each Sprint, the *Create Deliverables* process is used to develop the Sprint's outputs. The Scrum Master has to ensure that the Scrum processes are followed and facilitates the team to work in the most productive manner possible. The Scrum Team self-organizes and aims to create the Sprint Deliverables from the User Stories and tasks in the Sprint Backlog. In large projects, various cross-functional teams work in parallel across Sprints, delivering potentially shippable solutions at the end of each Sprint. After the Sprint is complete, the Product Owner accepts or rejects the deliverables based on the Acceptance Criteria in the *Demonstrate and Validate Sprint* process.

The benefit of iterative development is that it allows course correction as all the people involved get a better understanding of what needs to be delivered as part of the project and incorporate this learning in an iterative manner. Thus, the time and effort required to reach the final end point is reduced and the team produces deliverables that are better suited to the final business environment.

Figure 2-21 compares Scrum with traditional Waterfall methodology, highlighting differences in flexibility, planning, feedback loops, and delivery style.

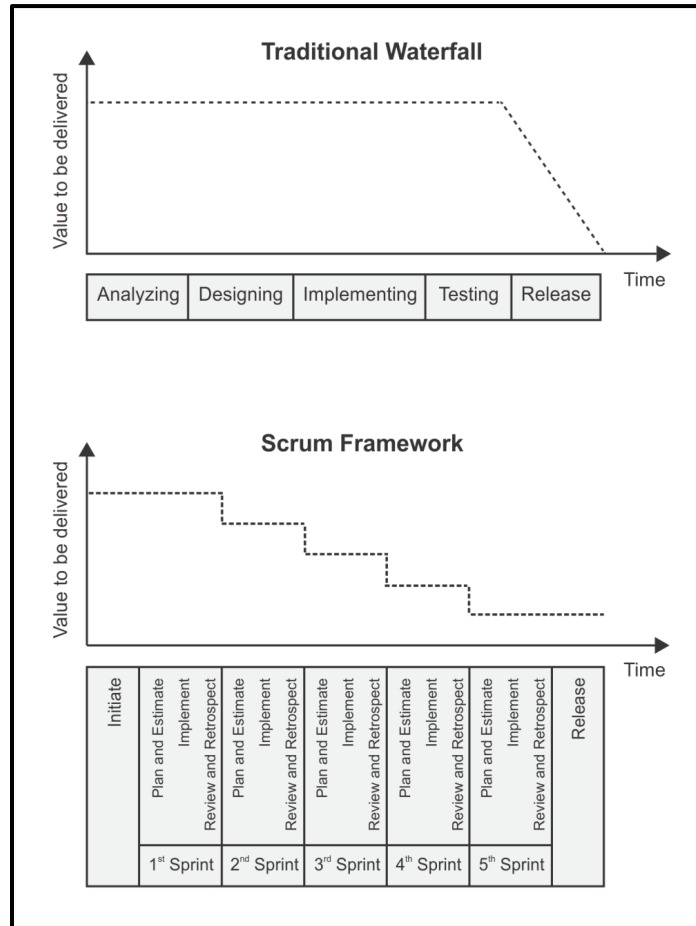


Figure 2-21: Scrum vs. Traditional Waterfall

2.8 Scrum vs. Traditional Project Management

The emphasis in traditional Project Management is to conduct detailed upfront planning for the project with emphasis on fixing the scope, cost, and schedule and managing those parameters. Traditional project management may at times lead to a situation where the plan has succeeded yet the customer is not satisfied.

The Scrum framework is founded on the belief that the knowledge workers of today can offer much more than just their technical expertise, and that trying to fully map out and plan for an ever-changing environment is not efficient. Therefore, Scrum encourages data-based, iterative decision making. In Scrum, the primary focus is on delivering products that satisfy customer requirements in small iterative shippable increments.

To deliver the greatest amount of value in the shortest amount of time, Scrum promotes prioritization and Time-boxing over fixing the scope, cost, and schedule of a project. An important feature of Scrum is self-organization, which allows the individuals who are actually doing the work to estimate and take ownership of tasks.

This Page is Intentionally Left Blank.

3. ORGANIZATION

3.1 Introduction

In this section, the various facets of a Scrum project organization are discussed, as well as core and non-core roles and how to form high performance Scrum Teams. *Organization*, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams. This chapter is divided into the following sections:

3.1.1 Roles Guide—This section identifies which section or subsection is important for a Product Owner, Scrum Master, and Scrum Team.

3.2 Scrum Project Roles—This section covers all the key core and non-core roles associated with a Scrum project.

3.3 Product Owner—This section highlights the key responsibilities of the Product Owner in relation to a Scrum project, program, and portfolio.

3.4 Scrum Master—This section focuses on the key responsibilities of the Scrum Master in the context of a Scrum project, program, and portfolio.

3.5 Scrum Team—This section emphasizes the key responsibilities of the Scrum Team in the context of a Scrum project.

3.6 Scrum in Projects, Programs, and Portfolios—This section focuses on how the Scrum framework can be tailored and used in the different contexts of programs and portfolios. It also highlights the specific responsibilities of the Scrum Team members in relation to communication, integration, and working with the corporate and program management teams.

3.7 Responsibilities—This section describes the responsibilities relevant to the Organization theme, for everyone working on a project, based on their roles.

3.8 Scrum vs. Traditional Project Management—This section explains the key differences and advantages of the Scrum model in relation to the traditional Waterfall model of project management.

3.9 Popular HR Theories and their Relevance to Scrum—This section contains some of the most popular HR theories useful for all the members in the Scrum Core Team.

3.1.1 Roles Guide

1. Product Owner—It is imperative for Product Owners to read the entire chapter.
2. Scrum Master—The Scrum Master should be familiar with this entire chapter with primary focus on sections 3.3, 3.5, 3.6, 3.8, and 3.9.4.
3. Scrum Team— The Scrum Team should focus on sections 3.3, 3.6, and 3.8.

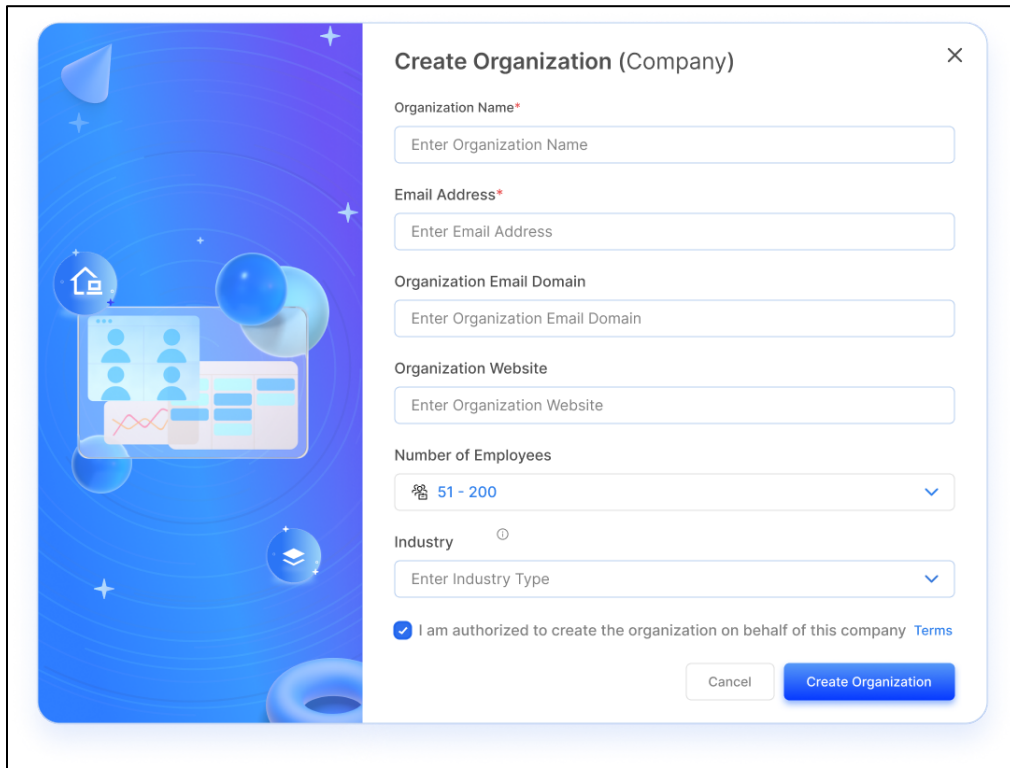
3.2 Scrum Project Roles

Understanding defined roles and responsibilities is particularly important for ensuring the successful implementation of Scrum projects.

Scrum roles fall into two broad categories:

1. **Core Roles**—Core roles are those roles which are mandatorily required for producing the product of the project, are committed to the project, and ultimately are responsible for the success of each Sprint of the project and of the project as a whole.
2. **Non-core Roles**—Non-core roles are those roles which are not mandatorily required for the Scrum project. They may include team members who are interested in the project but have no formal role on the project team. These individuals may interface with the team but may not be responsible for the success of the project. The non-core roles should also be considered in any Scrum project.

Figure 3-1 shows Vabro user interface form titled "Create Organization (Company)," requiring inputs like organization name, email, domain, website, number of employees, and industry, with a "Create Organization" button.



Create Organization (Company) ×

Organization Name*

Email Address*

Organization Email Domain

Organization Website

Number of Employees

Industry ⓘ

☒ I am authorized to create the organization on behalf of this company [Terms](#)

Figure 3-1: Organization Setup (Source: Vabro)

3.2.1 Core Roles

There are three core roles in Scrum that are responsible for meeting the project objectives. The core roles are the Product Owner, Scrum Master, and Scrum Team. Together they are referred to as the Scrum Core Team. It is important to note that, of these three roles, no role has authority over the others.

1. **Product Owner**—The Product Owner is the person responsible for maximizing business value for the project. He or she is responsible for articulating customer requirements and maintaining business justification for the project. The Product Owner represents the *Voice of the Customer*.
2. **Scrum Master**—The Scrum Master is a facilitator who ensures that the Scrum Team is provided with an environment conducive to completing the product's development successfully. The Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved in the project; clears impediments for the team; and ensures that Scrum processes are being followed.

Note that the Scrum Master role is vastly different from the role played by the Project Manager in a traditional Waterfall model of project management, in which the Project Manager works as a manager or leader for the project. The Scrum Master only works as a facilitator and he or she is at the same hierarchical level as anyone else in the Scrum Team—any person from the Scrum Team who learns how to facilitate Scrum projects can become the Scrum Master for a project or for a Sprint.

3. **Scrum Team**—The Scrum Team is a group or team of people who are responsible for understanding the business requirements specified by the Product Owner, estimating User Stories, and final creation of the project deliverables.

Figure 3-2 shows an Airtable workspace for “Employee onboarding” under the “Team members” tab. It lists employees with details like name, photo, title, team, email, and onboarding checklist tasks.

	Name	Photo	Title	Team	Email	Favorite X-Men?	Onboarding checklist
1	Charles Xavier		CEO	Leadership	charles@x.com	Professor X	New hire orientation with CEO
2	Ororo Monroe		Board member	Leadership	ororo@x.com	Storm	
3	Max "Magnus" Eisenhardt		Engineer	Engineering	max@x.com	Magneto	
4	Kurt Wagner		Head of engineering	Engineering	kurt@x.com	Nightcrawler	
5	Kelly Sall		Marketing director	Marketing	kelly@x.com	Sprite/Ariel/Shadowcat	Schedule an initial meeting with y
6	James "Logan" Howlett		Marketing manager	Marketing	logan@x.com	Wolverine	Initial check-in with your manager Week 1 check-in with your manager
7	Rémy Etienne LeBeau		Content marketing manager	Marketing	remy@x.com	Gambit	

Figure 3-2: Team Composition (Source: Airtable)

Figure 3-3 provides an overview of the core Scrum team roles, including Product Owner, Scrum Master, and Scrum Team, and explains their responsibilities.

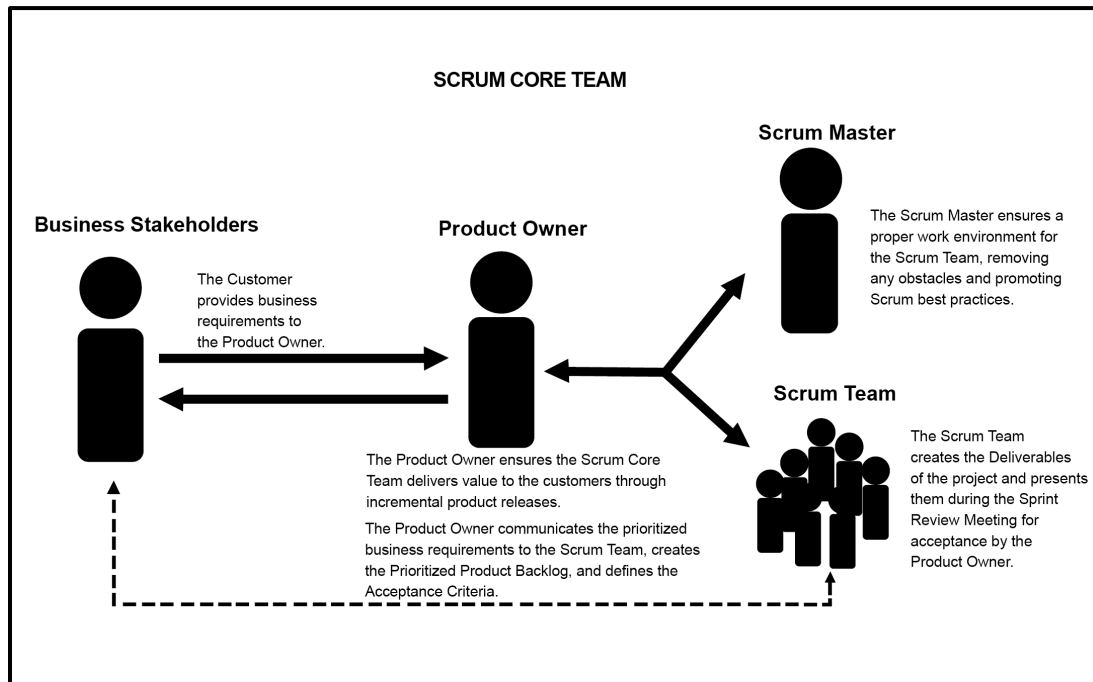


Figure 3-3: Scrum Roles—Overview

3.2.2 Non-core Roles

The non-core roles are those roles which are not mandatorily required for the Scrum project and may not be continuously or directly involved in the Scrum process. However, understanding non-core roles is important as these roles play a significant part in some Scrum projects.

Non-core roles include the following:

1. **Business Stakeholder(s)**—Business stakeholder(s) is a collective term that includes customers, users, and sponsor(s), who frequently interface with the Product Owner, Scrum Master, and Scrum Team to provide them with inputs and facilitate creation of the project's product, service, or other result. Business stakeholders(s) influence the project throughout the project's development. Business stakeholders may also have a role to play during the *Develop Epic(s)*, *Create Prioritized Product Backlog*, *Conduct Release Planning*, *Retrospect Sprint*, and other important processes in Scrum. Scrum requires complete support from the project's business stakeholders.

The responsibility for keeping business stakeholders engaged lies with the Product Owner. The following are actions recommended for maintaining stakeholder engagement and support:

- Ensure effective collaboration and business stakeholder involvement in the project
- Continually assess the business impact
- Maintain regular communication with business stakeholders
- Manage business stakeholders' expectations

At times, the same person or organization will have multiple business stakeholder roles; for example, the sponsor and the customer may be the same.

The following defines the roles of the relevant business stakeholders on a Scrum project:

- **Customer**—The customer is the individual, organization, or group of individuals acquiring the project's product, service, or other result. For any organization, depending on the project, there can be either internal customers (i.e., within the same organization) or external customers (i.e., outside of the organization).
- **Users**—Users are the individual, organization, or group of individuals that directly uses the project's product, service, or other result. Like customers, for any organization, there can be both internal and external users. Also, in some industries and on some projects, customers and users may be the same individuals.
- **Sponsor**—The sponsor is the individual or the organization that provides funding, support, and other resources for the project. The sponsor is also the business stakeholder to whom team members are accountable.

The sponsor should understand the financial bottom line related to a product or service and is typically more concerned with final outcomes rather than with individual tasks. It is important that the sponsor (or sponsors) who are funding the project have clarity on the following considerations:

- Benefits of applying Scrum practices on the project
 - Target deadlines and estimated costs of the Scrum project
 - Overall risks involved in the Scrum project and the steps to mitigate or avoid them
 - Expected release dates and final deliverables
2. **Supporting Services**—Supporting services are internal or external groups that support or are impacted by the Scrum project, for example, training, logistics, marketing, finance, infrastructure, and so on.
 3. **Vendors**—Vendors include external individuals or organizations that provide products and services that are not within the core competencies of the project organization.
 4. **Scrum Guidance Body**—The Scrum Guidance Body (SGB) is an optional role but highly recommended to formalize organizational practices related to Scrum. The Scrum Guidance Body consists of a group of documents and/or a group of experts who are typically involved with defining objectives related to quality, government regulations, security, and other key organizational parameters. These objectives guide the work carried out by the Product Owner, Scrum Master, and Scrum Team.
The Scrum Guidance Body also helps capture the best practices that should be used across all Scrum projects in the organization.

The Scrum Guidance Body does not make decisions related to the project. Instead, it acts as a consulting or guidance structure for all the hierarchy levels in the project organization—the portfolio, program, and project. Scrum Teams have the option of asking the Scrum Guidance Body for advice as required.

3.3 Product Owner

The Product Owner represents the interests of the stakeholder community. The Product Owner is responsible for ensuring clear communication of product or service functionality requirements to the Scrum Team, defining the Acceptance Criteria, and ensuring those criteria are met. In other words, the Product Owner is responsible for ensuring that the Scrum Team delivers value. The Product Owner must always maintain a dual view. He or she must understand and support the needs and interests of all business stakeholders, while also understanding the needs and workings of the Scrum Team. Because the Product Owner must understand the needs and priorities of the business stakeholders, including customers and users, this role is commonly referred to as the Voice of the Customer.

Table 3-1 summarizes the Product Owner's responsibilities in the various Scrum processes.

Process	Product Owner Responsibilities
8.1 Create Project Vision	- Defines the Project Vision - Helps create the Project Charter and project budget
8.2 Identify Scrum Master and Business Stakeholder(s)	- Helps finalize Scrum Master for the project - Identifies Business Stakeholder(s)
8.3 Form Scrum Team	- Helps determine Scrum Team members - Helps develop a Collaboration Plan - Helps develop the Team Building Plan with Scrum Master(s)
8.4 Develop Epic(s)	Creates Epic(s) and Personas
8.5 Create Prioritized Product Backlog	- Prioritizes items in the Prioritized Product Backlog - Defines Done Criteria and meets the Definition of Ready
8.6 Conduct Release Planning	- Creates Release Planning Schedule - Helps determine Length of Sprint
9.1 Create User Stories	- Accountable for creation of User Stories - Defines Acceptance Criteria for every User Story
9.2 Estimate User Stories	Clarifies User Stories
9.3 Commit User Stories	Works with Scrum Team to commit User Stories
9.4 Identify Tasks	Explains User Stories to the Scrum Team while creating the Task List
9.5 Estimate Tasks	Provides guidance and clarification to the Scrum Team in estimating effort for tasks
9.6 Update Sprint Backlog	Clarifies requirements for the Scrum Team while the team is creating the Sprint Backlog
10.1 Create Deliverables	Clarifies business requirements to the Scrum Team
10.3 Refine Prioritized Product Backlog	Refines Prioritized Product Backlog
11.1 Demonstrate and Validate Sprints	- Accepts/rejects deliverables - Provides necessary feedback to Scrum Master and Scrum Teams

	• Updates Release Plan and Prioritized Product Backlog
12.1 Ship Deliverables	• Helps deploy Product Releases and coordinate this with the customer
12.2 Retrospect Release	• May participate in Retrospect Release Meetings

Table 3-1: Responsibilities of the Product Owner in Scrum Processes

The other responsibilities of a Product Owner are:

- Determining the project's initial overall requirements and kicking off project activities; this may involve interaction with the Program Product Owner and the Portfolio Product Owner to ensure that the project aligns with the direction provided by senior management.
- Representing user(s) of the product or service with a thorough understanding of the user community
- Securing the initial and ongoing financial resources for the project.
- Focusing on value creation and overall Return on Investment (ROI).
- Assessing the viability and ensuring the delivery of the product or service.

The Product Owner may not always represent an external customer or business. For example, in an IT project, requirements, such as improving performance, scalability, testability, reliability, information security, and compliance may be owned by the technology groups inside the organization. Product Owners in such cases could also have roles such as technical architects, technical leads, etc.

3.3.1 Voice of the Customer (VOC)

As the representative of the customer and other business stakeholders, the Product Owner is said to be the Voice of the Customer. The Product Owner ensures that the explicit and implicit needs of the customer are translated into User Stories in the Prioritized Product Backlog and later on used to create project deliverables for the customer.

3.4 Scrum Master

The Scrum Master is the “supporting leader” of the Scrum Team who moderates and facilitates team interactions as team coach and motivator. The Scrum Master is responsible for ensuring that the team has a productive work environment by guarding the team from external influences, removing any obstacles, and enforcing Scrum principles, aspects, and processes.

Table 3-2 summarizes the Scrum Master’s responsibilities in the various Scrum processes.

Processes	Scrum Master Responsibilities
8.2 Identify Scrum Master and Business Stakeholder(s)	Helps identify business stakeholders(s) for the project
8.3 Form Scrum Team	- Facilitates selection of members to the Scrum Team(s) - Facilitates creation of the Collaboration Plan and the Team Building Plan - Ensures back-up resources are available
8.4 Develop Epic(s)	Facilitates creation of Epic(s) and Personas
8.5 Create Prioritized Product Backlog	Helps Product Owner in creation of the Prioritized Product Backlog and in definition of the Done Criteria and meeting the Definition of Ready
8.6 Conduct Release Planning	- Coordinates creation of Release Planning Schedule - Helps Product Owner and Scrum Team to determine Length of Sprint
9.1 Create User Stories	Facilitates creation of User Stories and their Acceptance Criteria
9.2 Estimate User Stories	Facilitates meetings of the Scrum Team to estimate User Stories
9.3 Commit User Stories	Facilitates meetings of the Scrum Team to commit User Stories
9.4 Identify Tasks	Facilitates the Scrum Team in creating the Sprint Task List
9.5 Estimate Tasks	Assists Scrum Team in estimating the effort required to complete the tasks agreed to for the Sprint
9.6 Update Sprint Backlog	Assists Scrum Team in developing the Sprint Backlog and the Sprint Burndown Chart
10.1 Create Deliverables	- Supports Scrum Team in creating the deliverables agreed to for the Sprint - Helps update the Scrumboard and the Impediment Log
10.2 Conduct Daily Standup	Ensures that the Scrumboard and the Impediment Log remain updated
10.3 Refine Prioritized Product Backlog	Facilitates Prioritized Product Backlog Review Meetings
11.1 Demonstrate and Validate Sprints	Facilitates presentation of completed deliverables by the Scrum Team for the Product Owner’s approval
11.2 Retrospect Sprint	Ensures that ideal project environment exists for the Scrum Team in the succeeding Sprints
12.2 Retrospect Release	Represents the Scrum Core Team to provide lessons from the current release

Table 3-2: Responsibilities of the Scrum Master in Scrum Processes

3.5 Scrum Team

The Scrum Team is sometimes referred to as the Development Team as they are responsible for developing the product, service, or any other result. It consists of a group of self-organized individuals who work on the User Stories in the Sprint Backlog to create the deliverables for the project. Table 3-3 summarizes the Scrum Team's responsibilities in the various Scrum processes.

Processes	Scrum Team Responsibilities
8.3 Form Scrum Team	Provides inputs for creation of the Collaboration Plan and the Team Building Plan
8.4 Develop Epic(s)	Ensures a clear understanding of Epics and personas
8.5 Prioritized Product Backlog	Understands the Epics and User Stories in the Prioritized Product Backlog
8.6 Conduct Release Planning	- Agrees with other Scrum Core Team members on the Length of Sprint - Seeks clarification on new products or changes in the existing products in the refined Prioritized Product Backlog
9.1 Create User Stories	Provides input to the Product Owner on creation of User Stories
9.2 Estimate User Stories	Estimates User Stories approved by the Product Owner
9.3 Commit User Stories	Commits User Stories to be done in a Sprint
9.4 Identify Tasks	Develops Task List based on agreed User Stories and dependencies
9.5 Estimate Tasks	Estimates effort for tasks identified and updates the Task List
9.6 Update Sprint Backlog	Defines the User Stories and tasks to be included in the Sprint Backlog and tracked on the Sprint Burndown Chart
10.1 Create Deliverables	- Creates deliverables - Identifies risks and implements risk mitigation actions - Identifies impediments to be tracked on the Impediments Log
10.2 Conduct Daily Standup	- Updates Scrumboard throughout each Sprint - Discusses issues faced by individual members and seeks solutions to motivate the team
10.3 Refine Prioritized Product Backlog	Participates in Prioritized Product Backlog Review Meetings
11.1 Demonstrate and Validate Sprints	Demonstrates completed deliverables to the Product Owner for approval
11.2 Retrospect Sprint	Identifies improvement opportunities from the current Sprint and agrees on any actionable improvements for the next Sprint
12.2 Retrospect Release	Participates in the Retrospect Release Meeting

Table 3-3: Responsibilities of the Scrum Team in Scrum Processes

3.5.1 Personnel Selection

Figure 3-4 lists desirable traits for core Scrum roles, emphasizing collaboration, decision-making, leadership, adaptability, and communication skills.

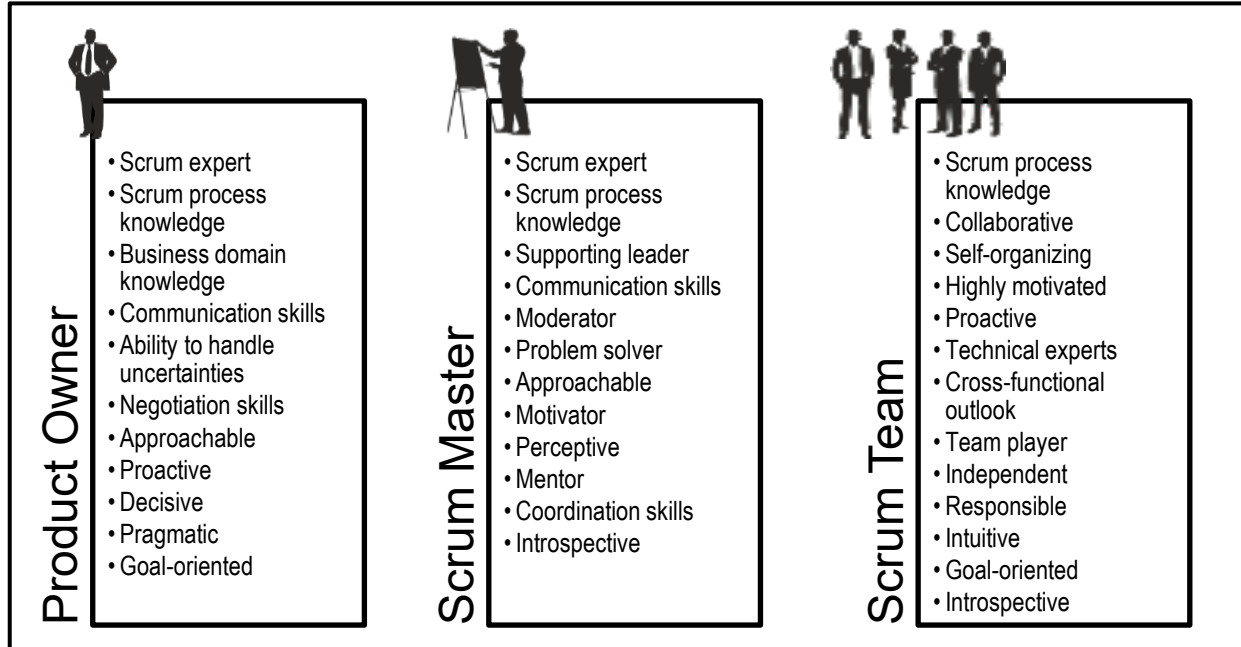


Figure 3-4: Desirable Traits for the Core Scrum Roles

3.5.2 Scrum Team Size

It is important for the Scrum Team to possess all the essential skills required to carry out the work of the project. It is also necessary to have an elevated level of collaboration to maximize productivity, so that minimal coordination is required to get things done.

The optimum size for a Scrum Team is six to ten members—large enough to ensure adequate skill sets—but small enough to collaborate easily. The goal is to have enough people on the team to get the work done but still stay small enough to have effective communication and collaboration within the team. There may be drawbacks to smaller team sizes. One potential drawback is that smaller teams are more significantly impacted by the loss of a team member than larger teams, even for a brief period of time. To address this problem, it may be possible for team members to have expert knowledge and skills outside their own specific role. However, this may be difficult and depends on the type of project, industry, and size of the organization. It is also recommended to have back-up people to replace any person who may have to leave the Scrum Team.

3.6 Scrum in Large Projects, Programs, and Portfolios

3.6.1 Making Scrum Work in a Large Project

The fundamental Scrum processes defined in chapters 8 through 12 are valid for Scrum projects with one Product Owner, one Scrum Master, and one to three Scrum Teams. These are usually considered small Scrum projects.

When dealing with large projects requiring the efforts of four or more Scrum Teams with multiple Product Owners and multiple Scrum Masters, the fundamental processes defined in chapters 8 through 12, remain valid, but some additional considerations and updates to inputs, tools, and outputs may be required. This may include additional coordination and synchronization efforts. The impacts to the fundamental Scrum processes when scaling Scrum to large projects are described in detail in chapter 13.

The definition of what constitutes a large project usually depends on the organization and/or the complexity of the projects being undertaken. A key criterion for whether a project is considered small versus large is whether the project requires multiple Scrum Masters and/or multiple Product Owners. If the project requires only one Scrum Master and one Product Owner, then these individuals can normally handle any additional communication and synchronization efforts required by the project.

Some reasons additional inputs, tools, and outputs would be needed for large projects are as follows:

Product Owners

- Need for collaboration between Product Owners when working with business stakeholders, refining the Prioritized Product Backlog, and working with multiple Scrum Teams

It is also important to note that as Scrum is scaled for large projects, additional supporting services may be needed such as architects, product managers, compliance, information security, governance bodies, and so on.

Scrum Masters

- Need for collaboration between Scrum Masters when addressing impediments and for synchronizing the work of the multiple Scrum Teams

Scrum Teams

- Increased interaction and dependencies among Scrum Teams as complexity increases for a large project
- Need to manage conflicts, resolve issues, and set priorities between the Scrum Teams
- Requirement for specialization as some Scrum Teams may require specialized resources for specific tasks (and these particular skill sets are not needed on all Scrum Teams)
- Necessity to define certain guidelines and standards that should be adhered to by all Scrum Teams (e.g., security standards within a company or legal and governmental guidelines for specific industries); these may be defined by the Scrum Guidance Body
- Requirement to set up an environment or working area for the large project, which would then be used by all Scrum Teams
- Need for coordinating the outputs from several Scrum Teams to facilitate the release of a large project

3.6.2 Additional Core Roles in a Large Project

When scaling Scrum to large projects, the following additional core roles may be needed:

3.6.2.1 Chief Product Owner

In the case of large projects with numerous Scrum Teams and multiple Product Owners, it is still necessary to have one single person who makes the day-to-day business decisions and has the role of Chief Product Owner. This role is responsible for coordinating the work of multiple Product Owners on a large Scrum project. With help from the Product Owners for each respective Scrum Team, the Chief Product Owner prepares and maintains the overall Prioritized Product Backlog and uses it to coordinate work top-down through the Product Owners of each Scrum Team. The Chief Product Owner is responsible for the final deliverables of the project, whereas the Product Owners of each team are responsible only for those deliverables being developed by their respective Scrum Teams.

In a large project, the Chief Product Owner will be tasked with prioritizing competing requests raised by the Product Owners of each Scrum Team based on their interactions with the business stakeholders. The complexity of this task increases with the increase in number of Scrum Teams and the number of Product Owners on the project. An important part of the complexity of this task is to ensure that the various components are properly integrated and at appropriate times. Therefore, it is imperative for the Chief Product Owner to develop a list of components and resources needed in common for all teams throughout the project. Although the Chief Product Owner makes the final business decisions, he or she collaborates with other Product Owners, the Chief Scrum Master, and the Scrum Masters to develop this list. The Chief Product Owner may also need to interface with a Program Product Owner to ensure alignment of the large project with the goals and objectives of the program.

Chief Product Owners should refer to the Roles Guide sections in the *SBOK® Guide* (see section 3.1.1) that define the role of the Product Owner. More detailed information about the role is presented in chapter 13, which describes Scaling Scrum for Large Projects.

3.6.2.2 Chief Scrum Master

Large projects require multiple Scrum Teams to work in parallel. Information gathered from one team may need to be appropriately communicated to other teams—the Chief Scrum Master is responsible for this activity.

The role of a Chief Scrum Master is necessary to ensure proper collaboration among the Scrum Teams. Coordination across various Scrum Teams working on a project is typically done through the Scrum of Scrums (SoS) Meeting (see section 13.3.5). There is no role hierarchy of Scrum Masters—they are all peers. The Chief Scrum Master works at a multi-team level, whereas the Scrum Masters each work at a single-team level.

Figure 3-5 is an interface from Vabro, showing a Kanban template with "Team" details. It features tabs for team selection, fields for assigning Kanban Manager and members, and a comment section with threaded replies, facilitating team collaboration and communication within a Kanban initiative.

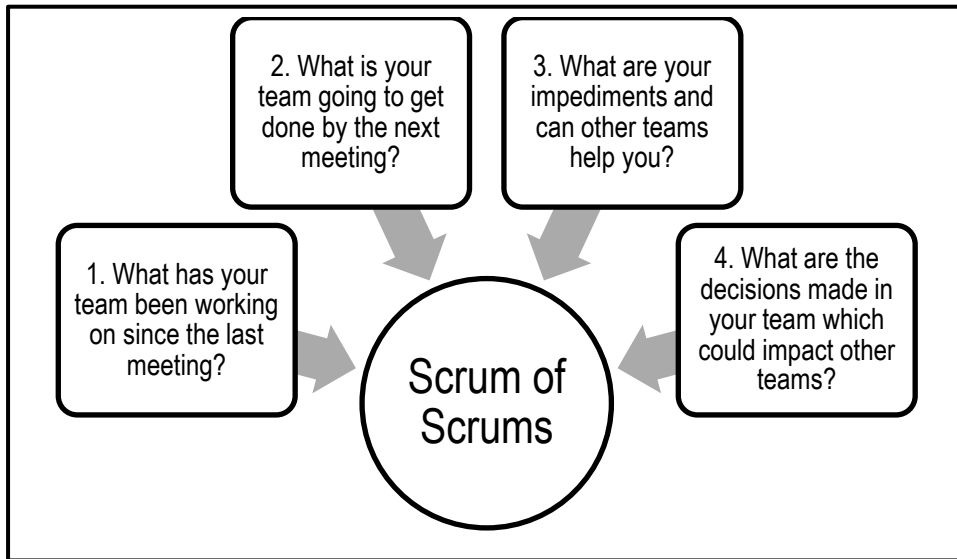


Figure 3-5: Questions Asked during a Scrum of Scrums Meeting

Typically, any inter-team issues are addressed by the interested parties in a session immediately following the Scrum of Scrums Meeting. The Chief Scrum Master facilitates this session.

The Chief Scrum Master can be chosen from the Scrum Masters assigned to a Scrum Team on the large project or can be someone else. For very large projects, it is recommended to have a Chief Scrum Master who is not also a Scrum Master for an individual project because the effort required for the Chief Scrum Master's role will prevent the Chief Scrum Master from also being able to dedicate enough time to the work with his/her Scrum Team. In either case, the Chief Scrum Master should have enough Scrum expertise to be able to foster collaboration and to help and coach others with the implementation of Scrum for a smooth delivery of the project's products.

Apart from clearing impediments and ensuring a conducive project environment for the Scrum Teams, the Chief Scrum Master also collaborates with other Scrum Masters, the Chief Product Owner, and the Product Owners in activities such as developing the list of components and resources needed in common for all teams throughout the project. He or she facilitates everything that goes beyond the realm of a single Scrum Team.

The Chief Scrum Master also interfaces with the Program Scrum Master to ensure alignment of the large project with the goals and objectives of the program.

Chief Scrum Masters should refer to the Roles Guide sections of the *SBOK® Guide* (see section 3.1.1) that describe the role of the Scrum Master. More detailed information about the role is presented in chapter 13, which describes Scaling Scrum for Large Projects.

3.6.3 Making Scrum Work in an Enterprise Environment

Applying Scrum practices to projects that are part of an enterprise model involve an understanding of how an enterprise is set up and governed. This would typically be handled using programs and portfolios.

Program—A program is a group of related projects with the objective to deliver overall business outcomes as defined in the Program Vision Statement. The Prioritized Program Backlog incorporates the Prioritized Product Backlogs for all the projects in the program.

Figure 3-6 displays a “Create Program” interface in Vabro. It includes fields for program name, description, visibility (private/public), Program Product Owner, Scrum Master, and optional additional roles, with a “Save & Continue” button.

Figure 3-6: Create Program Interface (Source: Vabro)

Portfolio—A portfolio is a group of related programs and/or projects with the objective to deliver business outcomes as defined in the Portfolio Vision Statement. The Prioritized Portfolio Backlog incorporates the Prioritized Program Backlogs for all the programs in the portfolio and the Prioritized Product Backlog of the standalone projects that are a part of the portfolio.

The problems and issues faced when using Scrum within a program or portfolio primarily involve coordination across numerous teams. This can lead to failure if not carefully managed. Tools used for communication need to be scaled to match the requirements of the many teams involved in a program or portfolio. Each Scrum Team must address not only internal communications, but also external communications with other teams and the relevant business stakeholders of the corresponding program or portfolio.

When applying Scrum to manage projects within the context of a program or portfolio, it is strongly recommended that the general principles of Scrum presented in this publication are adhered to. It is understood though, that in order to accommodate the overall program or portfolio activities and interdependencies, minor adjustments to the set of tools, as well as the organizational structure may be required. If the Scrum Guidance Body exists, it may be responsible to scrutinize the organization at different levels to understand and define the appropriate application of Scrum, and to act as a consulting body for everyone working on a project, program, or portfolio.

Programs and portfolios have separate teams with different sets of objectives. A program management team aims to deliver capabilities and to realize certain goals that contribute toward the achievement of the specific program objectives. In contrast, a portfolio team must balance the objectives of the various associated programs in order to achieve the strategic objectives of the organization as a whole.

It is important to note that as Scrum is scaled for the enterprise, additional supporting services may be needed such as architects, product managers, compliance, information security, governance bodies, and so on.

Figure 3-7 shows a “Create Portfolio” interface in the Vabro platform. It includes input fields for portfolio name, description, visibility (private/public), Portfolio Product Owner, Scrum Master, and optional additional users, with a “Save & Continue” button.

Figure 3-7: Create Portfolio Interface (Source: Vabro)

Chapter 14 presents detailed information regarding Scaling Scrum for the Enterprise.

3.6.4 Additional Core Roles in an Enterprise Environment

3.6.4.1 Program Product Owner

The Program Product Owner role is similar to that of the Product Owner role except that it aims to meet the needs of the program or business unit rather than the needs of a single Scrum Team.

The Program Product Owner defines the strategic objectives and priorities of a program. He or she is responsible for maximizing business value for the program by clearly articulating customer requirements and maintaining business justification for the program. The Program Product Owner also manages the Prioritized Program Backlog. He or she is responsible for and drives the creation and refining of deliverables at the program level, which requires coordination between the underlying projects in the program. The Program Product Owner is also responsible for coordinating with other Program Product Owners when other programs have shared dependencies and/or shared release plans.

The Program Product Owner also coordinates with the relevant Portfolio Product Owner to ensure that the program is aligned with the corresponding portfolio. The Program Product Owner interfaces with the Portfolio Product Owner to ensure alignment of the program with the goals and objectives of the portfolio. He or she is also involved with appointing Product Owners for each individual related project and ensuring that the vision, objectives, outcomes, and releases of the associated projects align with those of the program.

Program Product Owners should refer to the Roles Guide sections in the *SBOK® Guide* that define the role of the Product Owner. More detailed information about the role is presented in chapter 14, which describes Scaling Scrum for the Enterprise.

3.6.4.2 Portfolio Product Owner

The Portfolio Product Owner role is similar to the role of Product Owner and also the role of Program Product Owner except that it aims to meet the needs of the portfolio or business unit rather than the needs of a single Scrum Team or the needs of a program.

The Portfolio Product Owner makes decisions at the portfolio level. He or she will have the best perspective to help decide how to organize the enterprise to meet the vision. The Portfolio Product Owner is responsible for and drives the creation and refining of the Prioritized Portfolio Backlog.

Portfolio Product Owners should refer to the Roles Guide sections in the *SBOK® Guide* that define the role of the Product Owner. More detailed information about the role is presented in chapter 14, which describes Scaling Scrum for the Enterprise.

3.6.4.3 Program Scrum Master

The Program Scrum Master role is similar to that of the Scrum Master role except that it aims to meet the needs of the program or business unit rather than the needs of a single Scrum Team.

The Program Scrum Master is a facilitator who ensures that all project teams in the program are provided with an environment conducive to completing their projects successfully. The Program Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved in the program; provides guidance to the Scrum Masters of individual projects; clears impediments for the different project teams; coordinates with the Scrum Guidance Body to define objectives related to quality, government regulations, security, and other key organizational parameters; and ensures that Scrum processes are being effectively followed throughout the program. He or she is a facilitator, solving problems, and removing impediments at the Program Level. At the same time, he or she is also responsible for the coordination between all projects in the program and for coordinating with other programs with shared dependencies or shared release plans.

The Program Scrum Master interfaces with the Portfolio Scrum Master to ensure alignment of the program with the goals and objectives of the portfolio. He or she is also involved with appointing Scrum Masters for individual projects and ensuring that the vision, objectives, outcomes, and releases of each project in the program align with those of the program.

Program Scrum Masters should refer to the Roles Guide sections in the *SBOK® Guide* that define the role of the Scrum Master. More detailed information about the role is presented in chapter 14, which describes Scaling Scrum for the Enterprise.

3.6.4.4 Portfolio Scrum Master

The Portfolio Scrum Master role is similar to the role of the Scrum Master except that it aims to meet the needs of the portfolio or business unit rather than the needs of a single Scrum Team.

Portfolio Scrum Masters should refer to the Roles Guide sections in the *SBOK® Guide* that define the role of the Scrum Master. More detailed information about the role is presented in chapter 14, which describes Scaling Scrum for the Enterprise.

3.6.5 Examples of Projects, Programs, and Portfolios

The following are examples of projects, programs, and portfolios from different industries and sectors:

Example 1: Construction Company

- Project—Construction of a house
- Program—Construction of a housing complex
- Portfolio—All the housing projects of the company

Example 2: Aerospace Organization

- Project—Building the launch vehicle
- Program—Successful launch of a satellite
- Portfolio—All the active satellite programs

Example 3: Information Technology (IT) Company

- Project—Development of the shopping cart module
- Program—Development of a fully functional ecommerce website
- Portfolio—All the websites developed by the company so far

Figure 3-8 shows questions typically asked during a Scrum of Scrums (SoS) meeting. It includes prompts related to dependencies, progress, and impediments for multi-team coordination.

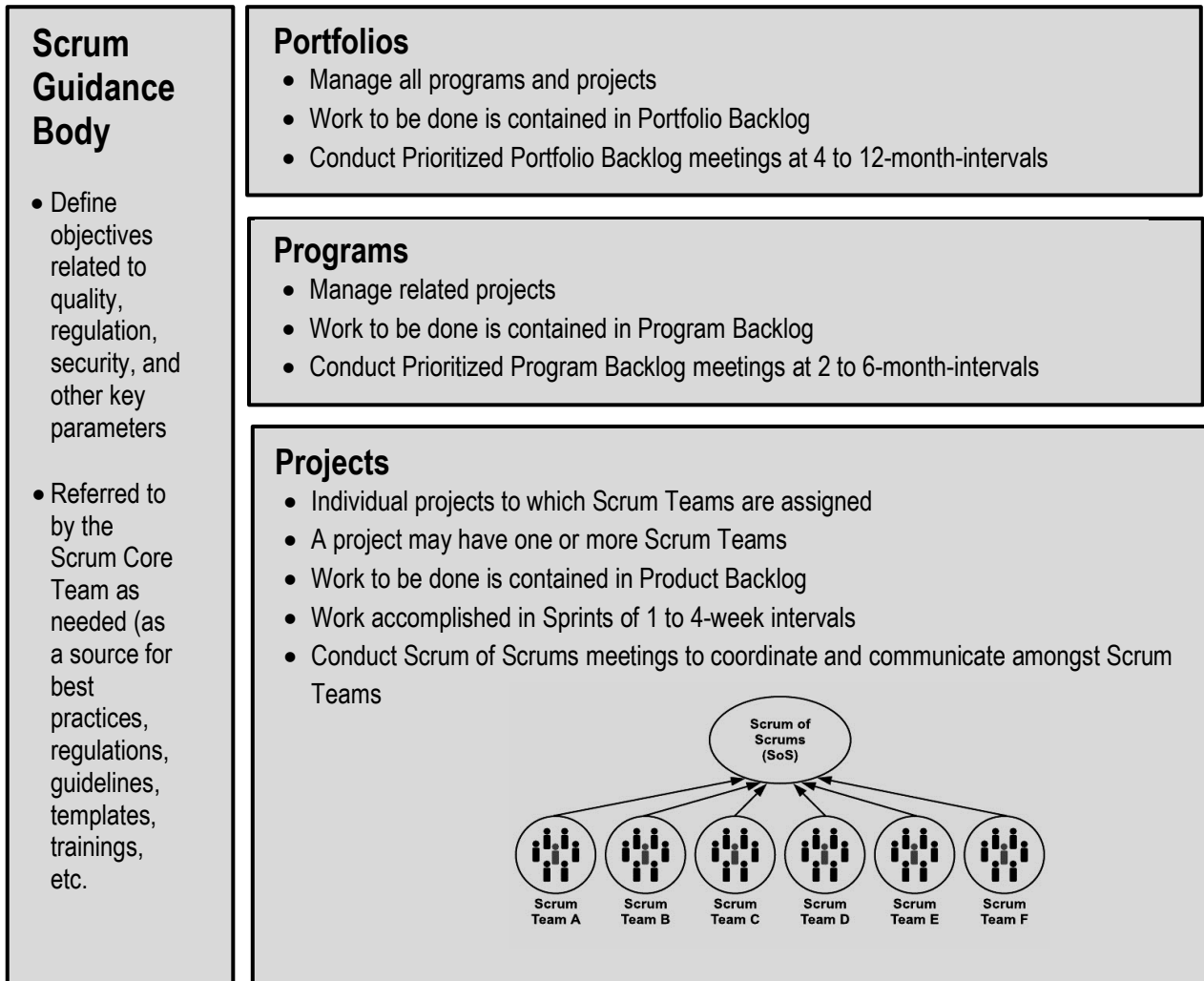


Figure 3-8: Scrum across the Organization for Projects, Programs, and Portfolios

3.7 Summary of Responsibilities

Role	Responsibilities
Scrum Team	• Takes collective responsibility and ensures that the project deliverables are created as per requirements • Assures Product Owner and Scrum Master that the allocated work is being performed according to plan • Agrees Sprint duration with the Product Owner
Product Owner/ Chief Product Owner	• Creates the project's initial overall requirements and gets the project rolling • Helps appoint appropriate people to the Scrum Master and Scrum Team roles • Helps secure the initial and ongoing financial resources for the project • Determines Project Vision • Assesses the viability and ensures delivery of the product or service • Ensures transparency and clarity of Prioritized Product Backlog Items • Decides minimum marketable release content • Provides Acceptance Criteria for the User Stories to be developed in a Sprint • Inspects deliverables • Agrees Sprint duration with the Scrum Team(s)
Scrum Master/ Chief Scrum Master	• Ensures that Scrum processes are correctly followed by all team members including the Product Owner • Ensures that development of the product or service is progressing smoothly and the Scrum Team members have all the necessary tools to get the work done • Oversees Release Planning Meeting and schedules other meetings
Program Product Owner	• Defines the strategic objectives and priorities for programs
Program Scrum Master	• Solves problems and coordinates meetings for programs
Portfolio Product Owner	• Defines the strategic objectives and priorities for portfolios
Portfolio Scrum Master	• Solves problems and coordinates meetings for portfolios
Business Stakeholder(s)	• Is a collective term that includes customers, users, and sponsors • Frequently interface with the Product Owner, Scrum Master, and Scrum Team to provide them inputs and facilitate creation of the deliverables of the project.
Scrum Guidance Body	• Establishes overall guidelines and metrics for developing role descriptions for Scrum Team members • Acts as a consultant to projects across an organization at various levels • Understands and defines appropriate levels of grouping, roles, and meetings for Scrum projects

Table 3-4: Summary of Responsibilities Relevant to Organization

3.8 Scrum vs. Traditional Project Management

Organization structure and definition of roles and associated responsibilities are some of the areas where Scrum differs in a major way from traditional project management methods.

In traditional project management methods, the organization structure is hierarchical and authority for all aspects of the project: Scrum across the Organization for Projects, Programs, and Portfolios is delegated from higher level to lower (e.g., project sponsor delegates authority to project manager and the project manager delegates authority to team members). Traditional project management methods emphasize individual accountability for project responsibilities rather than group ownership or accountability. Any deviation from the delegated authority is looked at as a sign of issues and may be escalated to a higher level in the organization hierarchy. It is usually the project manager who is responsible for successful completion of the project and he or she takes decisions on various aspects of the project, including initiating, planning, estimating, executing, monitoring, and controlling, and closing.

The emphasis in Scrum is on self-organization and self-motivation where the team assumes greater responsibility in making a project successful. This also ensures that there is team buy-in and shared ownership. This, in turn, results in team motivation leading to an optimization of team efficiencies. The Product Owner, Scrum Master, and the Scrum Team work very closely with relevant business Stakeholder(s) for refining requirements as they go through the *Develop Epic(s)*, *Create Prioritized Product Backlog*, and *Create User Stories* processes. This ensures that there is no scope for isolated planning in Scrum. Team experience and expertise in product development are used to assess the inputs needed to plan, estimate, and execute project work. Collaboration among Scrum Core Team members ensures that the project is carried out in an innovative and creative environment that is conducive to growth and team harmony.

3.9 Popular HR Theories and their Relevance to Scrum

3.9.1 Tuckman's Model of Group Dynamics

The Scrum approach and method may initially seem quite different and difficult for a new Scrum Team. A new Scrum Team, like any other new team, evolves through a four-stage process during its first Scrum project. This process is known as Tuckman's Model of group dynamics (Tuckman, 1965). The main idea is that the four stages—Forming, Storming, Norming, and Performing—are imperative for a team to develop by mitigating problems and challenges, finding solutions, planning work, and delivering results. A Scrum Master should be aware of the stage that the Scrum team is in and help the team to become a better performing team.

The four stages of the model are the following:

1. **Forming**—This is often experienced as a fun stage because everything is new, and the team has not yet encountered any difficulties with the project.
2. **Storming**—During this stage, the team tries to accomplish the work; however, power struggles may occur, and there is often chaos or confusion among team members.
3. **Norming**—This is when the team begins to mature, sort out their internal differences, and find solutions to work together. It is considered a period of adjustment.
4. **Performing**—During this stage, the team becomes its most cohesive, and it operates at its highest level in terms of performance. The members have evolved into an efficient team of peer professionals who are consistently productive.

Figure 3-9 is a diagram of Tuckman's Stages of Group Development, covering forming, storming, norming, performing, and adjourning stages of team evolution.

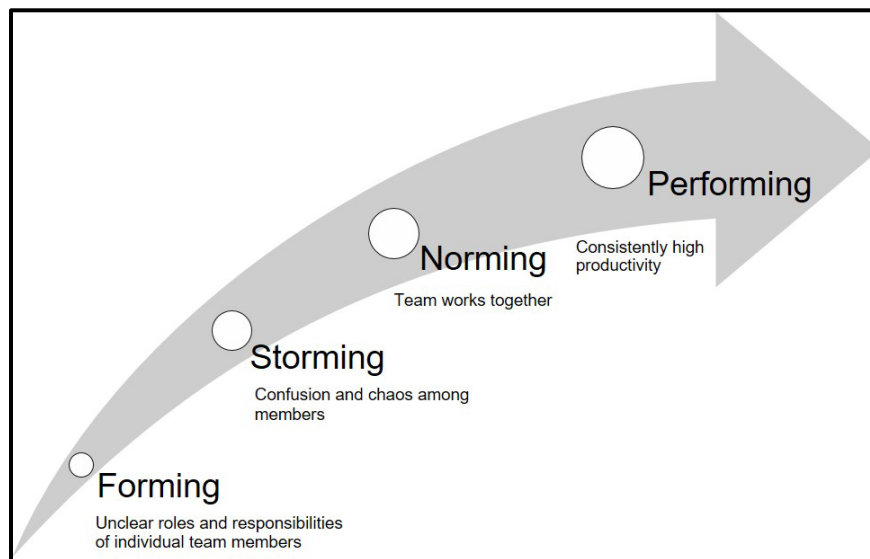


Figure 3-9: Tuckman's Stages of Group Development

3.9.2 Conflict Management

Organizations applying the Scrum framework encourage an open environment and dialogue among employees. Conflicts among Scrum Team members are resolved independently, with little or no involvement from management or others outside the Scrum Team. Conflict can be healthy when it promotes team discussions and encourages debates, as this usually results in benefits for the project and the respective team members. It is therefore important that the resolution of conflicts be encouraged, promoting an open environment where team members feel welcome to express their opinions and concerns with each other and about the project, and agree on what is to be delivered and how the work in each Sprint will be performed. Team members use conflict management techniques to manage any conflicts that arise during a Scrum project. Sources of conflict evolve primarily due to schedules, priorities, resources, reporting hierarchy, technical issues, procedures, personality, and costs.

3.9.3 Conflict Management Techniques

Usually there are four approaches to managing conflict in an organization applying Scrum processes:

1. Win–Win
2. Lose–Win
3. Lose–Lose
4. Win–Lose

3.9.3.1 Win–Win

It is usually best for team members to face problems directly with a cooperative attitude and an open dialogue to work through any disagreements to reach consensus. This approach is called *Win-Win*. Organizations implementing Scrum should promote an environment where employees feel comfortable to openly discuss and confront problems or issues and work through them to reach Win-Win outcomes.

3.9.3.2 Lose–Win

Some Team members may at times feel that their contributions are not being recognized or valued by others, or that they are not being treated equally. This may lead them to withdraw from contributing effectively to the project and agree to whatever they are being told to do, even if they disagree. This approach is called Lose-Win. This situation may happen if there are members in the team (including managers) who use an authoritative or directive style of issuing orders and/or do not treat all team members equally. This approach is not a desired conflict management technique for Scrum projects, since active contribution of every member of the team is mandatory for successful completion of each Sprint. The Scrum Master should encourage the involvement of any team members who are withdrawing from conflict situations. For example, it is important for all team members to speak and contribute at each Daily Standup Meeting so that any issues or impediments can be made known and managed effectively.

3.9.3.3 Lose–Lose

In conflict situations, team members may attempt to bargain or search for solutions that bring only a partial degree or temporary measure of satisfaction to the parties in a dispute. This situation could happen in Scrum Teams where team members try to negotiate for suboptimal solutions to a problem. This approach typically involves some “give and take” to satisfy every team member—instead of trying to solve the actual problem. This results in an overall *Lose-Lose* outcome for the individuals involved and consequently the project. The Scrum Team should be careful to ensure that team members do not get into a Lose-Lose mentality. Scrum Daily Standup and other Scrum meetings are conducted to ensure that actual problems get solved through mutual discussions.

3.9.3.4 Win–Lose

At times, a Scrum Master or another influential team member may believe he or she is a de facto leader or manager and try to exert their viewpoint at the expense of the viewpoints of others. This conflict management technique is often characterized by competitiveness and typically results in a *Win-Lose* outcome. This approach is not recommended when working on Scrum projects, because Scrum Teams are by nature self-organized and empowered, with no one person having true authority over another team member. Although the Scrum Team may include people with various levels of experience and expertise, every member is treated equally, and no person has the authority to be the primary decision maker.

3.9.4 Leadership Styles

Leadership styles vary depending on the organization, the situation, and even the specific individuals and objectives of the Scrum project. Some common leadership styles are as follows:

- **Supporting Leadership**—Leaders employ listening, empathy, commitment, and insight while sharing power and authority with team members. Supporting leaders are stewards who achieve results by focusing on the needs of the team. This style is the embodiment of the Scrum Master role.
- **Delegating**—Delegating leaders are involved in the majority of decision making; however, they delegate some planning and decision-making responsibilities to team members, particularly if they are competent to handle the assigned tasks. This leadership style is appropriate in situations where the leader is in tune with specific project details, and when time is limited.
- **Autocratic**—Autocratic leaders make decisions on their own, allowing team members little, if any involvement or discussion before a decision is made. This leadership style should only be used on rare occasions.
- **Directing**—Directing leaders instruct team members which tasks are required when they should be performed and how they should be performed.
- **Laissez Faire**—With this leadership style, the team is left unsupervised, so the leader does not interfere with their daily work activities. Often this style leads to a state of anarchy.

- **Coaching**—Coaching leaders issue instructions and then monitor team members through listening, assisting, encouraging, and presenting a positive outlook during times of uncertainty.
- **Task-Oriented**—Task-oriented leaders enforce task completion and adherence to deadlines.
- **Assertive**—Assertive leaders confront issues and display confidence to establish authority with respect.

3.9.4.1 Supporting Leadership

The preferred leadership style for Scrum projects is Supporting Leadership. Larry Spears identifies ten traits that every effective leader should possess:

1. **Listening**—Leaders are expected to listen intently and receptively to what is being said or not said. They are able to get in touch with their inner voice to understand and reflect on their own feelings.
2. **Empathy**—Good leaders accept and recognize individuals for their special and unique skills and abilities. They assume workers have good intentions and accept them as individuals, even when there are behavioral or performance issues.
3. **Healing**—The motivation and potential to heal oneself and one's relationship with others is a strong trait of leaders. Leaders recognize and take the opportunity to help their colleagues who are experiencing emotional pain.
4. **Awareness**—Awareness and particularly self-awareness is a trait of leaders. This allows them to better understand and integrate issues such as those related to ethics, power, and values.
5. **Persuasion**—Leaders use persuasion, rather than their positional authority to gain group consensus and make decisions. Rather than forcing compliance and coercion as is typical in some authoritarian management styles, leaders practice persuasion.
6. **Conceptualization**—The ability to view and analyze problems (in an organization) from a broader conceptual and visionary perspective, rather than focusing on merely the immediate short-term goals, is a unique skill of good leaders.
7. **Foresight**—Their intuitive minds allow leaders to use and apply past lessons and present realities to foresee the outcome of current situations and decisions.
8. **Stewardship**—Stewardship demands a commitment to serving others. Leaders prefer persuasion over control to ensure that they gain the trust of others in the organization.
9. **Commitment to the growth of others**—Leaders have a deep commitment to the growth of people within their organization. They take on the responsibility of nurturing the personal, professional, and spiritual growth of others (e.g., providing access to resources for personal and professional development, encouraging workers to participate in decision making).

10. **Building community**—Leaders are interested in building communities within a working environment, particularly given the shift in societies away from smaller communities to large institutions shaping and controlling human lives.

Scrum ideology supports the belief that all leaders of Scrum projects (including the Scrum Master and the Product Owner) should be supporting leaders who have all the above traits.

3.9.5 Theory X, Theory Y, and Theory Z

Douglas McGregor (1960) proposed two management theories:

- **Theory X**—Theory X leaders assume employees are inherently unmotivated and will avoid work if possible, warranting an authoritarian style of management.
- **Theory Y**—Theory Y leaders, on the other hand, assume employees are self-motivated and seek to accept greater responsibility. Theory Y involves a more participative management style.

Abraham H. Maslow (1960) proposed Theory Z and William Ouchi (1980) provided another version of Theory Z, expanding upon Theory X and Theory Y:

- **Theory Z**—In Maslow's version, Theory Z leaders assume that employees are motivated by harnessing their drive for self-transcendence without ignoring their motivations related to the hierarchy of needs. In Ouchi's version, Theory Z leaders assume that employees can be motivated by promoting stability through job security, high morale, and satisfaction both on and off the job.

Scrum projects are not likely to be successful with organizations that have Theory X leaders in the roles of Scrum Master or Product Owner. All leaders in Scrum projects should subscribe to a Theory Y and Theory Z belief system, whereby the leaders view team members as important assets in the organization and support developing each team member's skills and empowering him or her by expressing appreciation for the work he or she has completed to accomplish the project objectives.

This Page is Intentionally Left Blank.

4. BUSINESS JUSTIFICATION

4.1 Introduction

The purpose of this chapter is to understand the concept and purpose of Business Justification as it relates to Scrum projects. It is important for an organization to perform a proper business justification and create a viable Project Vision Statement prior to starting any project. This helps key decision makers understand the business need for a change or for a new product or service and the justification for moving forward with a project. It also helps the Product Owner to create a Prioritized Product Backlog along with the business expectations of senior management and business stakeholder (s).

Business Justification, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverables. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams. This chapter is divided into the following sections:

4.1.1 Roles Guide—This section provides guidance on which sections are relevant for each of the core Scrum roles: Product Owner, Scrum Master, and Scrum Team.

4.2 Value-driven Delivery—This section describes the concept of business value and its importance in any project. It also provides information regarding the responsibilities of the various individuals, including the Product Owner, involved in achieving business value.

4.3 Importance of Business Justification—This section details the importance of business justification, the factors that determine it, and how it is maintained and verified throughout the project.

4.4 Business Justification Techniques—This section describes in detail how business justification is assessed and verified using various tools.

4.5 Continuous Value Justification—This section details the importance of continuous value justification and expands on how it is achieved.

4.6 Confirm Benefits Realization—This section describes how benefits are realized throughout the project.

4.7 Summary of Responsibilities—This section defines the responsibilities relevant to business justification for project team members based on their roles.

4.8 Scrum vs. Traditional Project Management—This section highlights the business benefits of the Scrum method over traditional project management models.

4.1.1 Roles Guide

1. **Product Owner**—the Product Owner primarily conducts Business justification; therefore, this entire chapter is most applicable to this role.
2. **Scrum Master**—The Scrum Master should be familiar with this entire chapter, with primary focus on sections 4.3, 4.4, 4.6, 4.7 and 4.8.
3. **Scrum Team**—The Scrum Team should focus primarily on sections 4.3, 4.7 and 4.8.

4.2 Value-driven Delivery

A project is a collaborative enterprise to either create new products or services or to deliver results as defined in the Project Vision Statement. Projects are usually impacted by constraints of time, cost, scope, quality, people, and organizational capabilities. Usually, the results generated by projects are expected to create some form of business or service value.

Since value is a primary reason for any organization to move forward with a project, Value-driven delivery must be the main focus. Delivering value is ingrained in the Scrum framework. Scrum facilitates delivery of value early on in the project and continues to do so throughout the project lifecycle.

4

One of the key characteristics of any project is the uncertainty of results or outcomes. It is impossible to guarantee project success at completion, irrespective of the size or complexity of a project. Considering this uncertainty about achieving success, it is therefore important to start delivering results as early on the project as possible. This speedy delivery of results, and thereby value, provides an opportunity for reinvestment and proves the worth of the project to interested business stakeholders.

In order to provide value-driven delivery, it is important to:

1. Understand what adds value to customers and users and to prioritize the high value requirements on the top of the Prioritized Product Backlog.
2. Decrease uncertainty and constantly address risks that can potentially decrease value if they materialize. Also work closely with project business stakeholders showing them product increments at the end of each Sprint, enabling effective management of changes.
3. *Create Deliverables* based on the priorities determined by producing potentially shippable product increments during each Sprint so that customers start realizing value early on in the project.

The concept of value-driven delivery in Scrum makes the Scrum framework extremely attractive for business stakeholders and senior management. This concept is quite different when compared with traditional project management models where:

1. Requirements are not prioritized by business value.
2. Changing requirements after project initiation is difficult and can only be done through a time-consuming change management process.
3. Value is realized only at the end of the project when the final product or service is delivered.

Figure 4-1 contrasts Value-Driven Delivery in Scrum Projects with Traditional Project approaches.

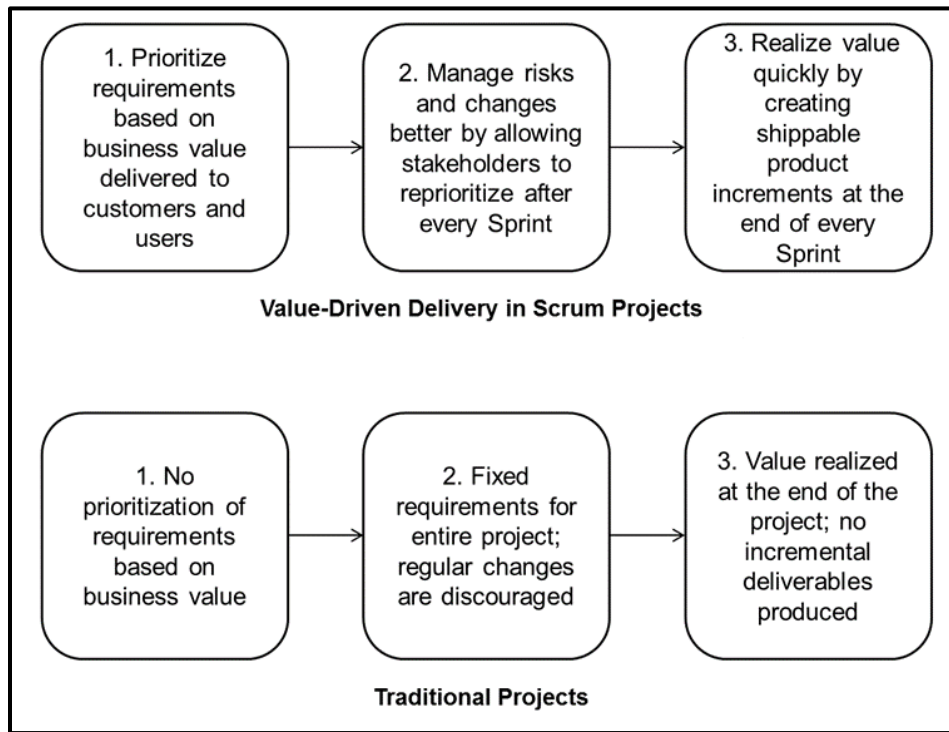


Figure 4-1: Delivering Value in Scrum vs. Traditional Projects

4.2.1 Responsibilities of the Product Owner in Business Justification

The responsibility of prioritizing and delivering business value in an organization for projects lies primarily with the Product Owner. For programs and portfolios, the responsibility lies with the Program Product Owner and Portfolio Product Owner, respectively. Their role is to act as effective representatives of the customer and/or sponsor. The guidelines for evaluating and measuring business value may typically be set forth by a Scrum Guidance Body.

Table 4-2 illustrates the business justification responsibilities in a hierarchical order.

Portfolio Product Owner	• Delivers value for portfolios • Creates business justification for portfolios • Provides value guidance for programs • Approves business justification for programs

Program Product Owner	• Delivers value for programs • Creates business justification for programs • Provides value guidance for projects • Approves business justification for projects
Product Owner	• Delivers value for projects • Creates business justification for projects • Confirms benefit realization to business stakeholders

Table 4-1: Hierarchy for Business Justification Responsibilities

4.2.2 Responsibilities of Other Scrum Roles in Business Justification

It is important to note that although the Product Owner is primarily responsible for business justification, other people working in Scrum projects also contribute significantly as follows:

1. The **sponsor** provides funding for the project and constantly monitors the project to confirm realization of benefits.
2. **Customers** and **users** are involved in defining the prioritized list of requirements and User Stories in the Prioritized Product Backlog, reviewing deliverables after every Sprint or release, and confirming that benefits are realized.
3. The **Scrum Guidance Body** may provide guidelines and recommendations related to business justification techniques and confirming benefits realization and so forth. Such guidelines and recommendations may then be referred to by Scrum Core Teams and business stakeholders(s).
4. The **Scrum Master** facilitates creation of the project's deliverables; manages risks, changes, and impediments during *Conduct Daily Standup*, *Retrospect Sprint*, and other Scrum processes. The Scrum Master coordinates with the Scrum Team to create the deliverables and with the Product Owner and other business stakeholders to ensure that benefits from the project are realized.
5. The **Scrum Team** works on creating the deliverables of the project and contributes to realizing business value for all business stakeholders and the project. The Scrum Team is also involved in the *Develop Epic(s)*, *Create Prioritized Product Backlog*, *Create User Stories*, *Estimate User Stories*, *Commit User Stories*, and associated processes where the business requirements are defined and prioritized. The Scrum Team also helps with identifying risks and submits Change Requests for improvements in Sprint Retrospect Meetings and other meetings.

4.3 Importance of Business Justification

Business justification demonstrates the reasons for undertaking a project. It answers the question “Why is this project needed?” Business justification drives all decision making related to a project. So, it is important to assess the viability and achievability of a project not only before committing significant expenditures or investment at initial stages of the project but also to verify the business justification for continuance throughout the project’s lifecycle. A project should be terminated if it is found to be unviable; the decision should be escalated to the relevant business stakeholders and to senior management. The business justification for a project must be assessed at the beginning of the project, at pre-defined intervals throughout the project, and at any time when significant issues or risks that threaten the project viability arise.

4.3.1 Factors Used to Determine Business Justification

There are numerous factors a Product Owner must consider when assessing the business justification for a project. The following are some of the key factors:

- 1. Project Reasoning**

Project reasoning includes all factors which necessitate the project, whether positive or negative, chosen or not (e.g., inadequate capacity to meet existing and forecasted demand, decrease in customer satisfaction, low profit, legal requirement, etc.).

- 2. Business Needs**

Business needs are those business outcomes that the project is expected to fulfill, as documented in the Project Vision Statement.

- 3. Project Benefits**

Project benefits include all measurable improvements in a product, service, or result which could be provided through successful completion of a project.

- 4. Opportunity Cost**

Opportunity cost covers the next best business option or project that was discarded in favor of the current project.

- 5. Major Risks**

Risks include any uncertain or unplanned events that may affect the viability and potential success of the project.

6. Project Timescales

Timescales reflect the length or duration of a project and the time over which its benefits will be realized.

7. Project Costs

Project costs are investment and other development costs for a project.

4.3.2 Business Justification and the Project Lifecycle

4

Business justification is first assessed prior to a project being initiated and is continuously verified throughout the project lifecycle. The following steps capture how business justification is determined:

1. Assess and Present a Business Case

Business justification for a project is typically analyzed and confirmed by the Product Owner. It is documented and presented in the form of a project business case prior to Initiate phase and involves considering the numerous factors specified in section 4.4.1. Once documented, the Product Owner should create a Project Vision Statement and obtain approval of the Project Vision Statement from the key decision-makers in the organization. This consists of executives and/or some form of a project or program management board.

2. Continuous Value Justification

Once the decision makers approve the Project Vision Statement, it is then baselined and forms the business justification. The business justification is validated throughout project execution, typically at predefined intervals or milestones, such as during portfolio, program, and Prioritized Product Backlog Review Meetings and when major issues and risks that threaten project viability are identified. This could happen in several Scrum processes including *Conduct Daily Standup* and *Refine Prioritized Product Backlog*. Throughout the project, the Product Owner should keep the business justification in the Project Vision Statement updated with relevant project information to enable the key decision makers to continue making informed decisions.

3. Confirm Benefits Realization

The Product Owner confirms the achievement of organizational benefits throughout the project, as well as upon completion of the User Stories in the Prioritized Product Backlog. Benefits from Scrum projects are realized during *Demonstrate and Validate Sprint*, *Retrospect Sprint*, *Ship Deliverables* and *Retrospect Release* processes.

Figure 4-2 summarizes the business justification process, linking value analysis with project initiation and lifecycle phases to ensure project alignment with business goals.

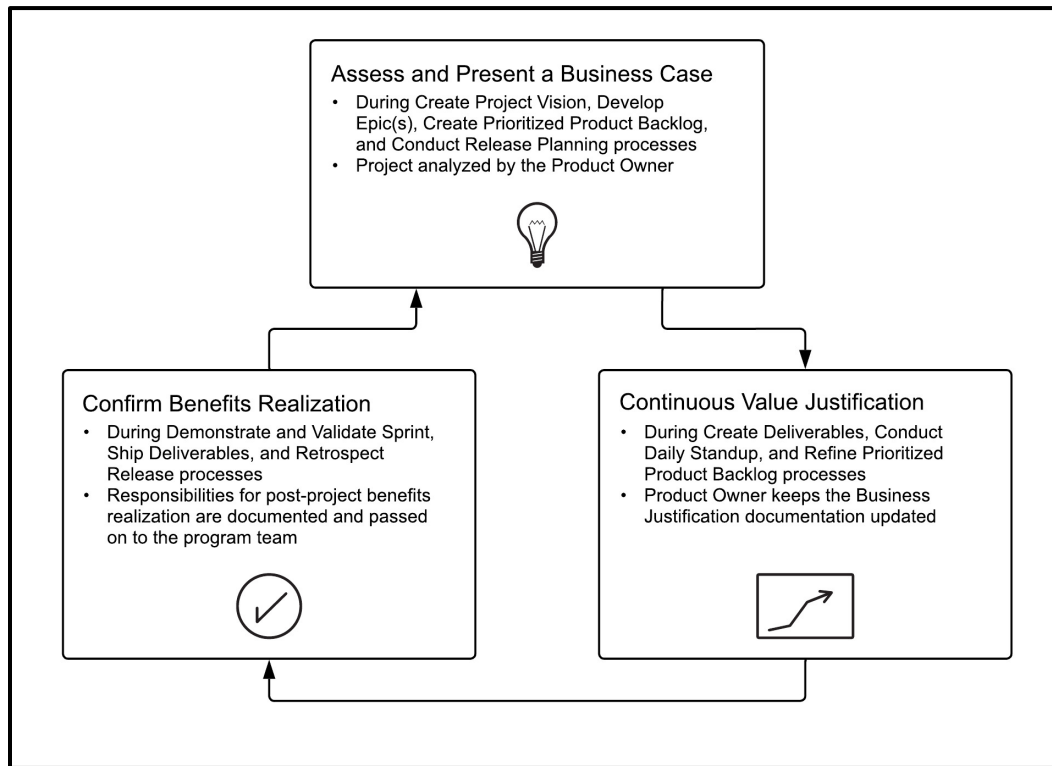


Figure 4-2: Business Justification and the Project Lifecycle

4.4 Business Justification Techniques

The following sections deal with some of the tools used to assess and evaluate business justification, as well as some other aspects associated with project justification and project selection. It is not necessary or even recommended to use every available technique for every project. Some techniques are not appropriate depending on the specific project, and techniques may be used to assess projects individually or to compare the expected value of multiple projects.

The Scrum Guidance Body (SGB), which can be a panel of experts or a set of documents on organizational standards and procedures, defines the guidelines and metrics that will be used to assess business value. Each respective Product Owner, however, is responsible for performing the activities that verify and track business value for his or her respective projects, programs, or portfolios.

4

4.4.1 Estimation of Project Value

The value to be provided by business projects can be estimated using various methods such as Return on Investment (ROI), Net Present Value (NPV), and Internal Rate of Return (IRR).

1. Return on Investment (ROI)

Return on Investment (ROI) when used for project justification, assesses the expected net income to be gained from a project. It is calculated by deducting the expected costs or investment of a project from its expected revenue and then dividing this (net profit) by the expected costs in order to get a return rate. Other factors such as inflation and interest rates on borrowed money may be factored into ROI calculations.

ROI formula:

$$\text{ROI} = (\text{Project Revenue} - \text{Project Cost}) / \text{Project Cost}$$

Example: The ROI for a project that will cost \$125,000 to develop, with expected financial benefits estimated at \$300,000 is calculated as follows:

$$\text{ROI} = (\$300,000 - \$125,000) / \$125,000 = 1.4$$

Therefore, the ROI is 1.4 times the investment (or 140%).

Frequent product or service increments is a key foundation of Scrum that allows earlier verification of ROI. This aids in assessing the justification of continuous value.

2. Net Present Value (NPV)

Net Present Value (NPV) is a method used to determine the current net value of a future financial benefit, given an assumed inflation or interest rate. In other words, NPV is the total expected income or revenue from a project, minus the total expected cost of the project, considering the time-value of money.

Example: Which of the following two projects is better to select if NPV is used as the selection criterion?

- Project A has a NPV of \$1,500 and will be completed in 5 years.
- Project B has a NPV of \$1,000 and will be completed in 1 year.

Solution: Project A, since its NPV is higher; the fact that Project B has a shorter duration than Project A is not considered here, because time is already accounted for in the NPV calculations (i.e., due to the fact that it is the current, not future value that is being considered in the calculation).

3. Internal Rate of Return (IRR)

Internal Rate of Return (IRR) is a discount rate on an investment in which the present value of cash inflows is made equal to the present value of cash outflows for assessing a project's rate of return. When comparing projects, one with a higher IRR is typically better.

Though IRR is not used to justify projects as often as some other techniques, such as NPV, it is an important concept to know.

Example: Based on IRR, which project is most desirable?

- Project A, which has an IRR of 15% and will be completed in 5 years.
- Project B, which has an IRR of 10% and will be completed in 1 year.

Solution: Project A, since its IRR is higher; the fact that Project B has a shorter duration than Project A is not considered here because time is already considered in the IRR calculations (i.e., as with NPV, it is the current, not future value that is used to determine the IRR).

4.4.2 Planning for Value

After justifying and confirming the value of a project, the Product Owner should consider the organizational policies, procedures, templates, and general standards dictated by the Scrum Guidance Body (or similar organizational project board or office) when planning a project; at the same time maximizing Value-driven delivery. The onus for determining *how* the value is created falls on the business stakeholders (sponsor, customers, and/or users), while the Scrum Team concentrates on *what* is to be developed. Some common tools recommended by a Scrum Guidance Body might include the following:

1. Value Stream Mapping

Value Stream Mapping uses process flowcharts to illustrate the flow of steps needed to complete a process. This technique may be used to streamline a process by helping to identify and eliminate non-value-added elements and to increase efficiencies. Value Stream Mapping can be used to streamline Scrum processes, for example, to improve Sprint velocity.

Figure 4-3 shows a Value Stream Map that visualizes product development flow, helping identify bottlenecks and streamline value delivery.

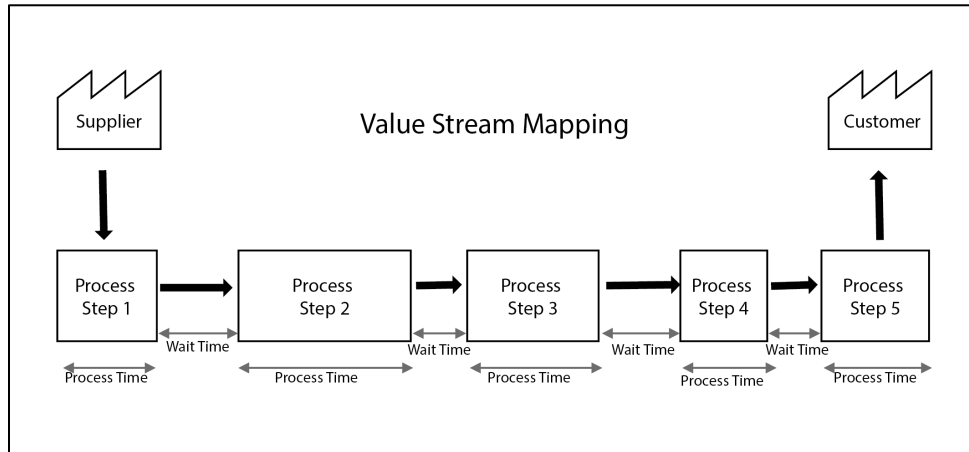


Figure 4-3: Value Stream Mapping

2. Customer Value-based Prioritization

Customer Value-based Prioritization places primary importance on the customer and strives to implement User Stories with the highest value first. Such high value User Stories are identified and moved to the top of the Prioritized Product Backlog. A team can use a variety of prioritization schemes to determine high-value features.

- **Simple Schemes**—Simple schemes involve labeling items as Priority “1”, “2”, “3” or “High”, “Medium” and “Low” and so on. Although this is a simple and straightforward approach, it can become problematic because there is often a tendency to label everything as Priority “1” or “High”. Even “High”, “Medium,” and “Low” prioritization schemes can encounter similar difficulties.
- **MoSCoW Prioritization**—The MoSCoW prioritization scheme derives its name from the first letters of the phrases “Must have,” “Should have,” “Could have,” and “Won’t have.” This prioritization method is more effective than simple schemes. The labels are in decreasing order of priority with “Must have” features being those without which the product will have no value and “Won’t have” features being those that, although they would be nice to have, are not necessary to be included.
- **Monopoly Money**—This technique involves giving the customer “monopoly money” or “false money” equal to the amount of the project budget and asking them to distribute it among the User Stories under consideration. In this way, the customer prioritizes based on what they are willing to pay for each User Story.

- **100-Point Method**—Dean Leffingwell and Don Widrig (2003) developed the 100-Point Method. It involves giving the customer 100 points they can use to vote for the features that they feel are most important.
- **Kano Analysis**—The Kano analysis was developed by Noriaki Kano (1984) and involves classifying features or requirements into four categories based on customer preferences:
 1. *Exciters/Delighters*: Features that are new, or of high value to the customer
 2. *Satisfiers*: Features that offer value to the customer
 3. *Dissatisfiers*: Features which, if not present, are likely to cause a customer to dislike the product, but do not affect the level of satisfaction if they are present
 4. *Indifferent*: Features that will not affect the customer in any way and should be eliminated

Figure 4-4 illustrates Kano Analysis, categorizing features into must-have, performance, and excitement attributes based on customer satisfaction and prioritization.

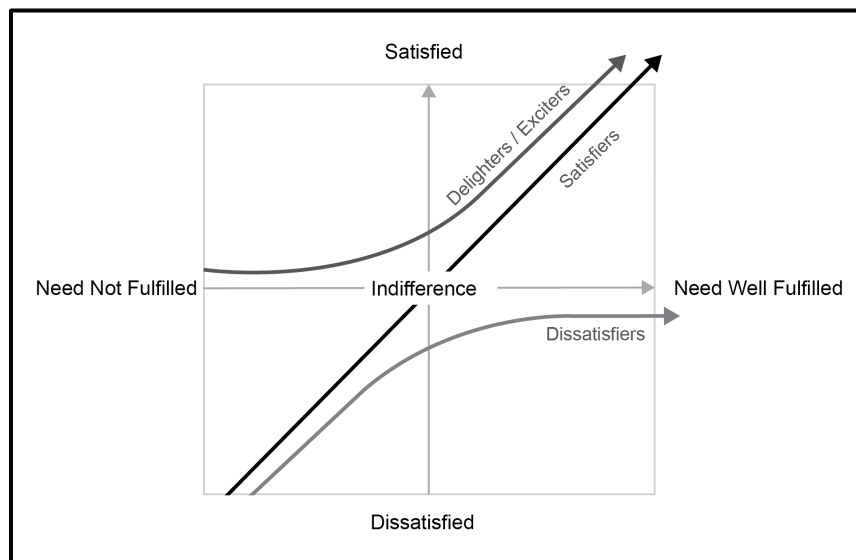


Figure 4-4: Kano Analysis

Interestingly, features usually move down the classification list over time; customers will come to expect features (e.g., cameras on phones) and these features will move from being exciters/delighters to satisfiers and eventually to dissatisfiers.

4.4.3 Relative Prioritization Ranking

A simple listing of User Stories in order of priority is an effective method for determining the desired User Stories for each iteration or release of the product or service. The purpose is to create a simple, single list with the goal of prioritizing features, rather than being distracted by multiple prioritization schemes.

This simple list also provides a basis for incorporating changes and identified risks when necessary. Each change or identified risk can be inserted in the list based on its priority relative to the other User Stories in the list. Typically, latest changes will be included at the expense of features that have been assigned a lower priority.

Defining the Minimum Marketable Features (MMF) is extremely important during this process, so that the first release or iteration happens as early as possible, leading to increased ROI. Normally, these User Stories would rank highest in priority.

4.4.4 Story Mapping

This technique is used to provide a visual outline of the product and its key components. Story Mapping, first formulated by Jeff Patton (2005), is commonly used to illustrate product roadmaps. Story maps depict the sequence of product-development iterations and map out which User Stories will be included in the first, second, third, and subsequent releases. When applying Scrum practices, it is well understood that this is only a current outlook and is expected to be reviewed and changed frequently.

The story map example in Figure 4-6 illustrates how the Scrum Team plans out different releases and assigns higher priority to the release happening in the near future. The team is expected to have a better understanding of the User Stories in the upcoming release, so the further out a release, the more likely details could change.

Figure 4-5 presents a Story Mapping diagram, organizing user stories by tasks and workflows to visualize the product from a user-centric perspective.

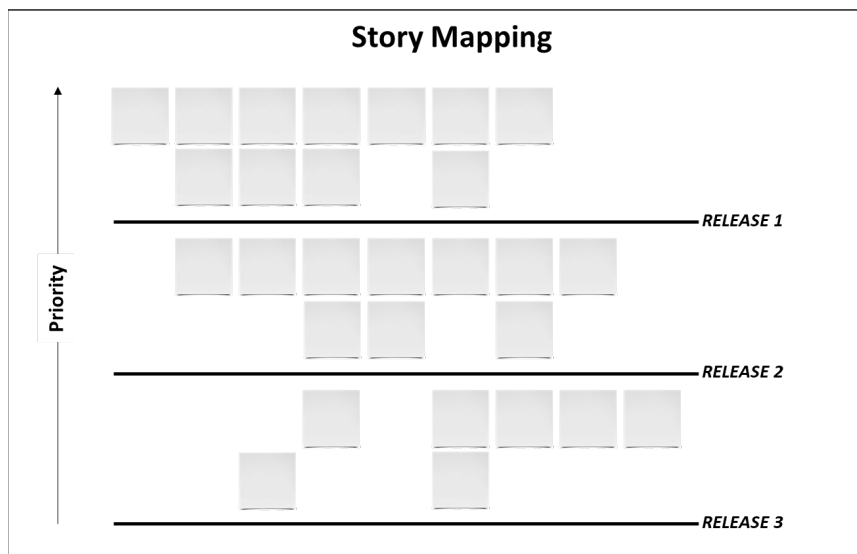


Figure 4-5: Story Mapping

4.5 Continuous Value Justification

Business value should be assessed regularly to determine whether the justification or viability of executing the project continues to exist. Frequent assessment of investment in the project relative to business value being created qualifies the continued viability of a project. The expected requirements from the project may change frequently, which can impact both project investment and value creation. A key aspect of Scrum is its ability to quickly adjust to chaos created by a rapidly changing business model. In projects with ambiguous user requirements and significant potential for frequent changes, Scrum provides considerable advantages over other development models. Monitoring the rate of delivering value is an important requirement for Scrum projects. Periodically tracking and reporting the creation of value assists in assessing project status and provides essential information to the customer and other business stakeholders.

4.5.1 Sprint Burndown or Burnup Chart

Burn Charts (Burndown or Burnup) are used to track progress in a Scrum project. A Burndown Chart is a graph that depicts the amount of work remaining in relation to the remaining time. Unlike the Burndown Chart, a Burnup Chart depicts what has been completed in relation to the remaining time. Burn Charts are used to track the Scrum Team's progress during a Sprint and to get an early indication if the team will be able to complete all the User Stories that were committed to for that Sprint. If the team members believe they will not be able to complete all the committed User Stories, they can act early during the Sprint to achieve the best possible outcome.

The initial Sprint Burndown Chart shows how the team envisions getting the work done. Because the team just committed to a set of User Stories and the associated tasks for the current Sprint, and because it is expected that the team meets its commitments, the initial planned burndown shows that on the last day of the Sprint, no work will be left to be done. That means that all work will ideally be done by the last day. The Burndown Chart should be updated by the team at the end of each day to show progress as work is completed.

Figure 4-6 shows a Sprint Burndown Chart, plotting remaining work against time to visualize sprint progress and identify potential delivery delays.

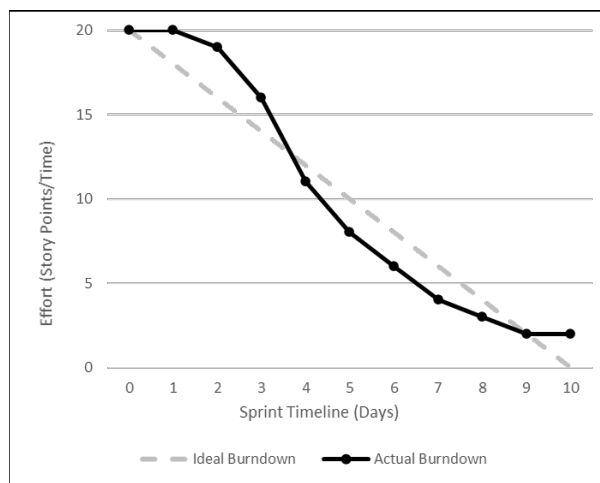


Figure 4-6: Sprint Burndown Chart

Figure 4-7 presents a Sprint Burnup Chart, comparing completed work versus scope changes over time, helping teams visualize progress against the sprint goal.

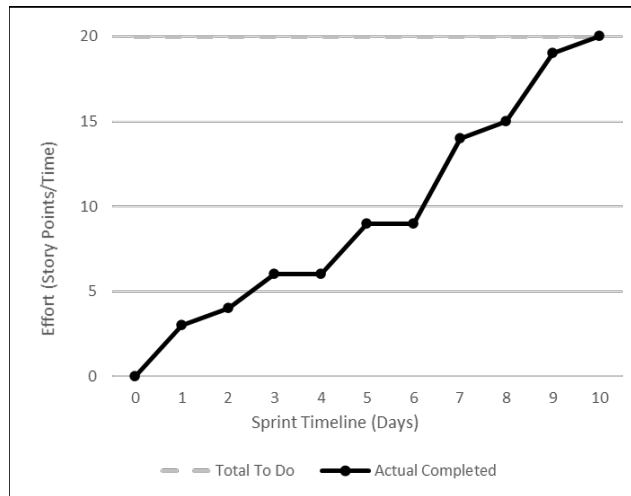


Figure 4-7: Sprint Burnup Chart

The Burndown Chart can be updated very easily, not only to show progress, but also to adjust for any over- or under-estimation of effort. It also provides a much better indication of a potential mismatch between remaining effort and remaining time, than a Burnup Chart does. Therefore, there are only very few Scrum teams that use Burnup Charts to track team progress during a Sprint.

4.5.2 Cumulative Flow Diagram (CFD)

A Cumulative Flow Diagram (CFD) is a useful tool for reporting and tracking project performance. It provides a simple, visual representation of project progress at a particular point in time. It is usually used to provide a higher-level status of the overall project and not daily updates for individual Sprints.

Figure 4-9 is an example of a CFD for a large project. It shows how many User Stories are yet to be created, in process of being created, and have been created. As customer requirements change, there is a change in the Cumulative User Stories which have to be delivered. Change points 1 and 2 are where the Product Owner removed existing user Stories in the Risk Adjusted Prioritized Product Backlog and Change points 3 and 4 are where the Product Owner added new User Stories in the Risk Adjusted Prioritized Product Backlog

This type of diagram can be a great tool for identifying roadblocks and bottlenecks within processes. For example, if the diagram shows one band becoming narrower while the previous band is becoming wider over time, there may be a bottleneck, and changes may be needed to increase efficiency and/or improve project performance.

Figure 4-8 is a sample Cumulative Flow Diagram (CFD), showing task distribution across workflow stages to highlight bottlenecks and process efficiency.

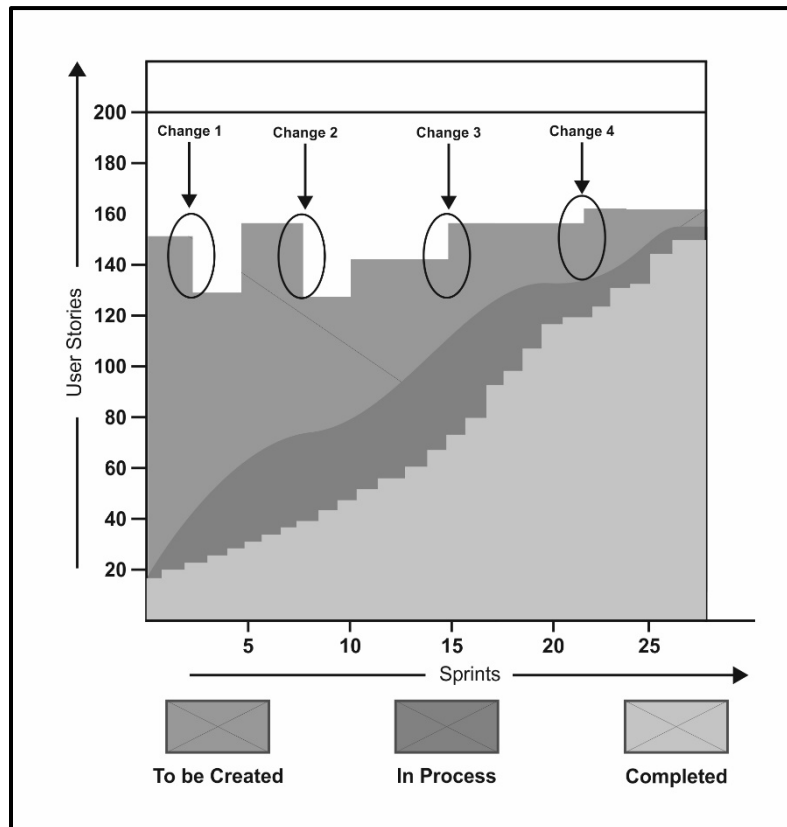


Figure 4-8: Sample Cumulative Flow Diagram (CFD)

4.6 Confirm Benefits Realization

Throughout a project, it is important to verify whether benefits are being realized. Whether the products of a Scrum project are tangible or intangible, appropriate verification techniques are required to confirm that the team is creating the deliverables that will achieve the benefits and value defined at the beginning of the project.

4.6.1 Prototypes, Simulations, and Demonstrations

4

Demonstrating prototypes to customers and simulating their functionalities are commonly used techniques for confirming value.

Often, after using the features or having them demonstrated, customers can more clearly determine whether the features are adequate and suitable for their needs. They might realize a need for additional features or may decide to modify previously defined feature requirements. In product development, this customer experience has come to be known as IKIWISI (I'll Know It When I See It).

Through demonstrations or access to early iterations, customers can also evaluate to what degree the team has successfully interpreted their requirements and met their expectations.

4.7 Summary of Responsibilities

Role	Responsibilities
Scrum Team	- Ensures that project deliverables are completed in accordance with agreed Acceptance Criteria - Performs Continuous Value Justification for projects
Product Owner/ Chief Product Owner	- Ensures value delivery for projects - Maintains the business justification for projects - Confirms and communicates project benefits to business stakeholders
Scrum Master/ Chief Scrum Master	- Ensures the desired outcomes of the project are communicated to and understood by the Scrum Team - Performs Continuous Value Justification for projects
Program Product Owner	- Ensures value delivery for programs - Creates the business justification for programs - Provides value guidance for projects within a program - Approves the business justification of projects within a program
Program Scrum Master	- Ensures the desired outcomes of the program are communicated and understood - Performs Continuous Value Justification for programs
Portfolio Product Owner	- Ensures value delivery for portfolios - Creates the business justification for portfolios - Provides value guidance for programs within portfolios - Approves the business justification of programs within a portfolio
Portfolio Scrum Master	- Ensures the desired outcomes of the portfolio are achieved - Performs Continuous Value Justification for portfolios
Business Stakeholder(s)	- Helps prioritize User Stories and requirements in the Prioritized Product Backlog - Communicates with Scrum Team and confirms realization of value at the end of every Sprint, Release, and the project
Scrum Guidance Body	- Establishes overall guidelines and metrics for evaluating value - Acts in a consulting capacity and provides guidance for projects, programs, and portfolios as required

Table 4-2: Summary of Responsibilities Relevant to Business Justification

4.8 Scrum vs. Traditional Project Management

Traditional projects emphasize on extensive upfront planning and adherence to the project plan created by the project manager. Usually, changes are managed through a formal change management system and value is created at the end of the project when the final product is delivered.

In Scrum projects, extensive long-term planning is not done prior to project execution. Planning is done in an iterative manner before each Sprint. This allows a quick and effective response to change, which results in lower costs and increased profitability and Return on Investment (ROI). Moreover, value-driven delivery (section 4.3) is a key benefit of the Scrum framework and provides significantly better prioritization and quicker realization of business value. Because of the iterative nature of Scrum development, there is always at least one release of the product with Minimum Marketable Features (MMF) available. Even if a project is terminated, there are usually some benefits or value created prior to termination.

This Page is Intentionally Left Blank.

5. QUALITY

5.1 Introduction

The purpose of this chapter is to define quality as it relates to projects and to present the Scrum approach to achieve the required levels of quality. *Quality*, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

5

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams. This chapter is divided into the following sections:

5.1.1 Roles Guide—This section provides guidance on which sections are relevant for each Scrum role: Product Owner, Scrum Master, and Scrum Team.

5.2 Quality Defined—This section provides the Scrum definition of quality, with a clear distinction from scope, and describes the relationship between quality and business value.

5.3 Acceptance Criteria and the Prioritized Product Backlog—This section emphasizes the importance of Acceptance Criteria, the Prioritized Product Backlog, and their relationship. It also explains the Scrum definition of Done.

5.4 Quality Management in Scrum—This section provides details on quality planning, quality control, and quality assurance in the context of Scrum.

5.5 Summary of Responsibilities—This section describes the responsibilities relevant to quality for each person or role in a project.

5.6 Scrum vs. Traditional Project Management—This section highlights the benefits of quality management in Scrum method over traditional project management models.

5.1.1 Roles Guide

1. Product Owner—It is important for anyone assuming the role of Product Owner in Scrum projects to read this complete chapter.
2. Scrum Master—The Scrum Master should also be familiar with this entire chapter with primary focus on sections 5.3, 5.4, 5.5.3, and 5.6.
3. Scrum Team—The Scrum Team should focus on sections 5.3, 5.4, and 5.6.

5.2 Quality Defined

There are numerous ways to define quality.

In Scrum, quality is defined as the ability of the completed product or deliverables to meet the Acceptance Criteria and achieve the business value expected by the customer.

To ensure that a project meets quality requirements, Scrum adopts an approach of continuous improvement whereby the team learns from experience and stakeholder engagement to constantly keep the Prioritized Product Backlog updated with any changes in requirements. The Prioritized Product Backlog is simply never complete until the closure or termination of the project. Any changes to the requirements reflect changes in the internal and external business environment and allow the team to continually work and adapt to achieve those requirements. Since Scrum requires work to be completed in increments during Sprints, this means that errors or defects get noticed earlier through repetitive quality testing, rather than when the final product or service is near completion. Moreover, important quality-related tasks (e.g., development, testing, and documentation) are completed as part of the same Sprint by the same team—this ensures that quality is inherent in any Done deliverable created as part of a Sprint. Thus, continuous improvement with repetitive testing optimizes the probability of achieving the expected quality levels in a Scrum project. Constant discussions between the Scrum Core Team and business stakeholders (including customers and users) with actual increments of the product being delivered at the end of every Sprint, ensures that the gap between customer expectations from the project and actual deliverables produced is constantly reduced.

5.2.1 Quality and Scope

Scope and quality requirements for a project are determined by taking into consideration numerous factors such as the following:

- The business need the project will fulfill
- The capability and willingness of the organization to meet the identified business need
- The current and future needs of the target audience

The scope of a project is the total sum of all the product increments and the work required for developing the final product. Quality is the ability of the deliverables to meet the quality requirements for the product and satisfy customer needs. In Scrum, the scope and quality of the project are captured in the Prioritized Product Backlog, and the scope for each Sprint is determined by refining the large Prioritized Product Backlog Items (PBIs) into a set of small but detailed User Stories that can be planned, developed, and verified within a Sprint.

The Prioritized Product Backlog is continuously refined by the Product Owner. The Product Owner ensures that any User Stories that the Scrum Team is expected to do in a Sprint are refined prior to the start of the Sprint. In general, the most valuable requirements in solving the customers' problems or meeting their needs are prioritized as high and the remaining are given a lower priority. Less important User Stories are developed in subsequent Sprints or can be left out altogether according to the customer's requirements. Throughout the entire project, the Product Owner, customer, and the Scrum Team discusses and changes the list of features of the product to comply with the changing needs of the customers.

5.2.2 Quality and Business Value

Quality and business value are intricately linked. Therefore, it is critical to understand the quality and scope of a project in order to correctly map the outcomes and benefits the project and its product must achieve in order to deliver business value. To determine the business value of a product, it is important to understand the business need that drives the requirements of the product. Thus, business needs determine the product required, and the product, in turn provides the expected business value.

Quality is a complex variable. An increase in scope without increasing time or resources tends to reduce quality. Similarly, a reduction in time or resources without decreasing scope also generally results in a decrease in quality. Scrum practices encourage maintaining a “sustainable pace” of work, which helps improve quality over a period of time.

The Scrum Guidance Body may define minimum quality requirements and standards required for all projects in the organization. The standards must be adhered to by all Scrum Teams in the organization.

5.3 Acceptance Criteria and the Prioritized Product Backlog

The Prioritized Product Backlog is a single requirements document that defines the project scope by providing a prioritized list of features of the product or service to be delivered by the project. The required features are described in the form of User Stories. User Stories are specific requirements outlined by relevant business stakeholders as they pertain to the proposed product or service. Each User Story will have associated User Story Acceptance Criteria (also referred to as “Acceptance Criteria”), which are the objective components by which a User Story’s functionality is judged. Acceptance Criteria are developed by the Product Owner according to his or her expert understanding of the customer’s requirements. The Product Owner then communicates the User Stories in the Prioritized Product Backlog to the Scrum Team members, and their agreement is sought. Acceptance Criteria should explicitly outline the conditions that User Stories must satisfy. Clearly defined Acceptance Criteria are crucial for timely and effective delivery of the functionality defined in the User Stories, which determines the success of the project.

At the end of each Sprint, the Product Owner uses these criteria to verify the completed deliverables; and can either accept or reject individual deliverables and their associated User Stories. If deliverables are accepted by the Product Owner, then the User Story is considered Done. A clear definition of Done is critical because it helps clarify requirements and allows the team to adhere to quality norms. It also helps the team think from the user’s perspective when working with User Stories. User Stories corresponding to rejected deliverables are added back to the Prioritized Product Backlog to be considered for completion in future Sprints. The rejection of a few individual deliverables and their corresponding User Stories is not a rejection of the final product or product increment. The product or product increment could potentially be shippable even if a few User Stories are rejected.

Figure 5-1 is a Project Increment Flow Diagram, showing how defined acceptance criteria lead to validated product increments delivered at the end of each sprint.

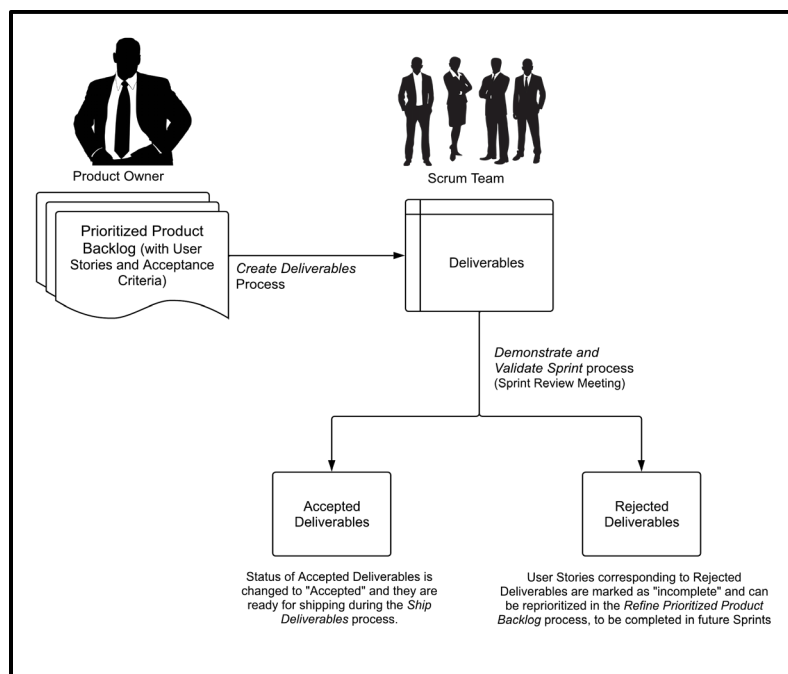


Figure 5-1: Project Increment Flow Diagram

5.3.1 Writing Acceptance Criteria

Acceptance Criteria are unique to each User Story and are not a substitute for a requirements list. The following is an example of how Acceptance Criteria could be written.

Example:

Persona: Janine is a married 36-year-old working professional with a family of three children. She is a busy, successful woman who balances her professional and personal life. She is comfortable with technology and is an early adopter of innovative services and products. She is always connected to the internet through multiple devices and regularly shops on ecommerce portals. User Story: “As an online grocery shopper Janine, I should be able to save and view my draft order from any of my devices so that I can complete the order process at my convenience.”

Acceptance Criteria:

- Every in-progress order is saved every five seconds to the logged-in user account as a draft order
- New draft orders show up as notifications on any devices the user logs in

It is important for a Product Owner to note that User Stories that fulfill most, but not all, Acceptance Criteria cannot be accepted as Done. Scrum projects operate in Time-boxed Sprints, with a dedicated Sprint Backlog for each Sprint. Often, the last bit of work might be the most complicated part of a User Story and might take longer than expected. If incomplete User Stories were given partial credit for being Done and carried over to the next Sprint, then the progress of the subsequent Sprint could be disrupted. Therefore, the Done status is black and white. A User Story can only be either Done or not Done.

5.3.2 Definition of Ready

The Definition of Ready is a set of rules or criteria applicable to each User Story in the Prioritized Product Backlog. A User Story must satisfy the Definition of Ready before being considered for estimation and inclusion into a Sprint. The Definition of Ready puts the onus on the Product Owner to properly define requirements for each User Story. Without properly defined requirements, it will be impossible to get reliable estimates, and the Scrum Team may not be able to effectively complete the required project work.

The Scrum Guidance Body should preferably define the Definition of Ready criteria. However, there can also be project- or organizational-specific Definition of Ready criteria that will need to be added or updated. There may also be additions or updates to the Definition of Ready from the Scrum Team.

The Scrum Team commits to work on those User Stories which satisfy the Definition of Ready criteria. Review of product backlog items against the Definition of Ready criteria is a continuous activity that takes place as part of the *Refine Prioritized Product Backlog* process.

Some of the Definition of Ready criteria are:

- User Stories are written in enough detail so that they are understood by the Scrum Team and can be used for estimating

- User Stories have well-defined Acceptance Criteria
- Any related documentation that provides clarity about the User Stories are included
- User Stories are broken down to be small enough to be completed in a single Sprint.

Figure 5-2 is an interface from Vabro, showing a sample Definition of Ready (DoR) with criteria ensuring backlog items are fully prepared for sprint planning.

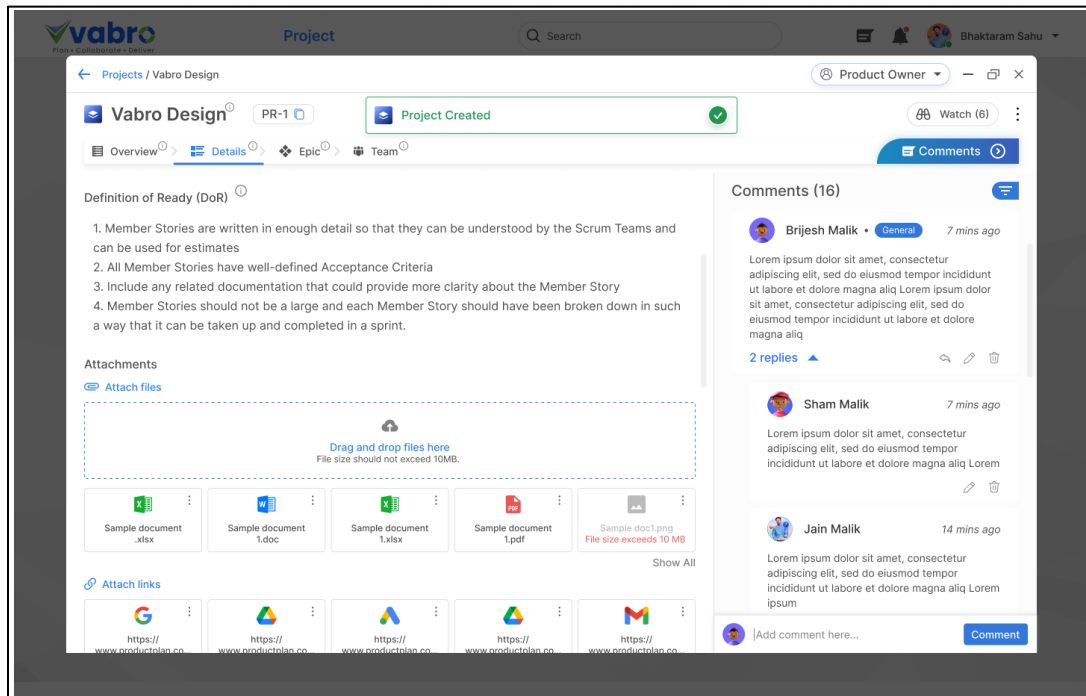


Figure 5-2: Sample DoR presented in an AI-powered Scrum Project Tool (Source: Vabro)

5.3.3 Definition of Done (or Done Criteria)

While Acceptance Criteria and the Definition of Ready are unique for individual User Stories, Done Criteria are a set of rules that are applicable to all the User Stories in a given Sprint. General Done Criteria could include any of the following:

- Reviewed by other team members
- Completed unit testing of the User Story
- Completion of quality assurance tests
- Completion of all documentation related to the User Story
- All issues are fixed
- Successful demonstration to business stakeholders and/or business representatives

Figure 5-3 is an interface from Vabro, showing a sample Definition of Done (DoD), including checklist items to ensure task completion, testing, and review.

The screenshot shows the Vabro Scrum Project Tool interface for configuring a sprint. The top navigation bar includes links for Home, IT, Goals, Report, and Projects. The main content area is titled 'Sprint43' and includes a 'Scrum Master' dropdown. The configuration form is divided into several sections:

- Team:** A dropdown menu set to 'Team 1'.
- Duration:** Two dropdown menus, one set to '1 Week' and the other to '4 Days'.
- Start Date:** A date picker set to 'Jun 13, 2023 12:00 AM'.
- End Date:** A date picker set to 'Jun 24, 2023 12:00 AM'.
- Name:** A text field set to 'Sprint 43'.
- Working Days:** A text field set to '9'.
- Definition of Done (DoD):** A text area containing a checklist:
 1. Reviewed by other team members
 2. Completed unit testing of the User Story
 3. Completion of quality assurance tests
 4. Completion of all documentation related to the User Story
 Below the list is a button labeled 'Generate with Vabro Genie'.
- Sprint Goal:** A text area containing placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliq...'. Below it is a button labeled 'Generate with Vabro Genie'.
- Default Scrumboard Columns:** A section with a 'Scrumboard Preview' window showing a Kanban board. It includes a checkbox for 'Add an Extra Column to' and a text field for 'Column Name' set to 'Testing'.
- Automatically renew Sprint:** A checkbox that is checked.
- Sprint Name Format:** A dropdown menu set to '{Start_Date}_{Team_Name}'.

At the bottom right, there are 'Cancel' and 'Save & Continue' buttons.

Figure 5-3: : Sample DoD presented in an AI-powered Scrum Project Tool (Source: Vabro)

As with the Acceptance Criteria, all conditions of the Done Criteria must be satisfied for each User Story to be considered Done. The Scrum Team should use a checklist of the general Done Criteria to ensure a task is finished and the result meets the Definition of Done (DoD). A clear Definition of Done is critical because it helps remove ambiguity and allows the team to adhere to required quality norms.

The Definition of Done (or the Done Criteria) is typically determined and documented by the Scrum Guidance Body. However, there can be project- or organization- specific Done Criteria that may need to be added or updated. There may also be additions or updates to the Done Criteria by the Scrum Team.

The required records and data to comply with the project's documentation requirements can be generated as the team proceeds through Sprints and Releases. The inclusion of activities such as holding review meetings and writing design documents can help ensure compliance with internal and external quality standards. The basic principles of Scrum such as short iterations, incremental building, customer involvement, adaptation to changing requirements, and constantly adjusting scope, time, and cost within the project will still apply.

5.3.4 Minimum Done Criteria

A higher-level business unit may announce mandatory minimum Done Criteria, which then become part of the Done Criteria for any User Story for that business unit. Any functionality defined by the business unit must satisfy these minimum Done Criteria if it is to be accepted by the respective Product Owner. The introduction of these Done Criteria may lead to a cascading set of Done Criteria for the portfolio, program, and project (see Table 5-1). Thus, the Done Criteria for a User Story in a project will implicitly include all the minimum Done Criteria from the higher levels, as applicable.

Portfolio Product Owner	• Sets the minimum Done Criteria for the entire portfolio • Reviews portfolio deliverables
Program Product Owner	• Sets the minimum Done Criteria for the entire program, which includes the Done Criteria from the portfolio • Reviews program deliverables
Chief Product Owner/ Product Owner	• Sets the minimum Done Criteria for the project, which includes the Done Criteria from the program • Reviews project deliverables

Table 5-1: Cascading Done Criteria

Once the minimum Done Criteria are defined, such criteria may then be documented in the Scrum Guidance Body documents and referred to by Scrum Teams as required.

5.3.5 Acceptance or Rejection of Prioritized Product Backlog Items

Toward the end of any iteration, the respective business unit and business stakeholders participate in a Sprint Review Meeting in which the product increment is demonstrated to the Product Owner, sponsor, customer, and users. While feedback from all the business stakeholders is gathered, only the Product Owner has the power to accept or reject a particular User Story as Done, according to the agreed upon Acceptance Criteria. Thus, the role of Acceptance Criteria in maintaining quality is critical and needs to be clearly understood by the team. It is the responsibility of the Scrum Master to ensure that the Acceptance Criteria for a User Story are not changed by the Product Owner in the middle of a Sprint. Partially completed User Stories are rejected as not Done and moved back into the Prioritized Product Backlog.

Figure 5-4 is an interface from Vabro, showing the user story approval process, detailing the steps for accepting or rejecting stories based on stakeholder validation.

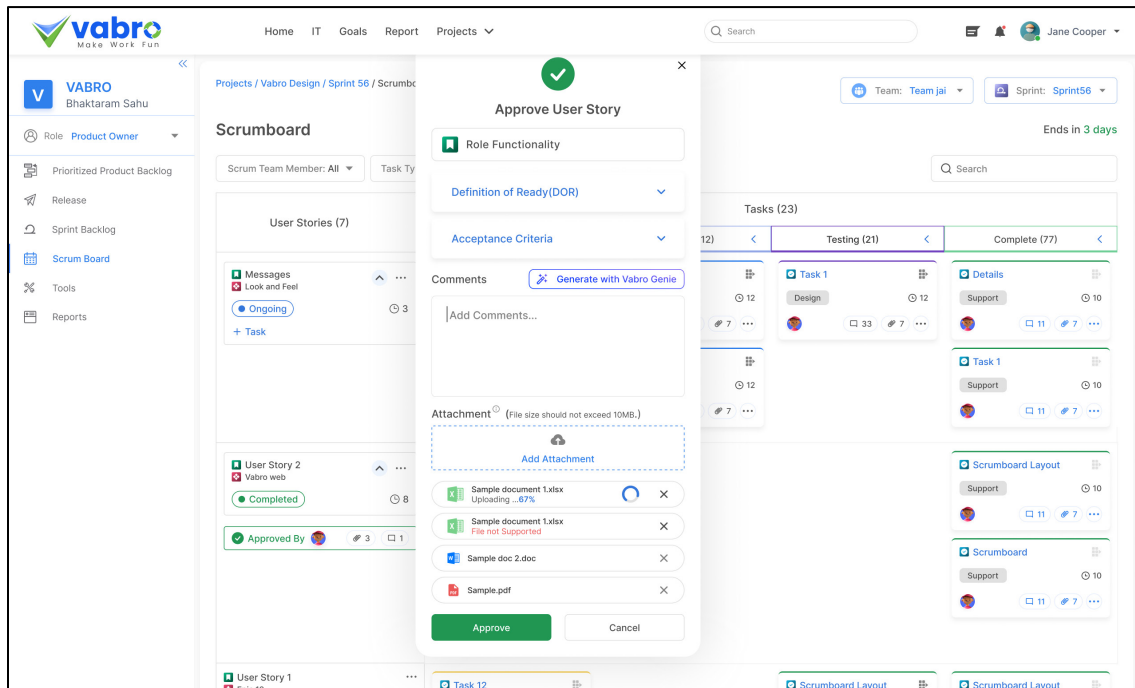


Figure 5-4: User Story Approval Process in an AI-powered Scrum Project Tool (Source: Vabro)

5.4 Quality Management in Scrum

The customer is the most important stakeholder in any project. Therefore, it is important to understand the customer's needs and requirements. The Voice of the Customer (VOC) can be referred to as the explicit and implicit requirements of the customer, which must be understood prior to the designing of a product or service. In a Scrum environment, the Product Owner's focus is on business requirements and objectives, which together represent the Voice of the Customer. The Product Owner can benefit from the guidance available from the Scrum Guidance Body (either through quality documents or standards, or from quality experts). These specialists should work with the Product Owner and the customer to ensure the appropriate level of detail and information in the User Stories, since User Stories are the basis for the success of any Scrum project.

It should be noted that external business stakeholders are not directly involved at the Scrum Team level and, instead, interact primarily with the Product Owner. For any Scrum project, the customer may be either of the following:

- Internal (that is, within the same organization)
- External (that is, outside the organization)

Quality management in Scrum enables customers to become aware of any problems in the project early and helps them recognize if a project is going to work for them or not. In Scrum, quality is about customer satisfaction and a working product, not necessarily meeting arbitrary metrics. This distinction becomes especially important from the customer's point of view because they are the ones investing time and money in the project.

Quality management in Scrum is facilitated through three interrelated activities:

1. Quality Planning
2. Quality Control
3. Quality Assurance

5.4.1 Quality Planning

One of the guiding principles of Scrum is to develop the functionality of the highest priority to the customer first. Less key features are developed in subsequent Sprints or can be left out altogether according to the customer's requirements. This approach gives the Scrum Team the required time to focus on the quality of essential functionality. A key benefit of quality planning is the reduction of technical debt. Technical debt—also referred to as design debt or code debt—refers to the work that teams prioritize lower, omit, or do not complete as they work toward creating the primary deliverables associated with the project's product. Technical debt accrues and must be paid in the future.

Some causes of technical debt can include the following:

- Quick-fix and building deliverables that do not comply with standards for quality, security, long-term architecture goals, etc.
- Inadequate or incomplete testing
- Improper or incomplete documentation

- Lack of coordination among different team members, or if different Scrum Teams start working in isolation, with less focus on final integration of components required to make a project or program successful
- Poor sharing of business knowledge and process knowledge among the business stakeholders and project teams
- Too much focus on short-term project goals instead of the long-term objectives of the organization. This oversight can result in poor-quality Working Deliverables that incur significant maintenance and upgrading costs.

In Scrum projects, any technical debt is not carried over beyond a Sprint, because there should be clearly defined Acceptance and Done Criteria. The functionality must satisfy these criteria to be considered Done. As the Prioritized Product Backlog is refined and User Stories are prioritized, the team creates Working Deliverables regularly, preventing the accumulation of significant technical debt. The Scrum Guidance Body may also include documentation and definition of processes which help in decreasing technical debt.

To maintain a minimal amount of technical debt, it is important to define the product required from a Sprint and the project along with the Acceptance Criteria, any development methods to be followed, and the key responsibilities of Scrum Team members in regard to quality. Defining Acceptance Criteria is an important part of quality planning, and it allows for effective quality control to be carried out during the project.

Technical debt is a very big challenge with some traditional project management techniques where development, testing, documentation, and so on are done sequentially and often-times by different people, with no one person being responsible for any particular Working Deliverable. As a result, technical debt accrues, leading to significantly higher maintenance, integration, and product release costs in the final stages of a project's release. Also, the cost of changes is extremely high in such circumstances as problems surface in later stages of the project. Scrum framework prevents the issues related to technical debt by ensuring that Done deliverables with Acceptance Criteria are defined as part of the Sprint Backlog and key tasks including development, testing, and documentation are done as part of the same Sprint and by the same Scrum Team.

5.4.1.1 Continuous Integration and Sustainable Pace

Maintaining a sustainable pace is an important tenet of Scrum. Sustainable pace translates to increased employee satisfaction, stability, and increased estimation accuracy, all of which leads to increased customer satisfaction. To develop a truly high-quality product and maintain a healthy work environment, it is important to carry out integration-type activities regularly, rather than delaying the integration work until the end of the project. To provide value at frequent intervals, the team should continuously develop, test, and integrate the functionalities of each Prioritized Product Backlog Item (PBI) in every Sprint with the use of techniques, such as continuous integration and automated product testing. It is also important, from the team's point of view, to ensure that the effort expended in the current Sprint is similar to the effort spent in the preceding Sprint in order to sustain an even pace throughout the project Sprints. This helps the team avoid phases of intense periods of work, ensuring they are always able to put forth the level of effort required to accomplish the work that needs to be done. Maintaining a sustainable pace is one of the most important tenets of Scrum and other Agile practices such as DevOps.

5.4.2 Quality Assurance and Quality Control

Quality is required not only in products, but also in processes. Quality assurance refers to the evaluation of processes and standards that govern quality management in a project to ensure that they continue to be relevant. Quality assurance activities are carried out as part of the work. In fact, quality assurance is a significant factor of the Definition of Done. The deliverable is not complete if appropriate quality assurance has not been conducted. Often, quality assurance is demonstrated during the Sprint Review Meeting.

Product Owners for respective projects, programs, and portfolios can monitor and evaluate quality assurance activities to ensure each team continues to agree and comply with the quality standards that have been set. End-to-end quality assurance may be addressed during final testing of the product, a Release, or a Sprint. A comparison of the number of issues encountered versus the number of User Stories completed can be done. The product components that have defects can be incorporated as Prioritized Product Backlog Items (PBIs), which can be worked upon by either the team or by one person at certain times during the Sprint, depending on the number of defects.

Quality control refers to the execution of the planned quality activities by the Scrum Team in the process of creating deliverables that are potentially shippable. It also includes learning from each set of completed activities in order to achieve continuous improvement. Within the cross-functional team, it is important to have the skills necessary to perform quality control activities. During the Sprint Retrospect Meeting, team members discuss lessons learned. These lessons act as inputs into continuous improvement and contribute to the improvement of ongoing quality control.

At times, the Scrum Guidance Body can define the processes and documents that can be referred to by Scrum Teams when doing their projects to ensure that uniform quality norms are followed by all projects within the organization.

5.4.3 Plan-Do-Check-Act (PDCA) Cycle

The Plan-Do-Check-Act Cycle—also known as the Deming or Shewhart Cycle—was developed by Dr. W. Edwards Deming, considered the father of modern quality control and Dr. Walter A. Shewhart. The following are some important points of Deming's philosophy:

- Management guidelines define quality. When management is able to provide a conducive environment and is able to motivate its employees to improve quality on a continuous basis, each employee will be able to contribute toward a superior quality product. Deming's "Theory of Profound Knowledge" advocates what management should do in order to create an environment in which each employee can make significant contributions to quality improvement.

Deming modified the Plan-Do-Check-Act to Plan-Do-Study-Act (PDSA) because he felt the term 'Study' emphasized analysis rather than simply inspection, as implied by the term 'Check.'

Both Scrum and the Deming/Shewhart/PDCA Cycle are iterative methods that focus on continuous improvement.

Figure 5-5 presents the PDCA Cycle in Scrum, linking Plan-Do-Check-Act stages with Scrum ceremonies like planning, review, and retrospective for continuous improvement.

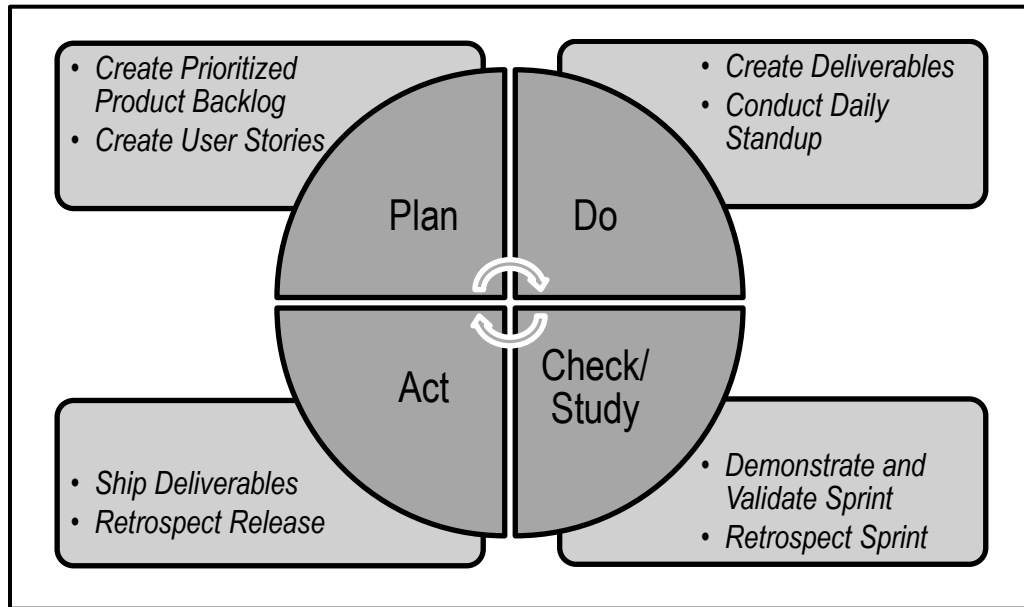


Figure 5-5: PDCA Cycle in Scrum

5.5 Summary of Responsibilities

Role	Responsibilities
Scrum Team	• Develops and maintains all deliverables during Sprints until they are handed over to end users • Practices and encourages effective communication so that requirements are clarified and fully understood • Shares knowledge to ensure that team members are familiar with the entire feature set and benefit from the experience of others • Makes appropriate changes to deliverables swiftly • Complies with the Definition of Done criteria for each deliverable
Product Owner/ Chief Product Owner	• States the business requirements for the product and defines requirements clearly in the Prioritized Product Backlog • Assesses viability and ensures that deliverables meet the quality requirements • Sets the minimum Done Criteria for the entire project, including the Acceptance Criteria for the respective projects • Facilitates creation of Acceptance Criteria for User Stories • Reviews and validates the deliverables during <i>Demonstrate and Validate Sprint</i>
Scrum Master/ Chief Scrum Master	• Facilitates a “team first” mentality when it comes to quality • Eliminates environmental obstructions that may affect the quality of deliverables and processes • Ensures that a sustainable pace is maintained in which the focus is on quality of features rather than strictly on velocity • Ensures that Scrum processes are correctly followed by all team members, including the Product Owner
Program Product Owner	• Sets the minimum Done Criteria for the entire program • Reviews program deliverables
Program Scrum Master	• Ensures that a sustainable pace is maintained in which the focus is on quality of features rather than strictly on velocity
Portfolio Product Owner	• Sets minimum Done Criteria for the entire portfolio • Reviews portfolio deliverables
Portfolio Scrum Master	• Ensures that a sustainable pace is maintained in which the focus is on quality of features rather than strictly on velocity
Business Stakeholder(s)	• Review and provide feedback about product deliverables • Work collaboratively with the Product Owner and the Scrum Team
Scrum Guidance Body	• Provides the Definition of Ready and the Definition of Done • Provides the framework and guidance for developing the Acceptance Criteria • Defines the range of tools that can be used by the Scrum Team to develop and verify the product components

Table 5-2: Summary of Responsibilities Relevant to Quality

5.6 Scrum vs. Traditional Project Management

Although there are similarities in Scrum and traditional project management methods with regard to definition of 'quality' (i.e., the ability of the product to meet the agreed Acceptance Criteria and achieve the business value expected by the customer), differences exist in terms of how the approaches address the implementation and achievement of the required quality levels.

In traditional project management methods, the users clarify their expectations; the project manager defines those expectations in measurable terms and gains agreement from the users. After detailed planning, the project team develops the product over an agreed period of time. If any of the agreed criteria are to be changed, changes can happen only through a formal change management system where impact of changes is estimated and the Project Manager gets approval from all relevant business stakeholders.

In Scrum, however, the Product Owner collaborates with the Scrum Team and defines the Acceptance Criteria for the User Stories related to the product to be delivered. The Scrum Team then develops the product in a series of short iterations called Sprints. The Product Owner can make changes to the requirements to keep pace with the user needs and these changes can be addressed by the Scrum Team either by terminating the current Sprint or including the adjusted requirements in the next Sprint as each Sprint is of short duration (i.e., one to four weeks).

One of the major advantages of Scrum is the emphasis on creating potentially shippable deliverables at the end of each Sprint cycle, instead of at the end of the entire project. So, the Product Owner and customers constantly inspect, approve, and accept deliverables after each Sprint. Also, even if a Scrum project is terminated early, there is some value created prior to termination through the deliverables created in individual Sprints.

This Page is Intentionally Left Blank.

6. CHANGE

6.1 Introduction

Every project, regardless of the method or framework used to deliver it, is exposed to change. It is imperative that project team members understand that the Scrum development processes are designed to embrace change. Organizations should try to maximize the benefits that arise from change and minimize any negative impacts through diligent change management processes in accordance with the principles of Scrum.

Change, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

6

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

This chapter is divided into the following sections:

6.1.1 Roles Guide—This section provides guidance on which sections are relevant for each of the primary Scrum roles: Product Owner, Scrum Master, and Scrum Team.

6.2 Overview—This section defines the concept of change, specifically within the context of Scrum processes. It also addresses how Change Requests are handled in Scrum processes.

6.3 Change in Scrum—This section details the importance of effectively managing change in a Scrum project. It also addresses how flexibility and stability can be achieved through appropriate handling of the Change Requests that arise throughout a project.

6.4 Integrating Change—This section details how Change Requests are assessed and approved (or rejected) when applying the Scrum framework.

6.5 Change to Programs and Portfolios—This section describes the impact of changes to programs and portfolios.

6.6 Summary of Responsibilities—This section defines the change management responsibilities of project team members.

6.7 Scrum vs. Traditional Project Management—This section discusses the benefits of managing change using Scrum methods over the methods used in traditional project management models.

6.1.1 Roles Guide

1. Product Owner—The responsibility of initiating change in a project lies primarily with the Product Owner; therefore, this entire chapter is applicable to this role.
2. Scrum Master—The Scrum Master should also be familiar with this entire chapter with primary focus on sections 6.3, 6.4, 6.5, and 6.7.
3. Scrum Team—The Scrum Team should focus on sections 6.3, 6.4.2, 6.5, and 6.7.

6.2 Overview

Change is inevitable in all projects. In today's hypercompetitive world where technology, market conditions, and business patterns are continuously shifting, change is the only constant.

A primary principle of Scrum is its acknowledgement that a) business stakeholders (e.g., customers, users, and sponsors) do change their minds about what they want and need throughout a project (sometimes referred to as 'requirements churn') and b) that it is very difficult, if not impossible, for business stakeholders to define all requirements during project initiation.

Scrum development projects welcome change by using small development cycles that incorporate customer feedback on the project's deliverables after each Sprint. This enables the customer to regularly interact with the Scrum Team members, view product increments as they are ready, and change requirements earlier on in the development cycle. Also, the portfolio or program management teams can respond to Change Requests pertaining to Scrum projects applicable at their level.

Scrum embodies a key principle from the Agile Manifesto (Fowler and Highsmith, 2001): "Responding to change over following a plan." Scrum is practiced on the basis of embracing change and turning it into a competitive advantage. Therefore, it is more important to be flexible than to follow a strict, predefined plan. This means it is essential to approach project management in an adaptive manner that enables change throughout rapid product development or service development cycles.

Being adaptive to change is a key advantage of the Scrum framework. Although Scrum works well for all projects in all industries, it can be very effective when the product or other project requirements are not fully understood or cannot be well defined up front, when the product's market is volatile, and/or when the focus is on making the team flexible enough to incorporate changing requirements. Scrum is especially useful for complex projects with a lot of uncertainty. Long-term planning and forecasting is typically ineffective for such projects, and they involve high quantities of risk. Scrum guides the team through transparency, inspection, and adaptation to the most valuable business outcomes.

6.2.1 Unapproved and Approved Change Requests

Requests for changes are usually submitted as Change Requests. Change Requests remain unapproved until they get formally approved. The Scrum Guidance Body usually defines a process for approving and managing changes throughout the organization. In the absence of a formal process, it is recommended that minor changes that do not have significant impact on the project be directly approved by the Product Owner. The tolerance for such minor changes could be defined at an organizational level or by the sponsor for a particular project. In most projects, 90% of Change Requests could be classified as minor changes that should be approved by the Product Owner. So, the Product Owner plays a key role in managing changes in a Scrum project. Changes that are beyond the approval level of the Product Owner may need approval from relevant business stakeholders working with the Product Owner.

At times, if a requested change could have a substantial impact on the project or organization, approval from senior management (e.g., Executive Sponsor, Chief Product Owner, Program Product Owner, and/or Portfolio Product Owner) may be required. Change Requests for the project are discussed and approved during the *Develop Epic(s)*, *Create Prioritized Product Backlog*, and *Refine Prioritized Product Backlog* processes. *Approved Change Requests* are then prioritized along with other product requirements and their respective User Stories and then incorporated into the Prioritized Product Backlog.

Figure 6-1 illustrates a sample change approval process, outlining steps for evaluating, reviewing, and approving change requests within Scrum projects.

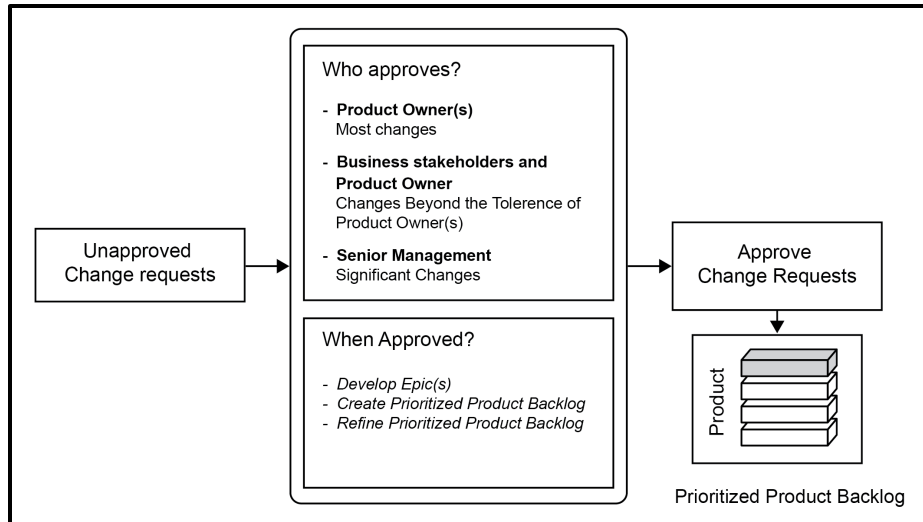


Figure 6-1: Sample Change Approval Process

Figure 6-2 shows how approved changes are updated in the Prioritized Product Backlog, ensuring real-time backlog refinement and transparency.

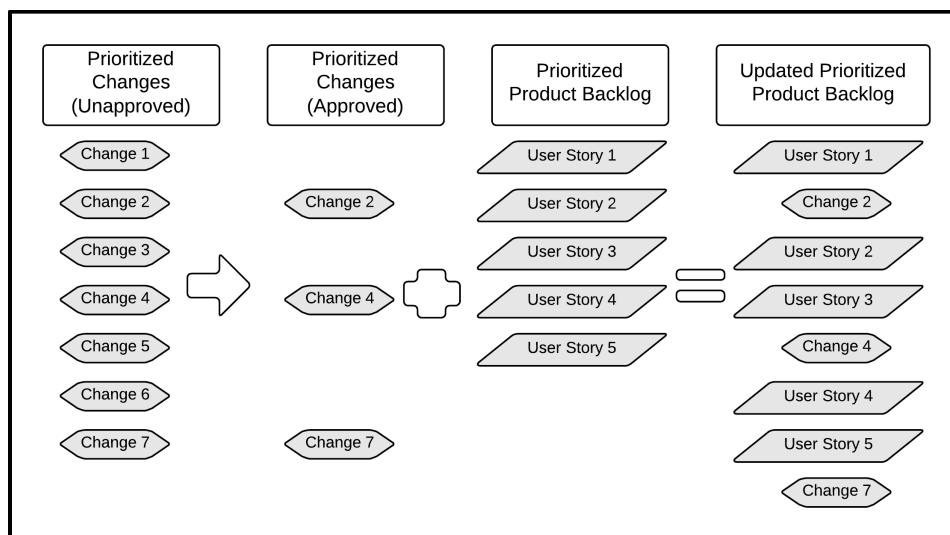


Figure 6-2: Updating Prioritized Product Backlog with Approved Changes

6.3 Understanding Change in Scrum

6.3.1 Balancing Flexibility and Stability

Scrum helps organizations become more flexible and open to change. However, it is important to understand that although the Scrum framework emphasizes flexibility, it is also important to maintain stability throughout the change process. In the same way that extreme rigidity is ineffective, extreme flexibility is also unproductive. The key is to find the right balance between flexibility and stability because stability is needed in order to get work done. Therefore, Scrum uses iterative delivery and its other characteristics and principles to achieve this balance. Scrum maintains flexibility in that Change Requests can be created and approved at any time during the project; however, they get prioritized when the Prioritized Product Backlog is created or updated. At the same time, Scrum ensures that stability is maintained by keeping the Sprint Backlog fixed and by not allowing interference with the Scrum Team during a Sprint. In Scrum, all requirements related to an ongoing Sprint are frozen during the Sprint. No change is introduced until the Sprint ends, unless a change is deemed to be significant enough to stop the Sprint. In the case of an urgent change, the Sprint is terminated, and the team meets to plan a new Sprint. This is how Scrum accepts changes without creating the problem of changing release dates.

6

6.3.2 Incorporating Flexibility

Because of its iterative nature and the empirical process control concepts of transparency, inspection, and adaptation, Scrum implementation must incorporate flexibility. Scrum provides an adaptive mechanism for project management in which a change in requirements can be accommodated without significantly impacting overall project progress. It therefore becomes possible (and necessary) to adapt to emerging business realities as part of the development cycle. Flexibility in Scrum is achieved through five key characteristics—iterative product development, Time-boxing, cross-functional teams, customer value-based prioritization, and continuous integration (see Figure 6-3).

Figure 6-3 highlights Scrum characteristics for achieving flexibility, including iterative delivery, continuous feedback, and adaptive planning.

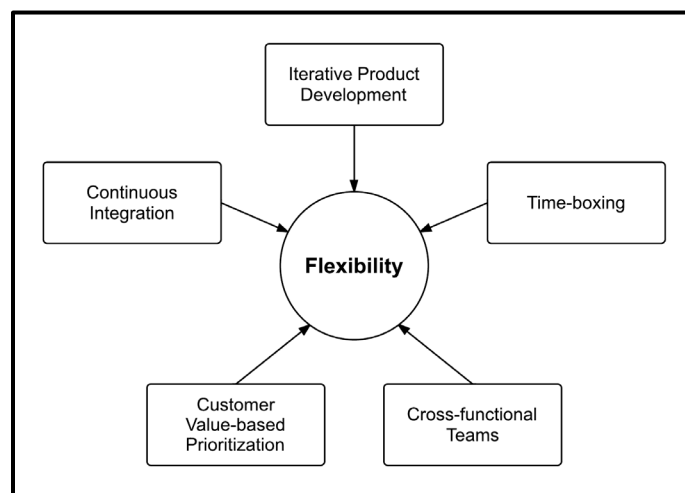


Figure 6-3: Scrum Characteristics for Achieving Flexibility

6.3.2.1 Flexibility through Iterative Product Development

Scrum follows an iterative and incremental approach to product and service development, making it possible to incorporate change at any step in the development process. As the product is developed, a Change Request for the project can come from multiple sources as follows:

1. Business stakeholders

Business stakeholders—particularly sponsors, customers, and users—may submit Change Requests at any time throughout the project. Change Requests could be due to changes in market conditions, organizational direction, legal or regulatory issues, or various other reasons. Moreover, business stakeholders may submit Change Requests as they are reviewing the deliverables during the *Demonstrate and Validate Sprint*, *Retrospect Sprint*, or *Retrospect Release* processes. All Change Requests get added to the Project Prioritized Product Backlog (also referred to as Prioritized Product Backlog or Product Backlog) once approved.

Figure 6-4 shows the motivation of business stakeholders for requesting changes, including evolving market demands, customer feedback, and compliance needs.

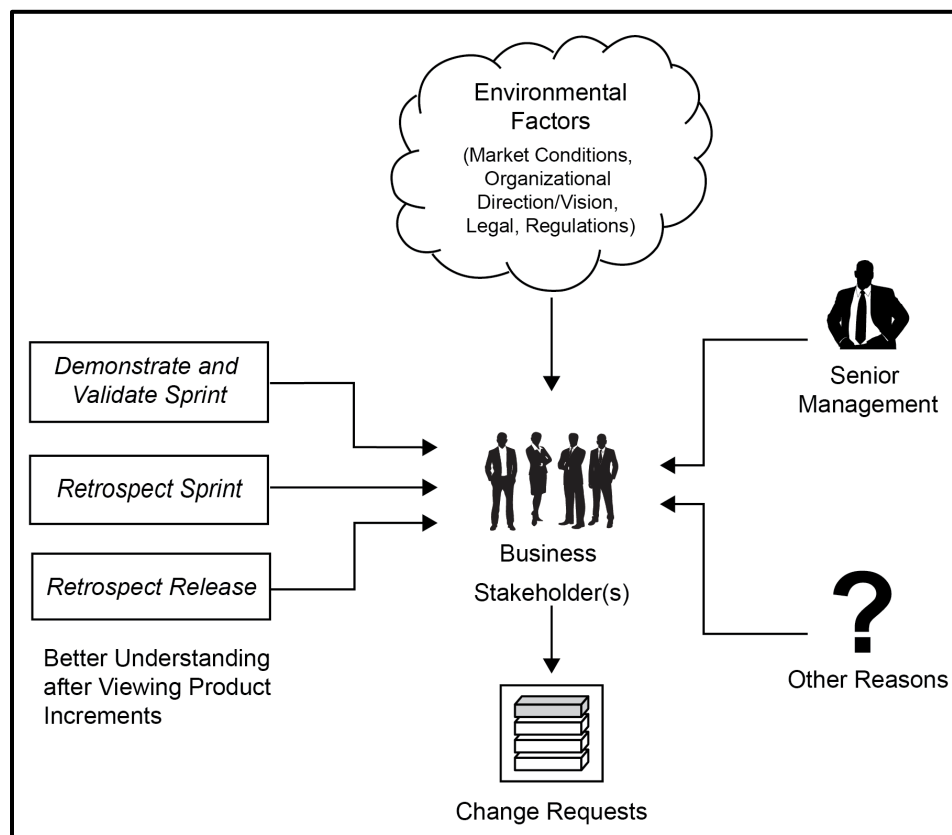


Figure 6-4: Motivation of Business Stakeholders for Requesting Changes

2. Scrum Core Team

The Scrum Core Team (i.e., the Product Owner, Scrum Master, and Scrum Team) are involved in creating the product deliverables. Ongoing interaction between the Scrum Core Team members in a Scrum Team and others, such as other Scrum Teams in the project, and internal and external project business stakeholders, may motivate Scrum Core Team members to suggest changes or improvements to the product, service, or some other part of the project. Usually such changes—like any others—are captured in Change Requests, and the Product Owner makes a final decision about which suggested changes from the Scrum Team and Scrum Master should be considered as formal Change Requests.

There may at times be challenges with creating certain deliverables, which may result in Change Requests. For example, the team may decide on a new feature to be added or modified to improve product performance. In most Scrum projects, recommendations for changes from the Scrum Core Team happen as Scrum Teams work to *Create Deliverables*, or when they participate in meetings associated with the *Conduct Daily Standup* or *Retrospect Sprint* processes.

Figure 6-5 displays the motivation of the Scrum Core Team for requesting changes, focused on technical improvements, team learning, and value delivery optimization.

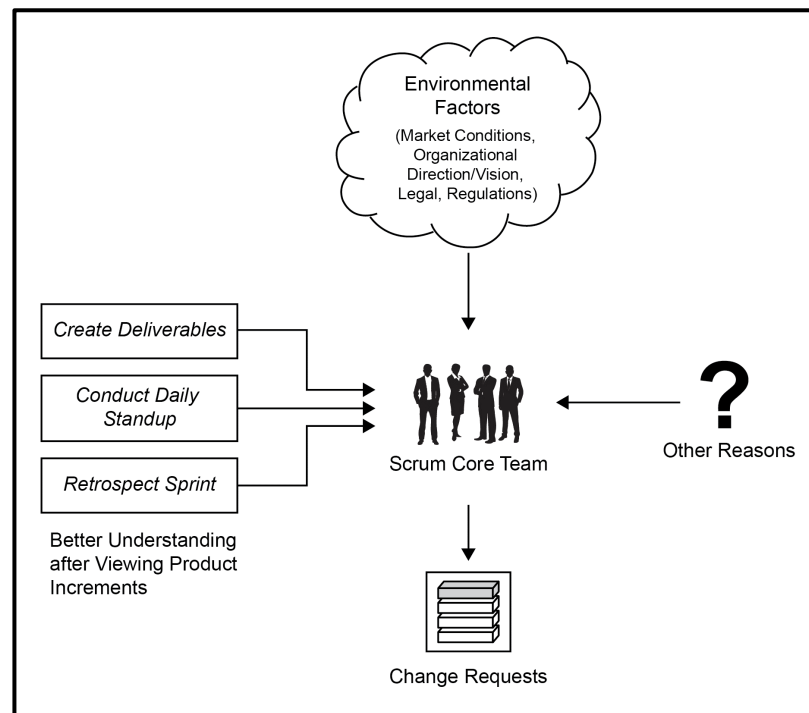


Figure 6-5: Motivation of Scrum Core Team for Requesting Changes

3. Senior Management

Senior management—including portfolio and program management—can recommend changes that affect the project. This can be because of changes in the strategic direction of the organization, competitive landscape, funding-related issues, and so forth.

Note that such changes get added to the Prioritized Product Backlog and need to go through the usual change management process. If any of these changes are urgent, any impacted Sprint may need to be terminated.

4. Scrum Guidance Body

The Scrum Guidance Body may submit Change Requests that affect all projects due to any of the following examples:

- Change in government regulations (e.g., privacy, security standards, or new legislation)
- Corporate directives for quality, security, or other organizational initiatives that need to be implemented across the organization
- Benchmarks or best practices to meet a certain standard
- Lessons learned from previous projects, which could be implemented by other Scrum Teams

The hallmark of Scrum is that it is change tolerant and adaptive. Scrum does not promote determining and firmly setting plans way in advance because it operates on the premise that project development is extremely prone to change and risk. The result is a high degree of flexibility and tolerance for change. The project is planned, executed, and managed incrementally, so it is typically easy to incorporate changes throughout.

6.3.2.2 Flexibility through Time-boxing

Time-boxing refers to setting short periods of time for work to be done. If the work undertaken remains incomplete at the end of the Time-box, it is moved into a subsequent Time-box. Examples of Time-boxing include limiting the Daily Standup Meetings to 15 minutes and setting Sprint durations to be one to four weeks. Time-boxed Sprints contribute toward meeting deadlines and achieving elevated levels of productivity. Errors or problems can be identified early and rectified quickly, enabling flexibility in Scrum projects. Sprints promote order and consistency in a volatile work environment. They provide a platform to gauge results and obtain feedback in a short span of time. Sprints also allow for frequent assessment of progress and the methods used to manage the project, including effective change management.

Furthermore, by incorporating Time-boxing in Sprints, the team frequently revisits the process of estimating the work to be done, so the projection of time and effort required becomes more accurate with each subsequent Sprint as the project progresses. These iterative cycles also motivate team members to achieve the projected targets and incremental goals toward reaching the larger project objectives.

6.3.2.3 Flexibility through Cross-functional and Self-organized Teams

Self-organization ensures that Scrum Team members have the flexibility to determine all the tasks they will work on in a Sprint and how they will complete the work. It also keeps teams self-motivated to complete their self-assigned tasks, removes bottlenecks, and encourages the sharing of knowledge with other team members.

The cross-functional and self-organized structures of the Scrum Team allow team members to be extremely focused on the desired Sprint results. The team has a defined set of objectives during each Sprint and the flexibility to account for a change in objectives prior to beginning the next Sprint.

The use of cross-functional teams also ensures that all of the skills and knowledge required to carry out the work of the project exists within the team itself. This provides an efficient working model that result in the creation of deliverables that are potentially shippable and ready for demonstration to the Product Owner and/or other business stakeholders.

Self-organization ensures that Scrum Team members determine on their own *how* to do the work of the project without a senior manager micromanaging their tasks.

Having cross-functional and self-organized teams allows the group to adapt and effectively manage the ongoing work and any minor issues or changes without having to obtain support or expertise from members outside the team, while in the process of creating deliverables.

6.3.2.4 Flexibility through Customer Value-based Prioritization

The prioritization of requirements and work in a Scrum project is always determined based on the value provided to the customer. First, at the start of a project, the initial requirements are prioritized based on the value each requirement will provide—this is documented in the Prioritized Product Backlog. Then, when a request is made for a new requirement or a change to an existing one, it is evaluated during the *Refine Prioritized Product Backlog* process. If the change is deemed to provide more value than other existing requirements, it will be added and prioritized accordingly in the updated Prioritized Product Backlog. So, the Prioritized Product Backlog provides scope for incorporating changes and adding new requirements when necessary.

It is important to note that new requirements and changes added to the Prioritized Product Backlog may lower the priority of other existing User Stories in the Backlog: so, such lower prioritized User Stories may be implemented later depending on their new prioritization. Because customers are very closely involved with the prioritization of requirements and their corresponding User Stories in the Prioritized Product Backlog, this practice ensures that the requirements that customers deem as “high value” get completed sooner and that the project starts delivering significant value much earlier on.

6.3.2.5 Flexibility through Continuous Integration

Using continuous integration techniques, Scrum Team members can incorporate new and modified features into the deliverables whenever possible. This mitigates the risk of multiple team members making changes to redundant components (e.g., obsolete code in software products, old designs for manufacturing parts). This ensures that only the latest feature or version is being worked on and avoids compatibility issues.

6.4 Integrating Change

Depending on the industry and type of project, the priority of features and requirements for a project may remain fixed for significant durations of time, or they may change frequently. If project requirements are generally stable, there are typically only minor changes made to the Prioritized Product Backlog throughout development, and Scrum Teams can work sequentially completing requirements that will provide maximum customer value as prioritized in the Prioritized Product Backlog. The length of the Sprint is usually longer, 4 to 6 weeks, in such stable environments. If project requirements change throughout the project, for example, due to changed business requirements, the same method continues to be effective. Before beginning a Sprint—during the *Create Prioritized Product Backlog* or *Refine Prioritized Product Backlog* processes—the highest priority requirements in the Prioritized Product Backlog are typically selected to be completed in that Sprint. Because changes have been accounted for in the Prioritized Product Backlog, the team only needs to determine how many tasks they can accomplish in the Sprint based on time and resources provided. Change management is executed in the ongoing processes of prioritizing and adding tasks to the Prioritized Product Backlog.

6.4.1 Changes to a Sprint

If there is a Change Request that may have significant impact on a Sprint in progress, the Product Owner, after consultation with relevant business stakeholders, decides whether the change can wait until the next Sprint or represents an urgent situation which may require ending the current Sprint and starting a new one. The Scrum framework clearly specifies that the scope of a Sprint cannot be changed once the Sprint begins. If the required change is so important that the results of the Sprint would be worthless without it, then the Sprint should be terminated. If not, then the change is incorporated into a later Sprint (as shown in Figure 6-6).

Figure 6-6 shows how a Product Owner manages change requests during a sprint in an Agile framework.

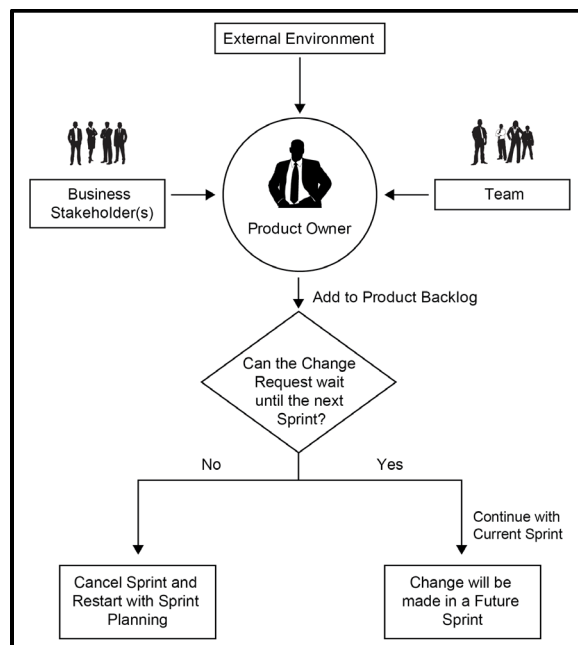


Figure 6-6: Incorporating a Change during a Sprint

There is only one exception to this rule about not changing the scope of a Sprint once a Sprint begins. If the Scrum Team determines it has heavily overestimated the effort during the Sprint and has spare capacity to implement additional User Stories, the team can ask the Product Owner which additional User Stories can be included in the current Sprint. By locking down the scope of every Sprint, the team is able to efficiently optimize and manage their work and effort. An additional benefit is that the team does not have to worry about managing changes once they start working on a Sprint. This is a big advantage of the Scrum framework as compared with traditional project management.

In traditional project management, changes can be requested and approved anytime during the project's lifecycle. This often creates confusion for project team members, decreases team motivation due to discontinuity, and results in a lack of focus and the team feeling that "nothing ever gets done." On the other hand, in Scrum projects, changes are not allowed once a Sprint starts. This ensures that in every Sprint, the team completes deliverables and tasks are Done. Furthermore, the business recognizes tangible benefits from potentially shippable deliverables at the end of each Sprint.

Moreover, since the Product Owner and business stakeholders are aware that changes are not allowed once a Sprint begins and a Sprint lasts between 1 and 4 weeks, they define and prioritize requirements during the appropriate processes of *Create Epic(s)*, *Create Prioritized Product Backlog*, and *Refine Prioritized Product Backlog*.

6

6.4.1.1 Impact of Expected Change on the Length of Sprint

Because changes are not allowed during a Sprint, the likelihood of any changes arising may have an impact on the decision related to the Length of Sprint when it is determined during the *Conduct Release Planning* process.

Figure 6-7 demonstrates an inverse relationship between sprint length and the probability of change during the sprint.

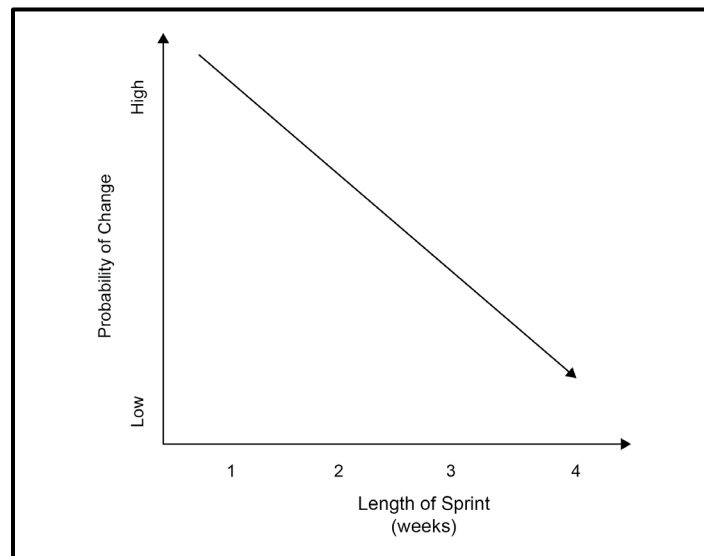


Figure 6-7: Impact of the Probability of Change on the Length of Sprint

A Sprint is Time-boxed with duration of one to four weeks. Most Scrum projects typically have Sprints Time-boxed at duration of two or three weeks. If project requirements are not very well defined, or if significant changes are expected in the immediate future, the Length of Sprint is typically set at one to three weeks. This provides enough stability to the Scrum Team members to work on shorter Sprints, and in turn, deliver results quickly (which are then evaluated by the Product Owner and business stakeholders at the end of each Sprint). This also provides enough flexibility for the team to clarify requirements and make changes to the Prioritized Product Backlog at the end of each Sprint.

However, if project requirements are stable and major changes are not expected in the near future, the Length of a Sprint may in some cases be set to extend up to six weeks. This provides more stability to the Scrum Team members to work on the Prioritized Product Backlog requirements for lengthy periods of time without having to prematurely go through the *Create User Stories*, *Estimate User Stories*, *Commit User Stories*, *Identify Tasks*, *Estimate Task*, and other related processes that are conducted for every Sprint. It is important to note that the likelihood of any change arising is not the only factor that should be used to determine the Length of Sprint. Other factors that also need to be considered include:

- Actual time to get the work done (if the project or corporate environment needs a specific time to get tasks done, this could determine the Length of Sprint)
- Planned date for a release (the Length of Sprint should take into consideration the release dates for the overall product or service)
- Any other factor as determined by the Product Owner or Scrum Master, that needs to be considered while determining the Length of Sprint

Changing the Length of Sprint should not be decided lightly or periodically. For example, it is not advisable to have the Sprint length as three weeks for the current Sprint, two weeks for the next Sprint, four weeks for the third Sprint, and so on. Length of Sprint should preferably be consistent. One of the greatest impacts of changing the Length of Sprint is that it causes a reset on all tracking at the project level. Previous velocities may become useless for forecasting and planning of future Sprints. Without an accurate velocity (which is a primary metric in any Scrum project), the Scrum Team cannot be measured for effectiveness or adequately choose the number of User Stories to take on when planning for a Sprint. So, once the Length of Sprint is decided, it should preferably be kept constant over the duration of the project or through multiple Sprint cycles.

6.4.1.2 Managing Changes through Prioritized Product Backlog Refining

A typical Prioritized Product Backlog will contain all User Stories, their time estimates (including any revised estimates), and the status of higher priority requirements. Any new or revised User Stories resulting from changes to business requirements, customer requests, external market conditions, and/or lessons learned from previous Sprints are also incorporated. One of the Product Owner's key responsibilities is refining the Prioritized Product Backlog to ensure the prioritized requirements in the Prioritized Product Backlog to be included in the next two to three Sprints are refined into suitable User Stories. It is recommended that the Product Owner should spend a significant amount of the time in each Sprint for Prioritized Product Backlog refining. The Product Owner is responsible for adding and revising Prioritized Product Backlog Items in response to any changes and is responsible for providing more detailed User Stories that will be used for the next Sprint.

Refining helps ensure that refining of requirements and their User Stories is done well in advance of the Sprint Planning Meeting so that the team has a well-analyzed and clearly defined set of stories that can be easily broken down into tasks and subsequently estimated. Based on lessons learned from the current Sprint, there may be changes to requirements, or there may be reprioritization that can be easily incorporated into subsequent Sprints. Refining supports and enhances the flexibility of the Scrum model by incorporating the latest business and technical insights into future Sprints.

A Product Backlog Review Meeting (also referred to as a Prioritized Product Backlog Refining Session) is a formal meeting during the *Refine Prioritized Product Backlog* process, which helps the Scrum Team review and gain consensus about the Prioritized Product Backlog. However, other than the Prioritized Product Backlog Review Meeting, Prioritized Product Backlog refining should happen throughout the project and can include situations in which the Product Owner writes new User Stories or reprioritizes User Stories in the existing Prioritized Product Backlog, Scrum Team members or business stakeholders give their suggestions about new User Stories to the Product Owner, and so forth. It is important to note that any item in the Prioritized Product Backlog is always open for re-estimation until the Sprint Backlog is finalized in the *Update Sprint Backlog* process. After that, changes can continue to be made until immediately prior to the Sprint Planning Meeting, if required.

6.4.1.2.1 Effective Product Backlog Review Meeting (or Prioritized Product Backlog Refining Session)

The Product Owner takes the lead in a Product Backlog Review Meeting which is conducted during the *Refine Prioritized Product Backlog* process. It is important that the Product Owner sets the objectives and ideally develop an agenda before the Product Backlog Review Meeting begins. Without the objectives, the session will be unstructured and may prove unproductive. It is also important to limit the number of business stakeholders participating in the meeting. Having too many participants tends to decrease the overall efficiency of the meeting. The Product Owner should invite only those business stakeholders whose feedback is required for the refining session. All Scrum Team members should be included because their input is valuable to the work being done and any issues encountered. If the refining session results in any reprioritization of or change in the Prioritized Product Backlog, it is important that the team agrees with those changes. An effective refining session should result in clearly defined Prioritized Product Backlog Items (PBIs) so that the Scrum Team clearly understands what the customer's requirements are. This also helps the team become familiar with all User Stories in case one or more of them needs to be included in a Sprint at short notice. Acceptance and Done Criteria may also be discussed during refining sessions. Scrum does not Time-box refining exercises. Prioritized Product Backlog refining is a continuous activity for the Product Owner.

6.4.1.3 Managing Changes During Demonstrate and Validate Sprint

Although the Product Owner has the final say on Prioritized Product Backlog Items and whether to accept or reject any User Stories (corresponding to Approved Change Requests) presented during the *Demonstrate and Validate Sprint* process, it is the Scrum Master's responsibility to ensure that the requirements and Acceptance Criteria are not altered during the Sprint Review Meeting for the User Stories completed in the current Sprint. This prevents the rejection of completed User Stories based on the fact that they do not meet newly changed requirements. If any requirements need to be changed, any corresponding PBI needs to be revised to accommodate the modified requirements in a future Sprint.

6.5 Change in Programs and Portfolios

Any change that arises in either programs or portfolios may have a cascading effect on all dependent projects and their corresponding Sprints. Therefore, it is advisable to minimize changes at these higher levels. If a change is required and all business stakeholders agree to make the change at these levels, the following should be kept in mind.

6.5.1 In Programs

1. It is not recommended to make changes in between two Program Backlog Meetings.
2. If the change is minor, the Program Product Owner should secure approval from the relevant business stakeholders (e.g., sponsor, customer, and end user) and the Portfolio Product Owner and then add the requirements to the Prioritized Program Backlog. Product Owners for the project will consider those requirements for inclusion in future Sprints.
3. If the change is major, the program efforts along with associated projects and Sprints need to stop, and a Prioritized Product Backlog Meeting should be conducted to determine next steps.
4. Prioritized Program Backlog Meetings (also referred to as Program Backlog Meetings) should preferably be conducted at two- to six-month intervals. The frequency and impact of changes to a program determine the time duration between two Program Backlog Meetings. If there are several changes expected to arise in the program, it is preferable to conduct Program Backlog Meetings at more regular intervals (e.g., two to three months); but if there are fewer changes expected and if requirements are stable, the duration between two Program Backlog Meetings could be increased (e.g., five to six months).

6.5.2 In Portfolios

1. It is not recommended to make changes in between two Portfolio Backlog Meetings.
2. If the change is minor, the Portfolio Product Owner should secure approval from the relevant business stakeholders (e.g., sponsor, customer, and end user) and then add the requirements to the Prioritized Portfolio Backlog. Product Owners of the program and project will consider those requirements for inclusion in future Sprints.
3. If the change is major, the portfolio efforts along with associated programs, projects, and Sprints need to stop, and a Portfolio Backlog Meeting should be conducted to determine next steps.
4. Prioritized Portfolio Backlog Meetings (also referred to as Portfolio Backlog Meetings) should be conducted at four- to twelve-month intervals. The frequency and impact of changes to a portfolio determine the time duration between two Portfolio Backlog Meetings. If there are several changes expected in the portfolio, it is preferable to conduct Portfolio Backlog Meetings at more regular intervals (e.g., four to six months); but if there are fewer changes expected and if requirements are stable, the duration between two Portfolio Backlog Meetings could be increased (e.g., nine to twelve months).

Figure 6-8 depicts how changes are incorporated in programs and portfolios. It outlines a structured approach to evaluating, approving, and integrating changes to ensure alignment with organizational goals.

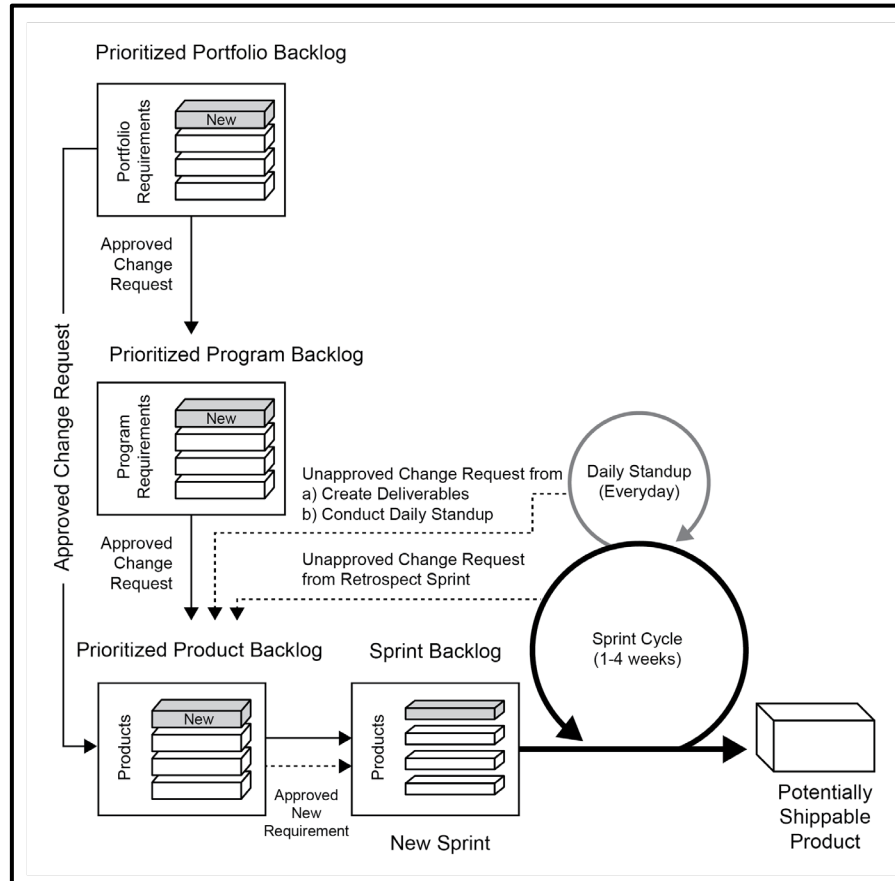


Figure 6-8: Incorporating Changes in Programs and Portfolios

6.6 Summary of Responsibilities

Role	Responsibilities
Scrum Team	Suggests improvements or changes during the <i>Create Deliverables</i> and <i>Conduct Daily Standup</i> processes
Product Owner/ Chief Product Owner	- Provides requests for changes in a project - Assesses the impact of requests for change raised for the portfolio, program, or project - Prioritizes User Stories in the project's Prioritized Product Backlog - Assesses the impact of problems on project objectives identified by the Scrum Team - Provides clear communication to business stakeholders on reprioritized Product Backlog Items
Scrum Master/ Chief Scrum Master	Facilitates identification, assessment, and escalation of problems and Change Requests by the Scrum Team
Program Product Owner	- Provides request for change for programs - Approves products that are amended, removed, or added according to program requirements
Program Scrum Master	Facilitates identification, assessment, and management of Change Requests for programs
Portfolio Product Owner	- Provides Change Requests for portfolios - Approves products that are amended, removed, or added according to portfolio requirements
Portfolio Scrum Master	Facilitates identification, assessment, and management of Change Requests for portfolios
Business Stakeholder(s)	- Provides request for changes - Involved with approving and prioritizing Change Requests
Scrum Guidance Body	Provides overall guidance for the change management procedures to be followed throughout the project

Table 6-1: Summary of Responsibilities Relevant to Change

6.7 Scrum vs. Traditional Project Management

Change management in traditionally managed projects is closely related to Configuration Management. All changes are considered based on their magnitude of variation from a baseline value. The Project Manager is given thresholds within which he or she can manage the day-to-day activities and decisions of the project. When a Change Request exceeds the defined tolerances, the Project Manager must escalate the proposed change to higher levels of management and await their decision before implementing the change. The Project Manager first logs the request for change in an Issue Log or Change Log and then escalates the change to higher authorities. These might include the sponsor of the project, as well as other relevant business stakeholders and decision makers. At some point, an impact assessment will be conducted. Based on the estimated impact of the change, a decision will be made regarding whether the change should be implemented or not. The Project Manager may also propose viable solutions to any problems posed by the change. If the higher authorities decide to proceed with making the change, the Project Manager is responsible for ensuring that the change is implemented correctly.

Change in Scrum works differently as compared with Traditional Project Management. The Scrum framework is highly tuned toward managing changes effectively and efficiently. Whenever the Product Owner or the Scrum Team recognizes a problem or defect or identifies a Prioritized Product Backlog Item that needs to be amended, replaced, or added, the change is made to the Prioritized Product Backlog. Similarly, senior management, the Product Owner, or business stakeholder(s) can add Change Requests to the Prioritized Product Backlog. The Product Owner and business stakeholder(s) approve Change Requests and reprioritize the Prioritized Product Backlog accordingly. Whenever there is a problem or new requirement that needs to be addressed immediately and mandates a change affecting the current Sprint, the Product Owner terminates the Sprint, with approval from relevant business stakeholders. Once terminated, the Sprint will be re-planned and restarted to incorporate the new requirements.

However, if the problem or change is not major and does not warrant a change within the current Sprint, the change will be added to the Prioritized Product Backlog and incorporated into the planning for a subsequent Sprint. This gives business stakeholders the ability to respond to changes in the external environment, while still maintaining a certain degree of control over the ongoing activities within the project. Also, at the end of each Sprint, completed deliverables are demonstrated by the Scrum Team. These deliverables are potentially shippable and can be reviewed by the Product Owner and other business stakeholders.

This Page is Intentionally Left Blank.

7. RISK

7.1 Introduction

The purpose of this chapter is to define risk, discuss the management of risks in a Scrum environment, and consider the tools that facilitate the management of risks. To ensure business viability, reduce the probability of project failure, and make more informed business decisions, it is important that risks are effectively managed through a well-organized and methodical approach.

In a Scrum environment, risks are minimized, due to the work being done in Sprints whereby a continuous series of deliverables is produced in short cycles, deliverables are compared to expectations, and the Product Owner is actively engaged in the project. However, even in the simplest of projects, things can go wrong, so it is important to have a strategy to identify and address risks.

Risk, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

This chapter is divided into the following sections:

7.1.1 Roles Guide—This section provides guidance on which sections are relevant for each Scrum role: Product Owner, Scrum Master, and Scrum Team.

7.2 What is Risk?—This section defines risk and explains how it can affect the objectives of a project and contribute to the success or failure of a project.

7.3 Risk Management Procedure—This section presents key techniques of risk management and elaborates on developing strategies to identify, assess, and manage risks.

7.4 Minimizing Risks through Scrum—This section explains the key aspects of Scrum that make it an ideal management framework for effectively handling risks at various levels—portfolio, program, and project.

7.5 Summary of Responsibilities—This section describes the responsibilities for each person or role on a project relative to risk management.

7.6 Scrum vs. Traditional Project Management—This section discusses the benefits of managing risk using Scrum methods over the methods used in traditional project management models.

7.1.1 Roles Guide

1. Product Owner—The major responsibilities of handling risks in a project lie with the Product Owner; therefore, the entire chapter is most applicable to this role.
2. Scrum Master—The Scrum Master should be familiar with this entire chapter with primary focus on sections 7.3, 7.4, and 7.7.
3. Scrum Team—The Scrum Team should focus primarily on sections 7.3 and 7.7.

7.2 What is Risk?

Risk is defined as an uncertain event that can affect the objectives of a project and may contribute to its success or failure. Risks with a potential for positive impact on the project are called opportunities, whereas threats are risks that could negatively impact a project. Managing risk must be done proactively, and it is an iterative process that should begin at project inception and continue throughout the life of the project. The process of managing risk should follow some standardized steps to ensure that risks are identified, evaluated, and a proper course of action is determined and acted upon accordingly.

Risks should be identified, assessed, and responded to on the basis of two factors primarily: the probability of an occurrence and the probable impact in the event of the occurrence. Risks with high probability and high impact rating should be addressed before those with a lower rating. In general, once a risk is identified, it is important to understand the basic aspects of the risk with regard to the probable causes, the area of uncertainty, and the potential effects if the risk occurs.

7.2.1 Difference between Risks and Issues

7

Risks are the uncertainties related to a project that could significantly alter the outcome of the project in a positive or negative way. Since risks are future uncertainties, they have no current impact on the project but could have a potential impact on the future. The following are some examples of risks:

- Even after extensive training, the customer service representatives might not be ready to take orders on the go-live date.
- The painting crew might be delayed due to heavy rain, which could negatively impact the project schedule.

Issues are well-defined certainties that are currently happening on the project—so there is no need to conduct a probability assessment as would be done for a risk. Issues must be dealt with. Some examples of issues include the following:

- Funding is not approved.
- Requirements are unclear.

Risks, if not addressed in time, may become issues. The goal of risk management is to be prepared, with plans in place to deal with any risks that may occur.

7.2.2 Risk Attitude

Business stakeholders include all people or organizations impacted by the project as well as those that have the ability to impact the project. It is important to understand the risk attitude of the business stakeholders. Risk attitude is influenced by the following three factors:

1. Risk appetite: refers to how much uncertainty the stakeholder or organization is willing to take on.
2. Risk tolerance: indicates the degree, amount, or volume of risk business stakeholders will withstand.
3. Risk threshold: refers to the level at which a risk is acceptable to the stakeholder organization. A risk will fall above or below the risk threshold. If it is below, then the stakeholder or organization is more likely to accept the risk.

The risk attitude of the business stakeholders determines how much risk the business stakeholders consider acceptable and hence when they decide to take action to mitigate potential adverse impacts of risks. Therefore, it is important to understand the tolerance levels of the business stakeholders in relation to several factors including cost, quality, scope, and schedule.

Utility Function is a model used for measuring stakeholder risk preference or attitude toward risk. It defines the business stakeholders' level or willingness to accept risk. The three categories of Utility Function are the following:

1. Risk averse: Stakeholder is unwilling to accept a risk no matter what the anticipated benefit or opportunity.
2. Risk neutral: Stakeholder is neither risk averse nor risk seeking and any given decision is not affected by the level of uncertainty of the outcomes. When two scenarios carry the same level of benefit, the risk neutral stakeholder will not be concerned if one scenario is riskier than the other.
3. Risk seeking: Stakeholder is willing to accept risk even if it delivers a marginal increase in return or benefit to the project.

7.3 Risk Management Procedure

Risk Management consists of the following five steps, which should be done iteratively throughout the project:

1. Risk identification: Using various techniques to identify all potential risks.
2. Risk assessment: Evaluating and estimating the identified risks.
3. Risk prioritization: Prioritizing risk to be included in the Prioritized Product Backlog.
4. Risk mitigation: Developing an appropriate strategy to deal with the risk.
5. Risk communication: Communicating the findings from the first four steps to the appropriate business stakeholders and determining their perception regarding the uncertain events.

7.3.1 Risk Identification

The Scrum Team members should attempt to identify all risks that could potentially impact the project. Only by looking at the project from different perspectives, using a variety of techniques, can they do this job thoroughly. Risk Identification is done throughout the project and Identified Risks become inputs to several Scrum processes including *Create Prioritized Product Backlog*, *Refine Prioritized Product Backlog*, and *Demonstrate and Validate Sprint*. The following techniques are commonly used to identify risks:

1. **Review Lessons Learned from Retrospect Sprint or Retrospect Release Processes**
Learning from similar projects and earlier Sprints in the same project and exploring the uncertainties that affected those projects and Sprints can be a useful way to identify risks.
2. **Risk Checklists**
Risk checklists can include key points to be considered when identifying risks, common risks encountered in the Scrum project, or even categories of risks that should be addressed by the team. Checklists are a valuable tool to help ensure comprehensive risk identification.
3. **Risk Prompt Lists**
Risk prompt lists are used in stimulating thoughts regarding the source from which risks may originate. Risk prompt lists for various industries and project types are available publicly.
4. **Brainstorming**
Sessions where relevant business stakeholders and members in the Scrum Core Team openly share ideas through discussions and knowledge sharing sessions, which are normally conducted by a facilitator.
5. **Risk Breakdown Structure (RBS)**
One of the key tools used in identifying risks is a risk breakdown structure. In this structure, risks are grouped based on their categories or commonalities. For example, risks may be categorized as financial, technical, or safety related. This allows the team to better plan for and address each risk.
6. **Interviews**
Interviews and informal meetings with business or other stakeholders such as senior management, Scrum Team members, technical experts, customers, and end users can help identify project risks.

7.3.2 Risk Assessment

The assessment of risk helps in understanding the potential impact of a risk, how likely it is to occur, and when the risk could materialize. The overall effect on business value should be estimated; if that impact is significant enough to outweigh the business justification, a decision must be made whether to continue the project.

The assessment of risks is done with regard to probability, proximity, and impact. Probability of risks refers to the likelihood of the risks occurring, whereas proximity refers to when the risk might occur. Impact refers to the probable effect of the risks on the project or the organization.

In addition to probability, risk assessment also evaluates the potential net effect of risks on the project or organization. These effects can be estimated using techniques such as Risk Models and Expected Monetary Value.

Risk Assessment Techniques:

1. Risk Meeting

Risks could be easily prioritized by the Product Owner by calling a meeting of the Scrum Core Team and optionally inviting relevant business stakeholders to the meeting. The team could meet and prioritize different risks based on their subjective assessment of the impact of the risks on project objectives.

2. Probability Trees

Potential events are represented in a tree with a branch extended for each outcome of a risk event. The probability of each outcome is indicated on the appropriate branch and then multiplied by its assessed impact to get an expected value for each outcome possibility. The outcome values are then summed together to calculate the overall expected impact of a risk to a project (see Figure 7-1).

Figure 7-1 explains a sample probability tree used in risk analysis. It presents outcomes and their likelihoods, supporting quantitative decision-making by visually mapping risk paths.

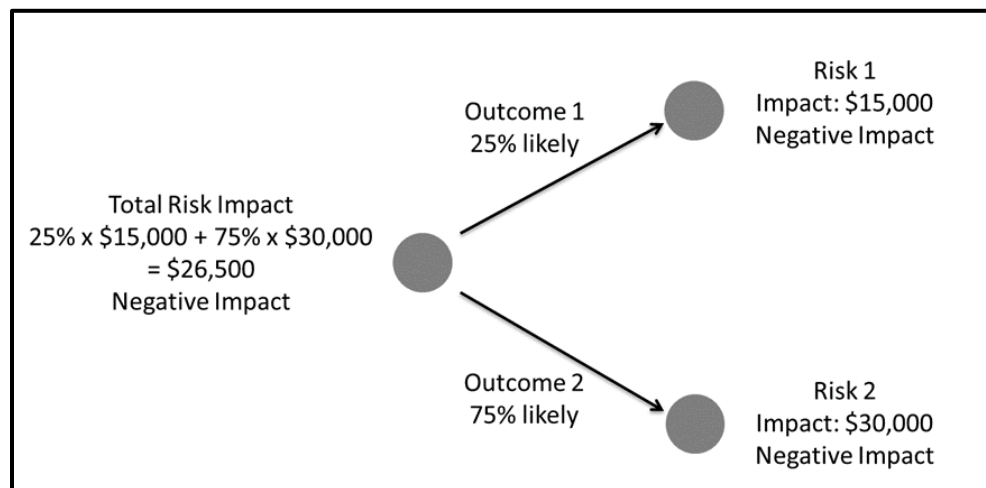


Figure 7-1: Sample Probability Tree

3. Pareto Analysis

This technique of assessing risk involves ranking risks by magnitude which helps the Scrum Team address the risks in the order of their potential impact on the project. For example, in Figure 7-2, Risk 1 has the highest impact and should preferably be addressed first.

Figure 7-2 shows a sample Pareto chart highlighting major contributors to problems. It helps teams prioritize issues based on frequency, aligning with the 80/20 rule in problem-solving.

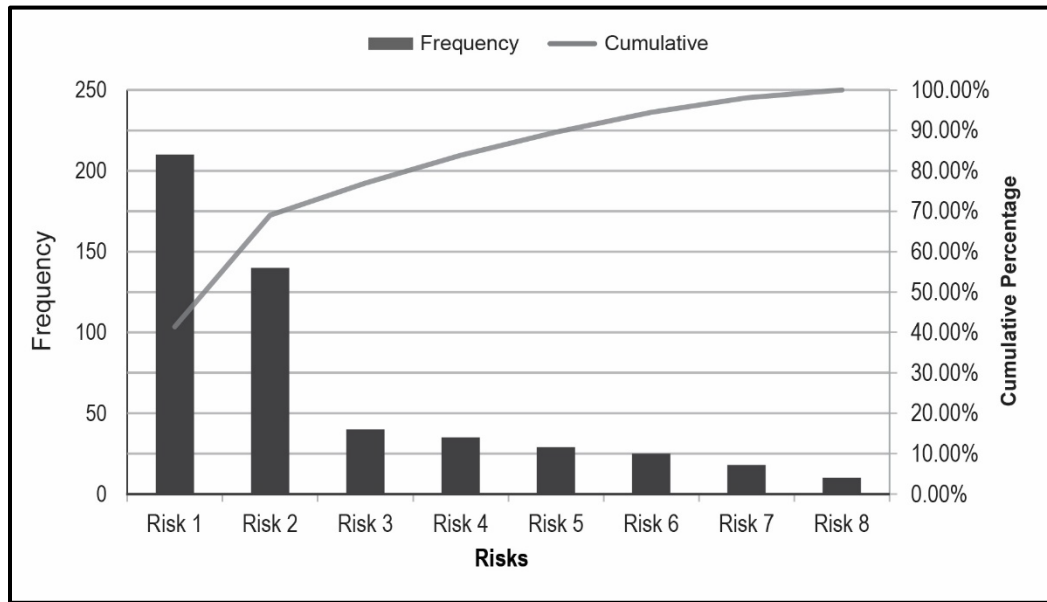


Figure 7-2: Sample Pareto Chart

4. Probability Impact Grid

Each risk is assessed for its probability of occurrence and for its potential impact on project objectives. A numerical rating is assigned to both probability and impact independently. The two values are then multiplied to derive a risk severity score (or PI value), which can be used to prioritize risks.

For example, the risk severity score for a risk with a probability of 50% and an impact rating of .6, where the impact is on a scale of 0 (low) to 1 (very high), would be calculated as follows:

$$0.5 (\text{Probability}) \times 0.6 (\text{Impact}) = 0.3$$

The rating schemes used are determined within the organization or for the project. Often a decimal scale is used, from zero to one, where a 0.5 probability rating would indicate 50% likelihood. Other options include a scale of one to ten, or High (3), Medium (2), and Low (1).

In the *Create Prioritized Product Backlog* or *Refine Prioritized Product Backlog* processes, the prioritized User Stories from the existing Prioritized Product Backlog and the prioritized list of risks are then combined to create an updated Prioritized Product Backlog which includes the Identified Risks:

Steps for updating a Prioritized Product Backlog with Identified Risks:

1. Create a list of prioritized risks. (e.g., the risks can be prioritized by value using Expected Monetary Value technique).
2. Select those Identified Risks that can be mitigated; and for which the team decides to take specific risk action during the Sprint to mitigate such risks.
3. Add the Identified Risks that can be mitigated in step 2 to the Prioritized Product Backlog (as User Stories) and then prioritize them to arrive at the Risk Adjusted Prioritized Product Backlog.

Figure 7-4 depicts the process for risk prioritization. It outlines steps for evaluating and ranking project risks to focus efforts on the most impactful threats and opportunities.

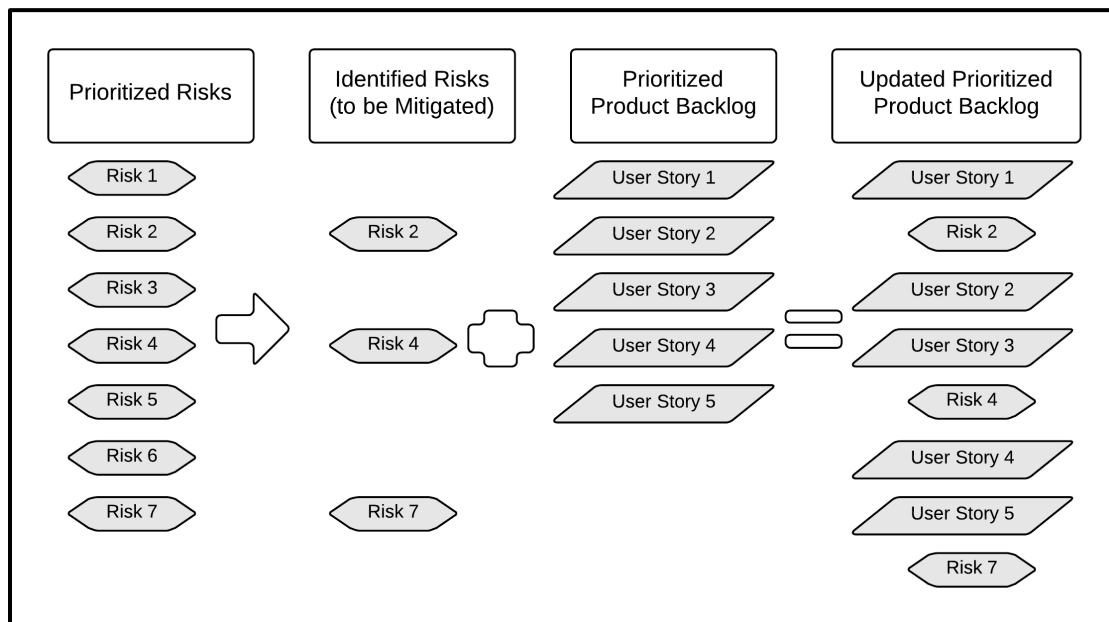


Figure 7-4: Process for Risk Prioritization

7.3.4 Risk Mitigation

The response to each risk will depend on the probability and impact of the risk. However, the iterative nature of Scrum with its rapid turnaround time and feedback cycles allows for early detection of failures; therefore, practically speaking, it has a natural mitigation feature built in.

Risk can be mitigated by implementing a number of responses. In most situations, responses are proactive or reactive. In the case of a risk, a plan B may be formulated, which can be used as a fall-back in case the risk materializes—such as plan B is a reactive response.

Sometimes risks are accepted and are an example of a risk response which is neither proactive nor reactive. Risks are accepted because of several reasons, as in a situation where the probability or impact of the risk is too low for a response.

Acceptance can also be the case in a situation where the apprehension of secondary risks may deter the product owner from taking any action. The effort made by the Product Owner to reduce the probability or impact—or both—of the risk is an example of a proactive response to mitigating risks.

Once Identified Risks are included as part of the Prioritized Product Backlog (see Figure 7-4), several risks get mitigated during the *Create Deliverables* process when the Tasks related to User Stories defined in the Prioritized Product Backlog process get completed.

In Scrum, the ownership of risk is clearly on the Product Owner for managing risks related to business aspects and on the Scrum Team for implementing risk responses during the course of a Sprint. The Scrum Guidance Body can be approached for advice on the way risk responses are implemented and whether the actions align with the guidelines of the organization as a whole. The Scrum Master keeps a close eye on the potential risks that could affect the project and keeps the Product Owner and Scrum Team informed.

Risk-based Spikes

A concept that can be useful in identifying risks is that of a risk-based spike. A spike is an experiment that involves research or prototyping to better understand potential risks. In a spike, an intense two to three-day exercise is conducted (preferably at the beginning of a project before the *Develop Epic(s)* or *Create Prioritized Product Backlog* processes) to help the team determine the uncertainties that could affect the project. Risk-based spikes are useful when the Scrum Team is working with and getting accustomed to modern technologies or tools, or when User Stories are lengthy. They also help in more accurately estimating time and effort.

Risk-based spikes can also be ongoing throughout the duration of the project and can be incorporated during any Sprint. Such spikes need to be added to the Prioritized Product Backlog. Risk-based spikes exploration are used to mitigate any future potential threats.

7.3.5 Risk Communication

Because business stakeholders have an interest in the project, it is important to communicate with them regarding risks. Information provided to business stakeholders related to risk should include potential impact and the plans for responding to each risk. This communication is ongoing and should occur in parallel with the four sequential steps discussed thus far—risk identification, assessment, prioritization, and mitigation. The Scrum Team may also discuss specific risks related to their Tasks with the Scrum Master during Daily Standup Meetings. The Product Owner is responsible for the prioritization of risks and for communicating the prioritized list to the Scrum Team.

A valuable tool which can be used for communicating information related to risks is the Risk Burndown Chart.

Risk Burndown Chart: Risk management is integral to ensuring value creation; therefore, risk management activities are performed throughout the project lifecycle and not just during project initiation. Each risk could be assessed using different Risk Assessment tools. However, the preferred tool for assessing risks to create a Risk Burndown Chart is Expected Monetary Value (EMV) as described in section 7.4.2.5.

The information gathered during risk assessment may be used to create a Risk Burndown Chart. This depicts cumulative project risk severity over time. The likelihoods of the various Risks are plotted on top of each other to show cumulative risk on the y-axis. The initial identification and evaluation of risks on the project and the creation of the Risk Burndown Chart are done initially. Then, at predetermined time intervals, new risks may be identified and assessed, and remaining risks should be re-evaluated and updated accordingly on the chart. An appropriate time to do this is during the Sprint Planning Meeting. Tracking risks in this manner allows the team to recognize trends in risk exposure and take appropriate action, as necessary.

Figure 7-5 shows a sample risk burndown chart that visualizes how identified risks decrease over time. It helps track the progress of risk mitigation efforts throughout a project.

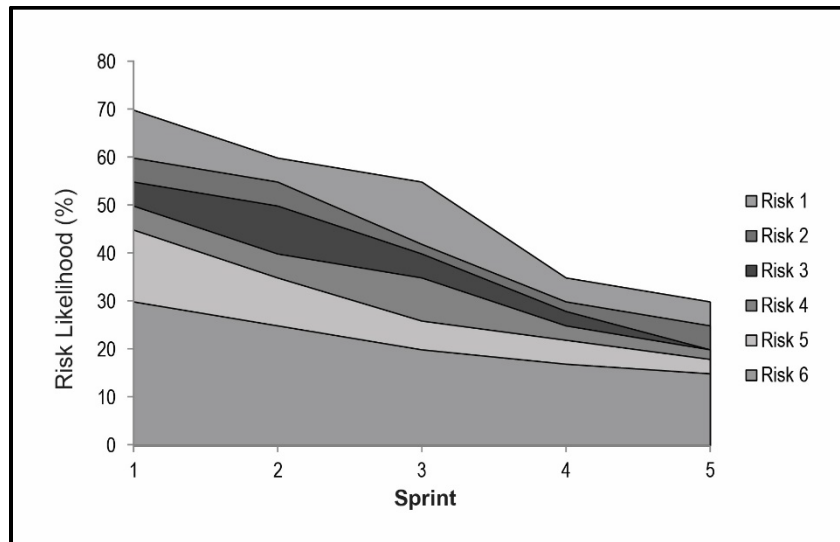


Figure 7-5: Sample Risk Burndown Chart

7.4 Minimizing Risks through Scrum

Being an Agile, iterative process, the Scrum framework inherently minimizes risk. The following Scrum practices facilitate the effective management of risk:

1. **Flexibility reduces business-environment-related risk**

Risk is minimized in Scrum due to the flexibility in adding or modifying requirements at any time in the project lifecycle. This enables the organization to respond to threats or opportunities from the business environment and unforeseen requirements whenever they arise, with low cost of managing such risks.

2. **Regular feedback reduces expectations-related risk**

Being iterative, the Scrum framework gives many opportunities to obtain feedback and set expectations throughout the project lifecycle. This ensures that the project business stakeholders, as well as the team, are not caught off guard by miscommunicated requirements.

3. **Team ownership reduces estimation risk**

The Scrum Team estimates and takes ownership of the Sprint Backlog Items, which leads to more accurate estimation and timely delivery of product increments

4. **Transparency reduces non-detection risk**

The Scrum principle of transparency around which the framework is built ensures that risks are detected and communicated early, leading to better risk handling and mitigation. Moreover, when conducting Scrum of Scrums Meetings, Impediments that one team is currently facing may be deemed a risk for other Scrum Teams in the future. This should be recognized in the Updated Impediment Log.

5. **Iterative delivery reduces investment risk**

Continuous delivery of value throughout the Scrum project lifecycle, as potentially shippable deliverables are created after every Sprint, reduces investment risk for the customer.

7.5 Risks in Portfolios and Programs

While few risks are specifically related to individual projects, others may originate in programs or portfolios and will be managed at the respective level. However, risks related to a portfolio or program will also impact projects that are part of the respective portfolio or program. During risk assessment in portfolios and programs, if it is determined that a risk may affect an individual project, relevant information about the risk must be communicated to the Product Owner and Scrum Team.

Depending on the severity or priority, when the program or portfolio team communicates a risk that will impact an individual project, the Scrum Team may have to stop and re-plan the current Sprint to address the risk. For less urgent risks, the team can continue the current Sprint and address the risk in a subsequent Sprint.

7.5.1 In Portfolios

1. When risks in Portfolio are identified, the Portfolio Product Owner will need to capture them and assess the proximity, probability, and impact of each identified risk in order to prioritize it and determine the appropriate response for the portfolio.
2. The Portfolio Product Owner will also need to communicate the risks to the relevant business stakeholders, the program teams, and the project teams. In some cases, the portfolio team may have to assume the ownership of specific risks.

7

7.5.2 In Programs

1. When program risks are identified, the Program Product Owner should enter them in the program Risk Adjusted Prioritized Product Backlog, assess the proximity, probability, and impact of each identified risk in order to prioritize it and determine the appropriate responses for programs.
2. The Program Product Owner will also need to communicate the risks to relevant business stakeholders and the project teams. In some cases, the program team would have to assume ownership of specific risks.

Figure 7-6 explains the approach to handling risks in portfolios and programs. It integrates governance structures and escalation mechanisms for risk tracking at higher organizational levels.

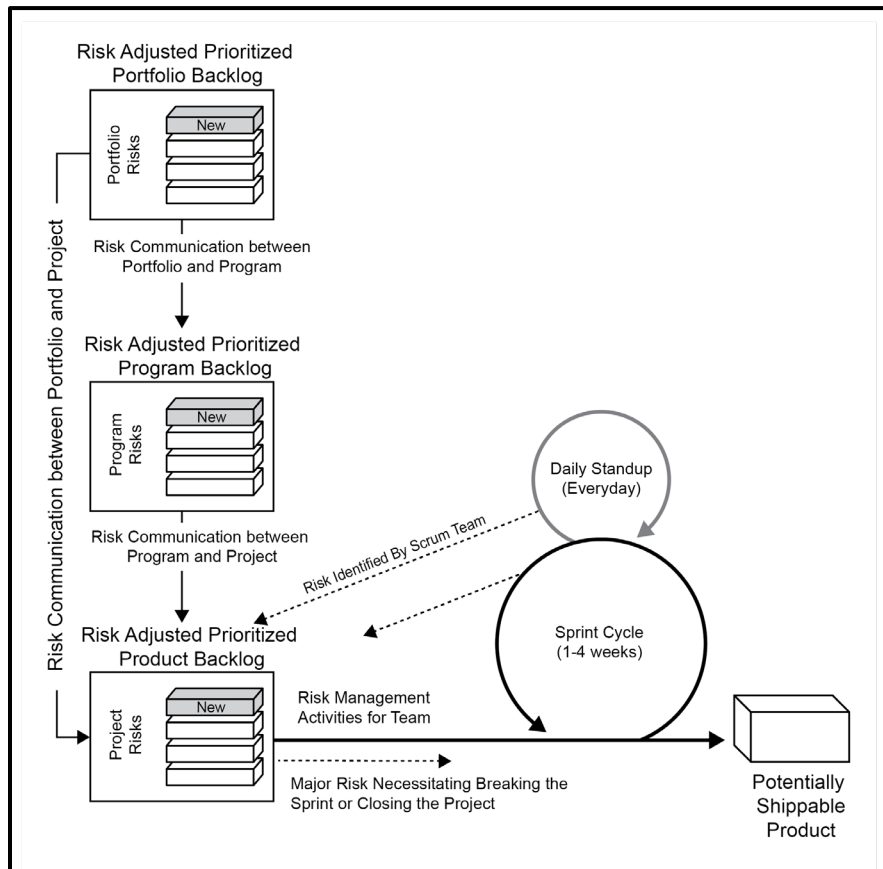


Figure 7-6: Handling Risks in Portfolios and Programs

7.6 Summary of Responsibilities

In Scrum, the risk management activities are divided among various roles with some responsibility resting with everyone in the Scrum Team and the Scrum Master facilitating the process.

Role	Responsibilities
Scrum Team	- Identifies risks during development of the product during the <i>Create Deliverables</i> process - Implements risk management activities as advised by the Product Owner
Product Owner/ Chief Product Owner	- Captures and assesses risks for project - Prioritizes and communicates risks to relevant business stakeholders, program, and portfolio teams - Ensures project risk levels are within acceptable limits
Scrum Master/ Chief Scrum Master	Facilitates identification and escalation of risks by the Scrum Team
Program Product Owner	- Captures and assesses risks for programs - Prioritizes and communicates risks to relevant business stakeholders and project teams
Program Scrum Master	Facilitates identification, assessment, and escalation of risks for programs
Portfolio Product Owner	- Captures and assesses risks for portfolios - Prioritizes and communicates risks to relevant business stakeholders, program, and project teams
Portfolio Scrum Master	Facilitates identification, assessment, and communication of risks portfolios
Business Stakeholder(s)	Interfaces with the Scrum Core Team to provide them inputs on management of risks that affect the achievement of expected outcomes and benefits from the project
Scrum Guidance Body	Provides overall guidance for the risk management procedure to be followed throughout the project

Table 7-1: Summary of Responsibilities Relevant to Risk

7.7 Scrum vs. Traditional Project Management

Scrum and most of the traditional project management methods define risk as ‘uncertain event(s) that could positively or negatively affect the achievement of project objectives.’ Also, risks are identified, assessed, planned for, and communicated continually.

In Traditional project management models, there is an emphasis on detailed upfront planning to identify, assess, and determine risk responses for all project risks. During project execution, any project team member can identify risks and the project manager, or the project management office or project support staff can update them in the Risk Log or Risk Register. The project manager regularly monitors and controls all risks and usually identifies specific individuals in the team to take responsibility for various aspects of risks.

In Scrum, any Scrum Team member can identify risks, and the Product Owner can update the identified risks in the Risk Adjusted Prioritized Product Backlog. The Scrum principles of Empirical Process Control and Iterative Development enable the Scrum Team to constantly keep identifying risks and adding them to the Prioritized Product Backlog, where such risks are prioritized with other existing User Stories, to be mitigated in subsequent Sprints. The Scrum Team has collective responsibilities for managing all risks for the Sprint.

8. INITIATE

This chapter includes the processes related to initiation of a project: *Create Project Vision*, *Identify Scrum Master and Business Stakeholder(s)*, *Form Scrum Team*, *Develop Epic(s)*, *Create Prioritized Product Backlog*, and *Conduct Release Planning*.

Initiate, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Product Owners, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter.

This chapter is written from the perspective of one Scrum Team working on one Sprint to produce potentially shippable deliverables, which could be part of a larger project, program, or portfolio. Additional information pertaining to Scaling Scrum for Large Projects is available in chapter 13. Additional information pertaining to Scaling Scrum for the Enterprise can be found in chapter 14.

The Initiate phase occurs at the beginning of a Scrum project. During this phase, the Scrum Core Team and business stakeholders are identified, starting with the Product Owner, who creates a Project Vision that serves as guidance throughout the entire project.

Based on Project Vision, an initial set of requirements is identified and documented in the form of Epics.

These initial requirements are prioritized and used to create an initial Prioritized Product Backlog (i.e., this is the requirements document in a Scrum project). During the last step of the Initiate phase, a Release Planning Schedule is created for the overall project. The Initiate phase does not produce a comprehensive and detailed plan for the entire project. There is no need for a comprehensive plan because change is expected and can easily be incorporated in a Scrum project due to the iterative principle incorporated into the Scrum processes. Rather, the goal of the Initiate phase of a Scrum project is to come up with a good initial plan for the project that aligns with the business needs and/or any high-priority regulations. This phase is expected to be short in order so that value creation can start as quickly as possible in the project.

It is also important to realize that although all phases and processes are defined uniquely in the SBOK® Guide, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to overlap some phases and/or processes, depending on the specific requirements of each project.

Figure 8-1 provides an overview of the Initiate phase processes, which are as follows:

8.1 Create Project Vision—In this process, the Product Owner is identified. Based on the project's business case, the Product Owner then creates a Project Vision Statement. This Project Vision Statement provides the overall guidance, inspiration, and focus for the project. The company may also choose to evaluate and select an AI-powered Scrum project tool that can significantly reduce the complexity of setting up the Scrum initiative and increase the productivity of Scrum projects within the organization.

8.2 Identify Scrum Master and Business Stakeholder(s)—In this process, the Scrum Master is identified using specific selection criteria that can effectively assess the soft skills and Scrum knowledge needed for this important role. Additionally, business stakeholders are also identified during this process.

8.3 Form Scrum Team—In this process, Scrum Team members are identified based on the skills required to complete the project deliverables, as well as considerations for the availability, costs, and soft skills important for members of a Scrum Team. Normally the Product Owner has the primary responsibility of selecting team members but often does so in collaboration with the Scrum Master.

8.4 Develop Epic(s)—In this process, the Project Vision Statement serves as the basis for developing Epics, which define the high-level requirements for the project. The Product Owner may use User Group meetings and other tools to collect requirements from business stakeholders.

8.5 Create Prioritized Product Backlog—In this process, Epics are refined, elaborated, and most importantly, prioritized according to their respective business value to create a Prioritized Product Backlog for the project. Additionally, based on the Scrum Guidance Body recommendations, the Product Owner and the Scrum Team establish the Done Criteria for the project.

8.6 Conduct Release Planning— In this process, the Product Owner, with inputs from business stakeholders and members of the Scrum Team, develops the initial Release Planning Schedule, which is communicated to, and shared with, all business stakeholders and Scrum Team Members. It is understood that the iterative nature of Scrum may necessitate future adjustments to the release schedule. The length of each Sprint is also determined in this process.



Figure 8-1: Initiate Overview

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 8-2 below shows the mandatory inputs, tools, and outputs for processes in the Initiate phase.

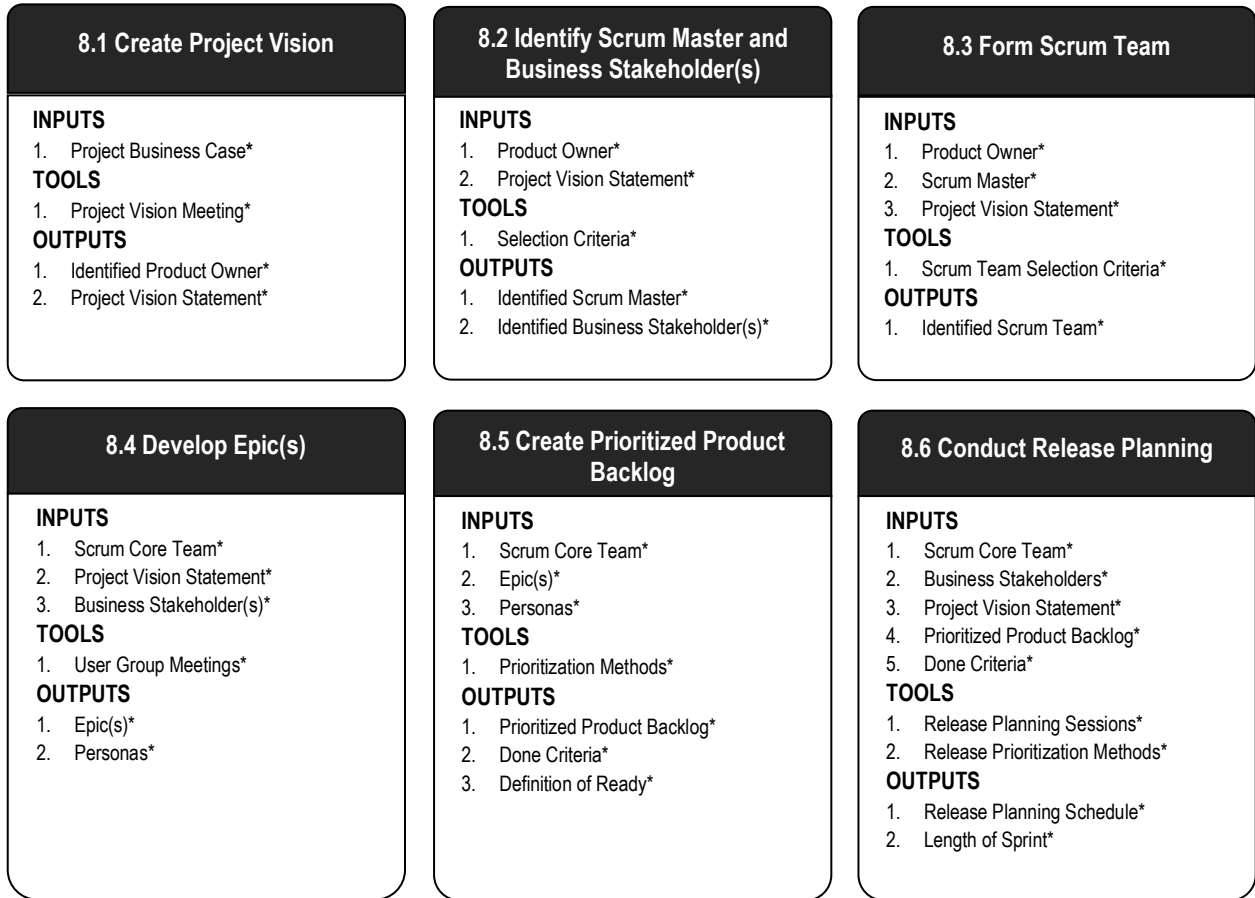


Figure 8-2: Initiate Overview (Essentials)

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

8.1 Create Project Vision

In this process, the Product Owner is identified. Based on the project's business case, the Product Owner then creates a Project Vision Statement. This Project Vision Statement provides the overall guidance, inspiration, and focus for the project. The company may also choose to evaluate and select an AI-powered Scrum project tool that can significantly reduce the complexity of setting up the Scrum initiative and increase the productivity of Scrum projects within the organization.

Figure 8-3 shows all the inputs, tools, and outputs for the *Create Project Vision* process.

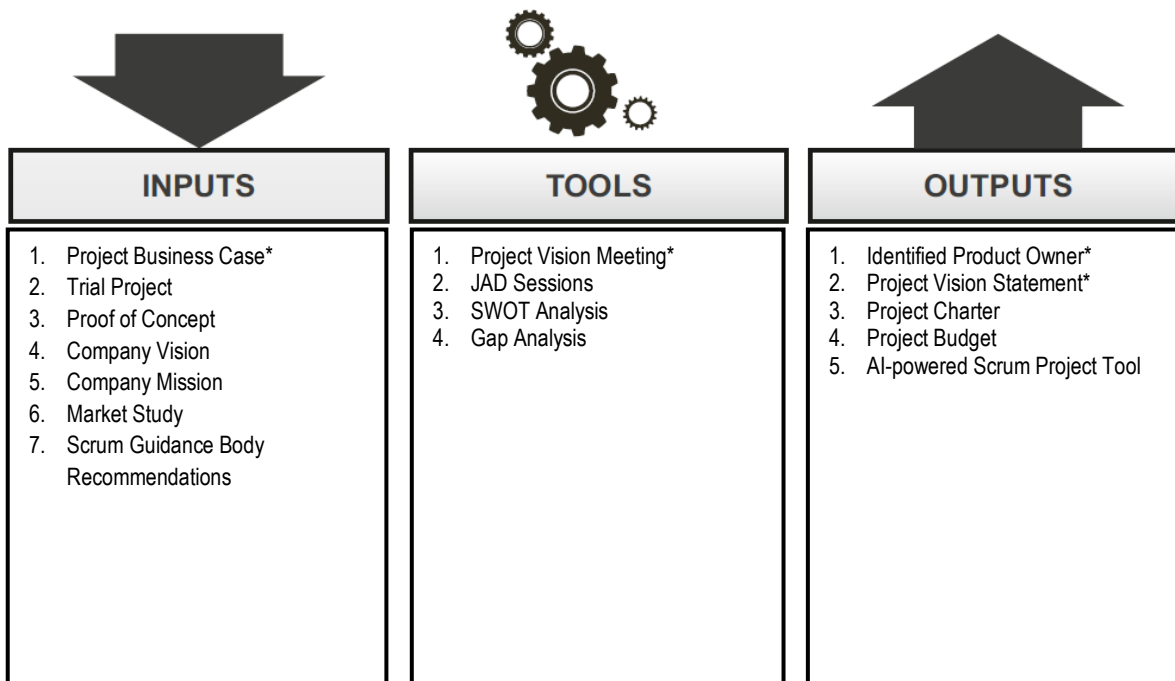


Figure 8-3: Create Project Vision—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 8-4 illustrates the data flow during the project vision creation process. It visually connects stakeholder inputs to decisions and deliverables for initial project alignment.

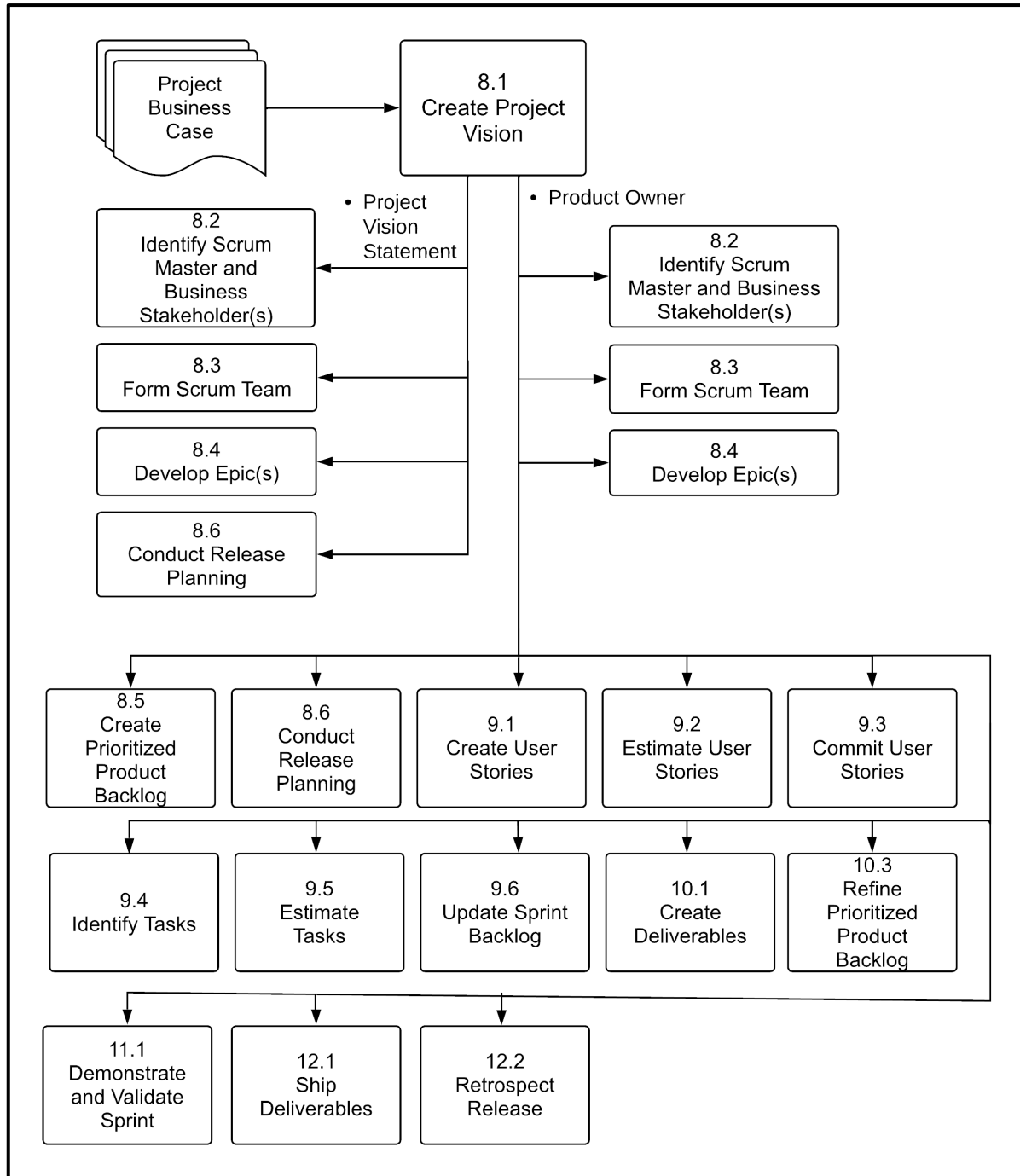


Figure 8-4: Create Project Vision—Data Flow Diagram

8.1.1 Inputs

8.1.1.1 Project Business Case*

The project commences with the presentation of the business case to the business stakeholders and sponsor(s). Business stakeholders should understand the expected business benefits of the project and the sponsor(s) should confirm that they will provide the financial resources for the project. A business case may be a well-structured document or simply a verbal statement that expresses the rationale for initiating a project. It may be formal and comprehensive, or informal and brief. The business case for a project often includes substantial information pertaining to the project's background, the intended business purpose and desired outcomes, SWOT and/or Gap analysis reports, a list of already identified risks, and any high-level estimations of time, effort, and cost. The project business case should also take into consideration any relevant external factors, such as legal regulations, government requirements, data privacy concerns, and so on.

8.1.1.2 Trial Project

If feasible, a small-scale demo or trial project could be run as an experiment to predict and evaluate the viability, time, costs, risks, and effects of the actual project. This helps evaluate the practical environment and guides the actual project design prior to the initiation of the project on a full scale.

8

8.1.1.3 Proof of Concept

Proof of concept demonstrates and verifies that the idea behind the current project is potentially viable in the real-world environment. Often in the form of a prototype, it is designed to determine financial and technical viability, help understand requirements and assist in the assessment of design decisions. However, Proof of Concept does not need to necessarily represent actual project deliverables.

8.1.1.4 Company Vision

Understanding the company vision helps the project keep its focus on the organization's long-term objectives and the future direction of the company. The Product Owner takes guidance and direction from the company vision to create the Project Vision Statement.

8.1.1.5 Company Mission

The company mission provides a framework for formulating the strategies of the company and guides overall decision making within the company. The Project Vision Statement must be framed such that its fulfillment helps the organization fulfill its mission.

8.1.1.6 Market Study

Market Study refers to the organized research, gathering, collation, and analysis of data related to customers' preferences for products. It often includes extensive data on market trends, market segmentation, and marketing processes. Market study could also include an analytical study of competitors which provides a better understanding of competitors' strengths and weaknesses and can help decision makers formulate better positioned products.

8.1.1.7 Scrum Guidance Body Recommendations

The Scrum Guidance Body (SGB) is an optional role. It consists of a group of documents and/or a group of experts who are typically involved with defining objectives related to quality, government regulations, security, and other key organizational parameters. These objectives guide the work carried out by the Product Owner, Scrum Master, and Scrum Team. The Scrum Guidance Body also helps to capture the best practices that should be used across all Scrum projects in the organization. The Scrum Guidance Body does not make decisions related to the project. Instead, it acts as a consulting or guidance structure for all the hierarchy levels in the project organization—portfolios, programs, and projects. Scrum Teams have the option of asking the Scrum Guidance Body members for advice when needed. It is important to ensure that the project vision aligns with recommendations provided by the Scrum Guidance Body and that processes comply with any standards and guidelines that are established.

8.1.2 Tools

8.1.2.1 Project Vision Meeting*

A Project Vision Meeting is a meeting with the relevant business stakeholders who understand the company's vision and the justification for the project. This may include the Chief Product Owner, Chief Scrum Master, Program Product Owner, Program Scrum Master, Portfolio Product Owner, and Portfolio Scrum Master if the project is a part of a larger project, program, or portfolio. This meeting helps identify the business context, business requirements, and business stakeholder expectations which can help when developing an effective Project Vision Statement. Applying Scrum practices involves engaging and collaborating with all business representatives to get their buy-in for the project and to deliver greater value.

8.1.2.2 JAD Sessions

A Joint Application Design (JAD) session is a requirement gathering technique. It is a highly structured facilitated workshop which hastens the *Create Project Vision* process as it enables the business stakeholder(s) and other decision makers to come to a consensus on the high-level scope, high-level objectives, and other specifications of the project.

JAD sessions consist of methods for increasing user participation, speeding development, and improving specifications. Relevant program and portfolio Product Owners and Scrum Masters could meet to outline and analyze the desired business outcomes and to discuss their vision for the upcoming Scrum project.

8.1.2.3 SWOT Analysis

A SWOT Analysis is a structured approach to project planning that helps evaluate the strengths, weaknesses, opportunities, and threats related to a project. This type of analysis helps identify both the internal and the external factors that could impact the project. Strengths and weaknesses are internal factors, whereas opportunities and threats are external factors. Identification of these factors helps business stakeholders and decision makers finalize the processes, tools, and techniques to be used to achieve project objectives. Conducting a SWOT Analysis allows for early identification of priorities, potential changes, and risks.

8.1.2.4 Gap Analysis

A Gap Analysis is a technique used to compare the current, actual state with some desired state. In an organization, it involves determining and documenting the difference between current business capabilities and the final desired set of capabilities. A project is normally initiated to bring an organization to the desired state, so conducting a Gap Analysis would help decision makers determine the need for the project.

Figure 8-5 explains the gap analysis process. It helps identify disparities between current performance and develop outcomes to plan corrective actions or improvements.

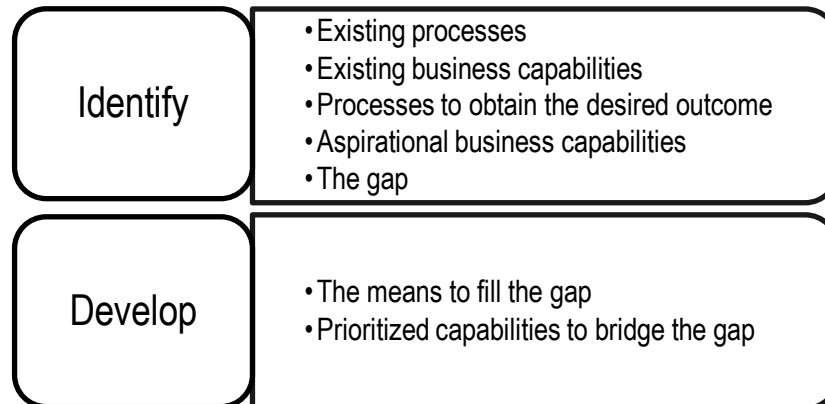


Figure 8-5: The Gap Analysis Process

8.1.3 Outputs

8.1.3.1 Identified Product Owner*

One of the outputs of this process is the identification of the Product Owner. The Product Owner is the person responsible for achieving maximum business value for the project. He or she is also responsible for articulating customer requirements and maintaining business justification for the project. The Product Owner represents the Voice of the Customer.

The Product Owner role is described in more detail in section 3.4.

8.1.3.2 Project Vision Statement*

The key output of the *Create Project Vision* process is a well-defined and structured Project Vision Statement. A good Project Vision statement explains the business need(s) the project is intended to meet (rather than how it will meet those needs).

The Project Vision Statement should not be too specific and should have room for flexibility. It is possible that the current understanding of the project may be based on assumptions that will change as the project progresses, so it is important that the statement is flexible enough to accommodate these changes. The project vision should focus on the problem rather than the solution.

8.1.3.3 Project Charter

A Project Charter is an official statement of the desired objectives and outcomes of the project. In most organizations, the Project Charter is the document that officially and formally authorizes the project, providing the team with written authority to begin the project work.

8.1.3.4 Project Budget

The project budget is a financial document that includes the cost of people, materials, and other related expenses in a project. The project budget is typically approved and signed off by the sponsor(s) to ensure that sufficient funds are available to complete the project. Once signed off, the Product Owner and the Scrum Master manage the project budget on a regular basis and also ensure that the people and other resources required for project activities are available.

8.1.3.5 AI-powered Scrum Project Tool

The selection of an AI-powered Scrum tool must be grounded in a thorough understanding of both agile methodologies and the capabilities of artificial intelligence. A well-executed selection process not only enhances Scrum effectiveness but also fosters a culture of data-driven decision-making and continuous improvement. The selection of an AI-powered Scrum project tool is a strategic decision that should align with the organization's agile maturity, operational needs, and long-term objectives. The following structured approach provides a formal framework for evaluating and selecting an appropriate tool.

1. Assess Organizational Requirements

- Begin by conducting a comprehensive assessment of the organization's project management needs, with a particular focus on how Scrum is currently implemented. This includes:
 - The scale of Scrum adoption across teams.
 - Current challenges in backlog management, sprint planning, or retrospectives.
 - The technical sophistication of the teams and their readiness to adopt AI-driven processes.
 - The desired level of automation, analytics, and predictive capabilities.

2. Define Selection Criteria

- Clearly defined criteria are essential for an objective comparison of available tools. Criteria should include both functional and non-functional requirements:
 - Functional Criteria
 - Support for all core Scrum components (Epics, User Stories, Tasks, Sprint Backlogs, Retrospectives).
 - AI-driven features such as:
 - Automated backlog refinement
 - Recommended breakdown and classifications for projects, Epics, User Stories
 - AI-powered reporting and Data Analytics
 - Intelligent prioritization
 - Predictive risk identification
 - Sentiment analysis in retrospectives

Non-Functional Criteria

- Ability to integrate with teams using other methodologies such as Kanban Workflows, DevOps, OKRs etc.
- Ease of integration with existing tools (e.g., Vabro, Git, Jira, Slack, CI/CD pipelines).
- Scalability to accommodate multiple teams or departments.
- Compliance with data security and privacy standards (e.g., GDPR, SOC 2).
- Usability and learning curve for end-users.

3. Market Research and Tool Shortlisting

Conduct thorough research on available AI-powered Scrum tools in the market. Consider industry benchmarks, analyst reports, and peer reviews. Based on the initial assessment and criteria, shortlist 3–5 tools for deeper evaluation. Examples of prominent tools in this space include:

- Vabro with Vabro Genie AI
- Jira with Atlassian Intelligence
- ClickUp AI
- Forecast.ai
- Monday.com (AI integrations)

Figure 8-6 depicts a screenshot from the Vabro tool used in Scrum projects. It highlights functionalities like backlog tracking, team collaboration, and project planning.

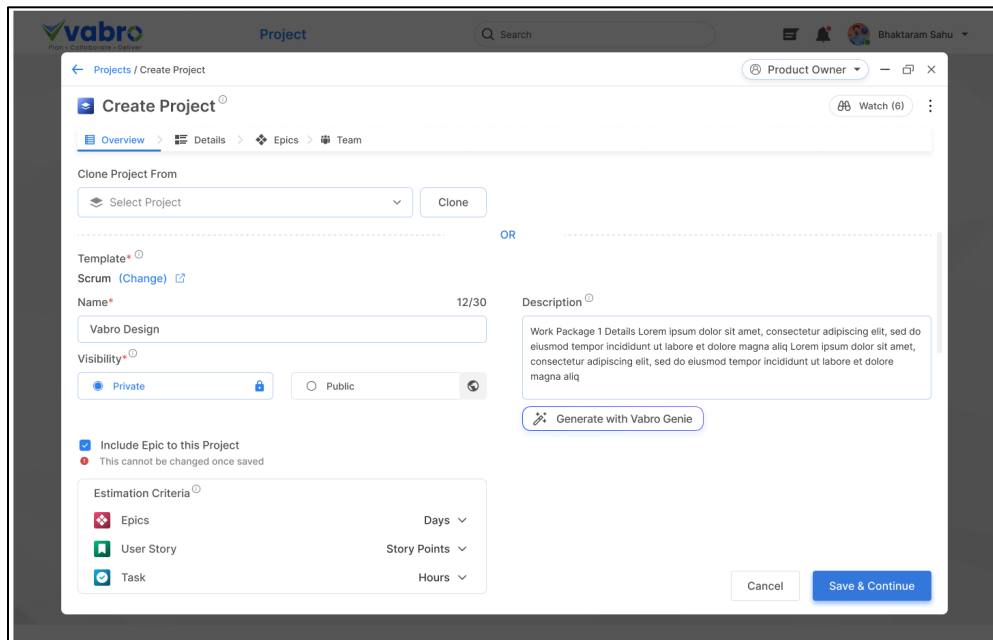


Figure 8-6: Scrum Project Tools – Vabro

Figure 8-7 shows Jira's Scrum interface for project management. It includes tools for sprint planning, backlog grooming, and tracking progress through agile workflows.

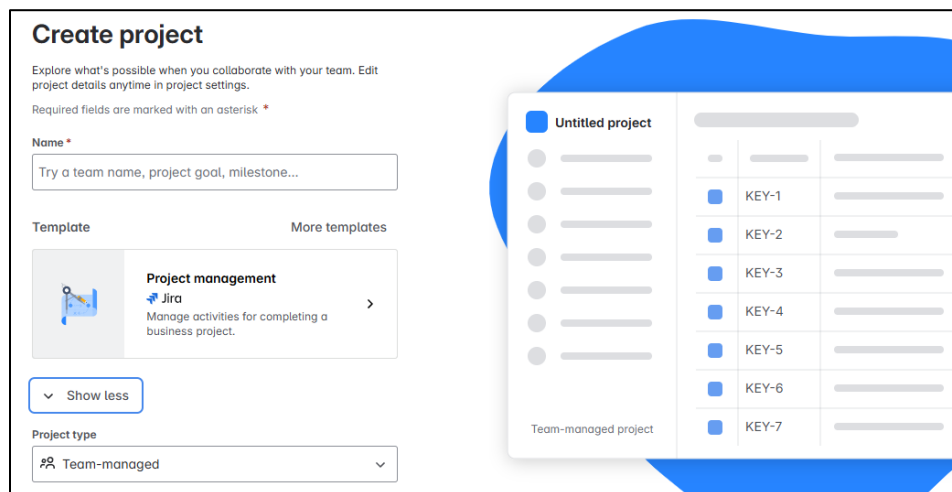


Figure 8-7: Scrum Project Tools - Jira

4. Conduct Pilot Evaluations

Organize pilot programs for each shortlisted tool using real project data and team workflows. During the pilot phase, measure the following:

- Accuracy and relevance of AI recommendations.
- Improvement in team productivity and planning efficiency.
- User adoption and satisfaction.
- Technical performance and reliability.
- Feedback should be collected from all key Scrum roles: Product Owners, Scrum Masters, and Development Team members.

5. Analyze ROI and Total Cost of Ownership

Evaluate the total cost of ownership (TCO), including licensing fees, training, support, and implementation time. Estimate the return on investment (ROI) based on projected gains in productivity, reduced cycle times, and improved delivery accuracy due to AI-enhanced decision-making.

6. Final Selection and Implementation Planning

Based on the evaluation outcomes, select the tool that best meets organizational goals. Develop a phased implementation plan that includes:

- Training sessions for Scrum teams
- Integration with existing systems
- Governance and change management strategies
- Ongoing performance monitoring and feedback loops

8.2 Identify Scrum Master and Business Stakeholder(s)

In this process, the Scrum Master is identified using specific selection criteria that can effectively assess the soft skills and Scrum knowledge needed for this key role. Additionally, business stakeholders are also identified during this process.

Figure 8-8 shows all the inputs, tools, and outputs for the *Identify Scrum Master and Business Stakeholder(s)* process.

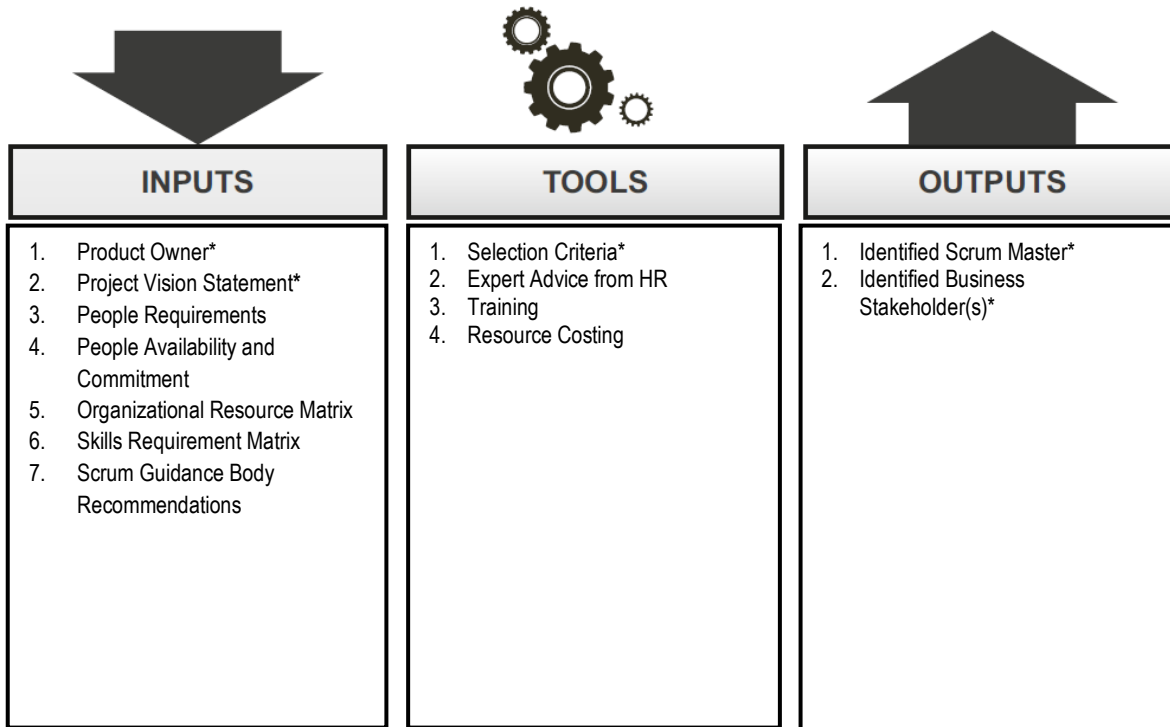


Figure 8-8: Identify Scrum Master and Business Stakeholder(s)—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 8-9 illustrates the data flow for Identifying the Scrum Master and Business Stakeholders. It maps how selection criteria and organizational needs inform the identification process.

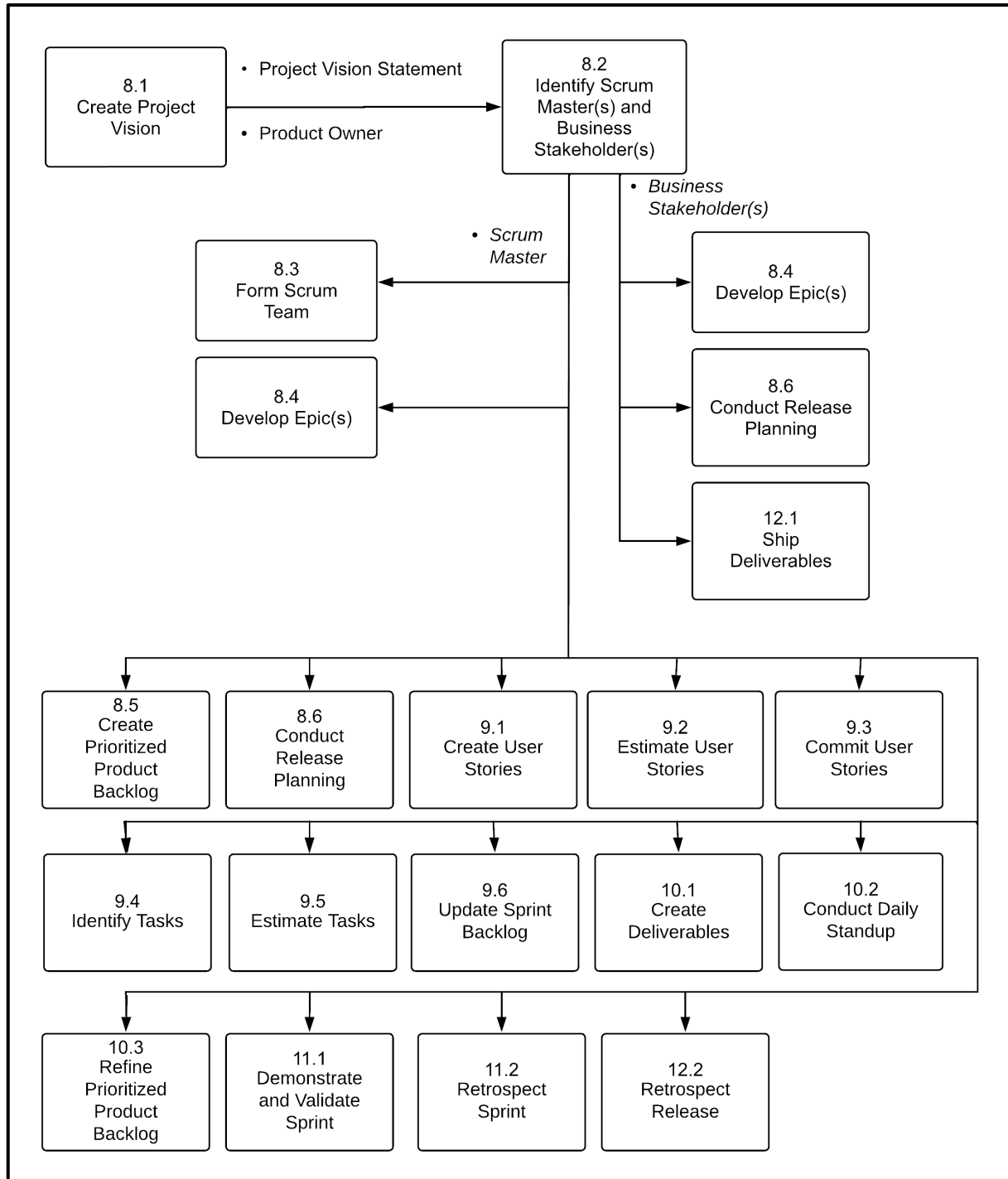


Figure 8-9: Identify Scrum Master and Business Stakeholder(s)—Data Flow Diagram

8.2.1 Inputs

8.2.1.1 Product Owner*

Described in section 8.1.3.1.

8.2.1.2 Project Vision Statement*

Described in section 8.1.3.2.

8.2.1.3 People Requirements

Identifying the required skill set of the Scrum Master is one of the initial steps to be performed before a Scrum Team can be fully formed (see process 8.3). It is important to document the role and responsibilities of the Scrum Master position, and the desired knowledge and skills needed to ensure a project's success. Usually, the Product Owner works with the Human Resource department or other relevant internal stakeholders to determine and finalize the Scrum Master's role description.

Key business stakeholders are also identified and their role in the project is determined.

8.2.1.4 People Availability and Commitment

Prior to selecting the Scrum Master and business stakeholder(s), their availability and commitment must be confirmed. Only Scrum Masters and business stakeholders who are available and can fully commit to the project should be considered. Availability is commonly depicted in the form of calendars showing if, and when human resources are available to work throughout the duration of the project. Commitment can then be gained once the availability of the desired Scrum Master (and business stakeholders) is confirmed. The Human Resource department can assist the Product Owner in selection of the appropriate candidates.

8.2.1.5 Organizational Resource Matrix

The Organizational Resource Matrix is a hierarchical depiction of a combination of a functional organizational structure and a projectized organizational structure. Matrix organizations bring together project team members from different functional departments, such as information technology, finance, marketing, sales, manufacturing, and other departments, to create cross-functional teams. Team members in a matrix organization fulfill two objectives—functional and project. Team members are directed by the Product Owner with respect to project-related activities, while the functional managers perform managerial activities related to their departments, such as performance appraisals and approving leaves.

8.2.1.6 Skills Requirement Matrix

The Skills Requirement Matrix, also known as a competency framework, is used to assess skill gaps and training requirements for team members. A skills matrix maps the skills, capabilities, and interest level of team members in using those skills and capabilities required for a project. Using this matrix, the organization can assess any skill gaps in team members and identify the employees who will need further training in a particular area or competency.

8.2.1.7 Scrum Guidance Body Recommendations

Described in section 8.1.1.7.

8.2.2 Tools

8.2.2.1 Selection Criteria*

Selecting the appropriate Scrum Master and identifying relevant business stakeholders(s) is crucial to the success of any project. In some projects, some roles (such as Scrum Master) may already be preassigned prior to project initiation. When there is flexibility in choosing a Scrum Master, the following are important selection criteria:

1. **Problem-solving skills**—This is one of the primary criteria to be considered while selecting a Scrum Master. The Scrum Master should have the necessary skills and experience to help remove any impediments for the Scrum Team.
2. **Availability**—The Scrum Master should be available to schedule, oversee, and facilitate various meetings, including the Release Planning Meeting, Daily Standup Meeting, and other Sprint-related meetings.
3. **Commitment**—The Scrum Master should be highly committed to ensure that the Scrum Team is provided with a conducive work environment to ensure successful delivery of Scrum projects.
4. **Supporting Leadership Style** (see section 3.9.4.1).

When identifying the business stakeholders, it is important to remember that business stakeholders are all the customers, users, and sponsors, who frequently interface with the Product Owner, Scrum Master, and Scrum Team to provide inputs and facilitate creation of the project's products. The business stakeholders influence the project throughout its lifecycle.

8.2.2.2 Expert Advice from HR

Expert advice from Human Resource personnel can be valuable in identifying the Scrum Master and the business stakeholders. The HR team members possess knowledge about the skill sets of employees within the organization and they also have the specialized knowledge and experience of various techniques that might help in identifying the Scrum Master and business stakeholder(s).

8.2.2.3 Training

Scrum is a radically different framework from traditional methods of project management. Team members may not yet possess the required knowledge or skills needed for working in a Scrum environment. The Product Owner should evaluate the training needs of potential team members and facilitate training to bridge any knowledge gap in the team. The Product Owner is normally responsible for evaluating and selecting team members, but often does this in consultation with the Scrum Master who may have additional knowledge of the resources from working with them on other projects.

Appropriate training should be provided to the Scrum Team members both prior to the commencement of work and also while they are working on their projects. Scrum Team members should be ready to learn from each other and from more experienced persons in the team.

8.2.2.4 Resource Costing

One of the primary considerations when selecting the appropriate Scrum Master for the project has to do with the trade-offs related to experience versus salary. There are other people-related factors impacting cost that may also need to be considered. Ideally, the Scrum Master, team members, and business stakeholder(s) should be colocated, so that they can communicate frequently and easily. If colocation is not possible and the team is distributed, additional resources may be needed to facilitate communications, understand cultural differences, synchronize work, and foster knowledge sharing.

8.2.3 Outputs

8.2.3.1 Identified Scrum Master*

A Scrum Master is a facilitator and supporting leader who ensures that the Scrum Team is provided with an environment conducive to completing the project successfully. The Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved in the project, clears impediments for the team, and ensures that Scrum processes are being followed. It is primarily the responsibility of the Product Owner to identify the Scrum Master for a Scrum project. The Scrum Master role is described in more detail in section 3.5.

8.2.3.2 Identified Business Stakeholder(s)*

Business stakeholder(s) is a collective term that includes customers, users, and sponsors who frequently interface with the Scrum Core Team and influence the project throughout the product development process. It is for the business stakeholders that the project produces collaborative benefits. The business stakeholder(s) role is described in section 3.2.2.

8.3 Form Scrum Team

In this process, Scrum Team members are identified based on the skills required to complete the project deliverables, as well as considerations for the availability, costs, and soft skills important for members of a Scrum Team. Normally the Product Owner has the primary responsibility of selecting team members but often does so in collaboration with the Scrum Master.

Figure 8-10 shows all the inputs, tools, and outputs for the *Form Scrum Team* process.

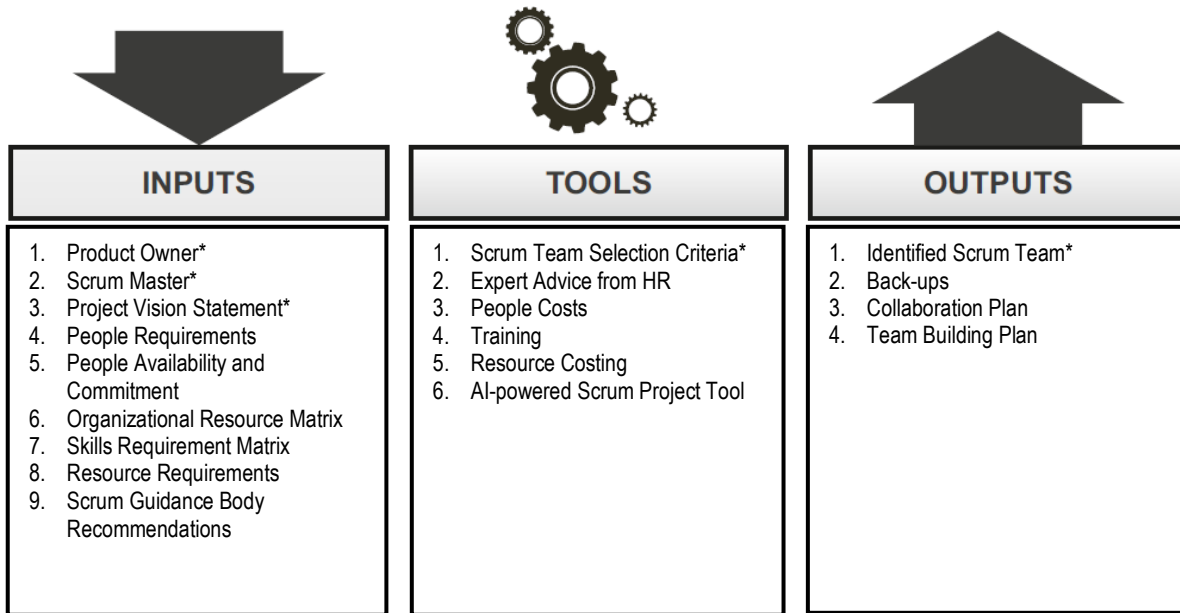


Figure 8-10: Form Scrum Team—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 8-11 explains the data flow for Form Scrum team. It charts the process from identifying team needs to assembling a cohesive, functional team unit.

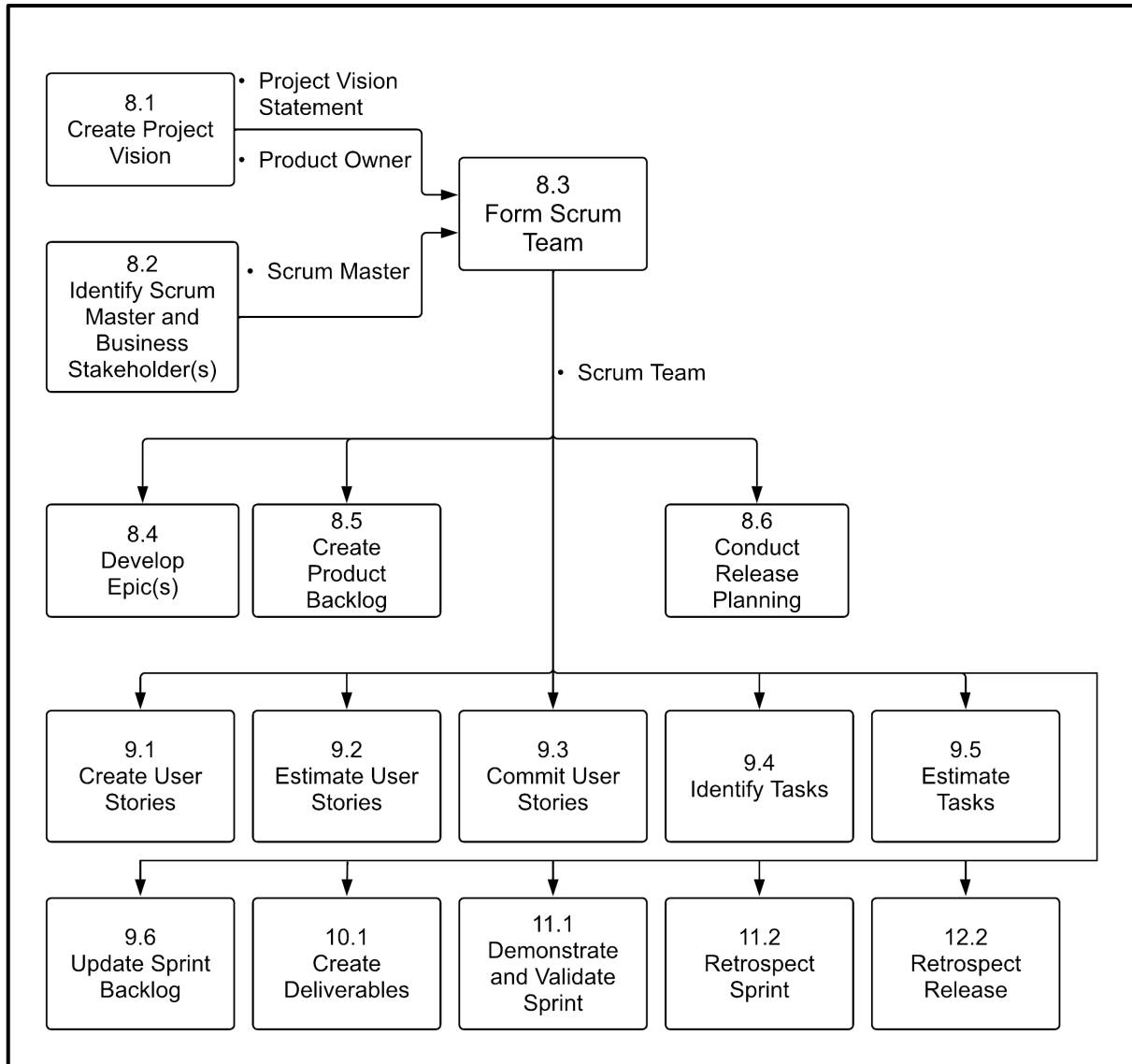


Figure 8-11: Form Scrum Team—Data Flow Diagram

8.3.1 Inputs

8.3.1.1 Product Owner*

Described in section 8.1.3.1.

8.3.1.2 Scrum Master*

Described in section 8.2.3.1.

8.3.1.3 Project Vision Statement*

Described in section 8.1.3.2.

8.3.1.4 People Requirements

It is important to identify and document the roles and responsibilities of the Scrum Team members and the desired knowledge and skill sets needed to ensure a project's success. Usually, the Product Owner works with the Human Resource department or other relevant people in the company to determine and finalize the roles and responsibilities of Scrum Team members.

8

8.3.1.5 People Availability and Commitment

Prior to selecting Scrum Team members, their availability and commitment must be confirmed. Availability is commonly depicted in the form of resource calendars showing if, and when human resources are available to work throughout the duration of the project. Commitment can then be gained once the availability of the desired team members is confirmed. Only team members who are available and can fully commit to the project should be selected as Scrum Team members. The Human Resource department can assist the Product Owner in selection of the appropriate candidates.

To be effective, Scrum Teams should ideally have six to ten members. Changing team members is not advisable in Scrum Core Teams. So, it is important to have Scrum Core Team members who are fully available and committed to the project.

8.3.1.6 Organizational Resource Matrix

Described in section 8.2.1.5.

8.3.1.7 Skills Requirement Matrix

Described in section 8.2.1.6.

8.3.1.8 Resource Requirements

These requirements include all the resources—other than people—required for the Scrum Team to function effectively. Examples of these resources include office infrastructure, meeting space, work equipment, and the appropriate tools needed to apply Scrum effectively (e.g., Scrumboard, Burndown Charts, index cards, estimation cards, etc.). In the case of virtual teams, additional resources such as collaboration tools, video conferencing, shared document repositories, and translation services must be considered.

8.3.1.9 Scrum Guidance Body Recommendations

Described in section 8.1.1.7.

8.3.2 Tools

8.3.2.1 Scrum Team Selection Criteria*

The Scrum Team is the core of any Scrum project and getting the appropriate team members is important for the successful delivery of a Scrum project. Scrum Team members are expected to be generalists/specialists in that they have knowledge of various fields and are experts in at least one. Beyond their subject-matter expertise, it is the soft skills of team members that determine the success of self-organizing teams. Ideal members of a Scrum Team are independent, self-motivated, customer-focused, responsible, and collaborative. The team should be able to foster an environment of independent thinking and group decision-making in order to extract the most benefits from the structure.

8.3.2.2 Expert Advice from HR

Expert advice from Human Resource (HR) personnel can be valuable while forming a Scrum Team. The HR team members possess knowledge about the skill sets of employees in the organization and they also have the specialized knowledge and experience of various techniques that might help Product Owners, Scrum Masters, and sponsors identify the right team members.

8.3.2.3 People Costs

All costs associated with people requirements need to be assessed, analyzed, approved, and budgeted for.

8.3.2.4 Training

Team members may not possess all the required skill sets or knowledge to carry out specialized tasks. The Product Owner and/or Scrum Master should evaluate the training needs of potential team members. Once candidates are selected, the appropriate training should be provided for any skill or knowledge gaps found. For a truly effective Scrum implementation, there must be a significant level of awareness within the organization of Scrum principles and values. This awareness aids in the successful execution of Scrum. The Scrum Team has to be sensitized and trained in the practices of Scrum and the Scrum Master should play the role of a coach for the team. Because planning Sprints is a major success factor, training will help teams understand how to discuss and identify achievable Sprint goals. The Scrum Master needs to bring out the best from the Scrum Team members by motivating them and facilitating the development process. By training and coaching team members, the Scrum Master can help them articulate any issues and challenges they may face. Normally any issues or conflicts experienced within the team are solved by the team with coaching and assistance from the Scrum Master as required. The Scrum Master should address issues such as low morale or lack of coordination within the team. He or she is responsible for removing impediments for the team. When required, the Scrum Master can escalate external issues and impediments to management for resolution or removal.

Training is also discussed in the *Identify Scrum Master and Business Stakeholder(s)* process in section 8.2.2.3.

8.3.2.5 Resource Costing

The costs associated with all non-people requirements must be assessed, analyzed, approved, and budgeted for. A resource in the project environment is anything used to perform a task or activity including—but not limited to—equipment, material, outside services, and physical space.

8.3.2.6 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 8-12 depicts the Vabro interface used for creating Scrum teams. It features roles, permissions, and collaboration tools within a Scrum project environment.

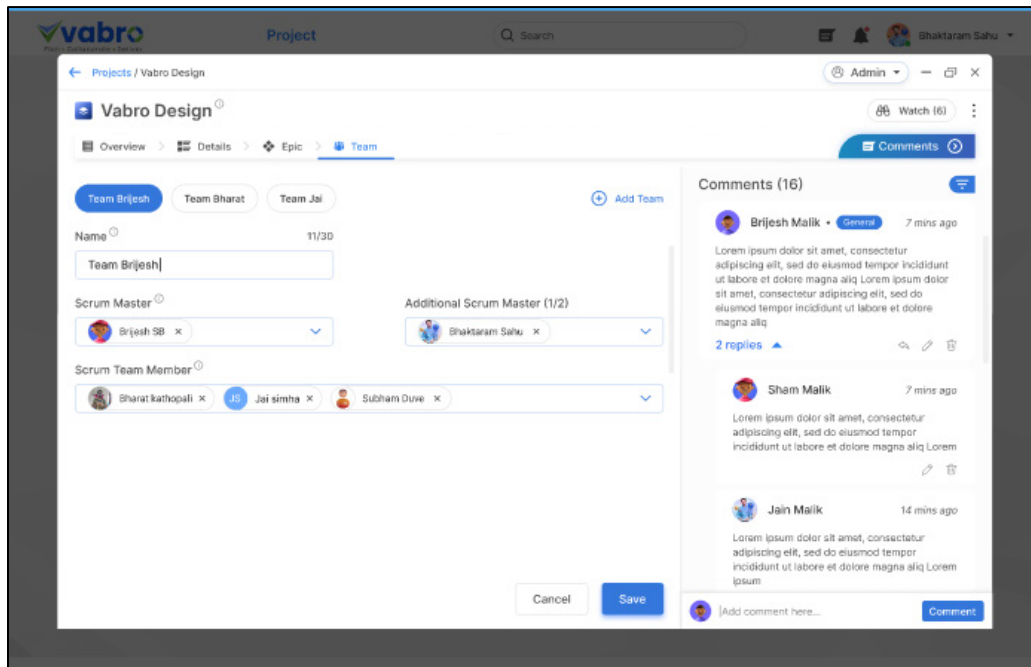


Figure 8-12: Creating a Team (Source: Vabro)

Figure 8-13 shows the team setup screen in Jira. It allows team member configuration, permissions assignment, and integration into ongoing Scrum workflows.

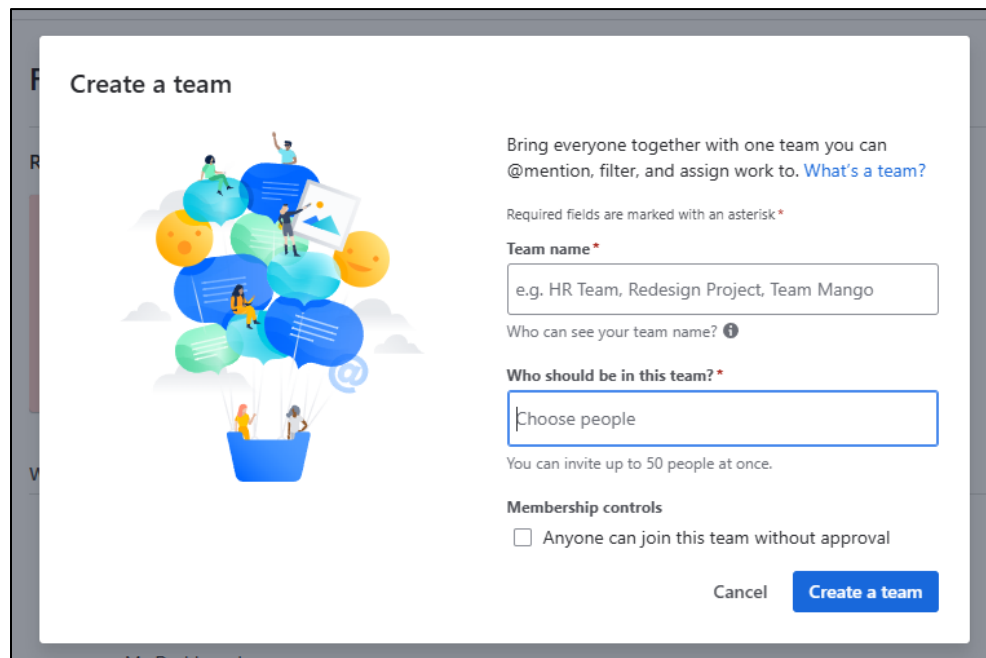


Figure 8-13: Team Set-up (Source: Jira)

8.3.3 Outputs

8.3.3.1 Identified Scrum Team*

The Scrum Team, sometimes referred to as the Development Team, is a group or team of people responsible for understanding the business requirements specified by the Product Owner, estimating User Stories, and creating the project deliverables. The Scrum Team consists of cross-functional team members who carry out all the work involved in creating potentially shippable deliverables, including work related to satisfying the desired quality assurance and quality control parameters of each deliverable. Scrum Teams are cross-functional and self-organizing. The team decides the amount of work to commit to in a Sprint and determines the best way to perform the work. Identifying the Scrum Team is the responsibility of the Product Owner, often in consultation with the Scrum Master. The Scrum Team role is described in more detail in section 3.6.

8.3.3.2 Back-ups

When selecting team members, it is important to identify backups for critical skills, preferably within the same Scrum Team. Although people availability and commitment are confirmed for team members in advance, issues may arise such as illness, family emergencies, or team member loss. Scrum Teams work in small groups of six to ten people. Having back-ups assigned to tasks ensures that there is no major decrease in productivity due to the loss of a team member.

8.3.3.3 Collaboration Plan

Collaboration is an important Scrum principle. Planning for how the various decision makers, business stakeholders, and team members engage and collaborate with each other is vital. The Collaboration Plan is an optional output that may be formal or informal. At times, it may simply be an oral understanding between the various business stakeholders and the Scrum Team (since creating unnecessary documentation should be avoided). However, for larger, more complex projects, especially those with distributed teams, a more formal agreement may need to be put in place. The plan may address how the Scrum Core Team members, business stakeholders(s) and others involved in the Scrum project will communicate and collaborate throughout the project and may also define specific tools or techniques to be used for that purpose. For example, in distributed teams, there may be a need for an agreement on when and how meetings will be conducted, what type of communication tools will be used, and who should be involved in each type of meeting.

8.3.3.4 Team Building Plan

Since a Scrum Team is cross-functional, each team member needs to participate actively in all aspects of the project. The Scrum Master identifies issues with team members and addresses them diligently in order to maintain an effective team. To build team cohesion, the Scrum Master should ensure that relationships among the team members are positive and that the team members are unified in achieving the overall project and organizational goals, thus leading to greater efficiency and increased productivity.

Section 3.9 discusses popular HR theories and their relevance to a Scrum project environment.

8.4 Develop Epic(s)

In this process, the Project Vision Statement serves as the basis for developing Epics, which define the high-level requirements for the project. The Product Owner may use User Group meetings and other tools to collect requirements from business stakeholders.

Figure 8-14 shows all the inputs, tools, and outputs for the *Develop Epic(s)* process.

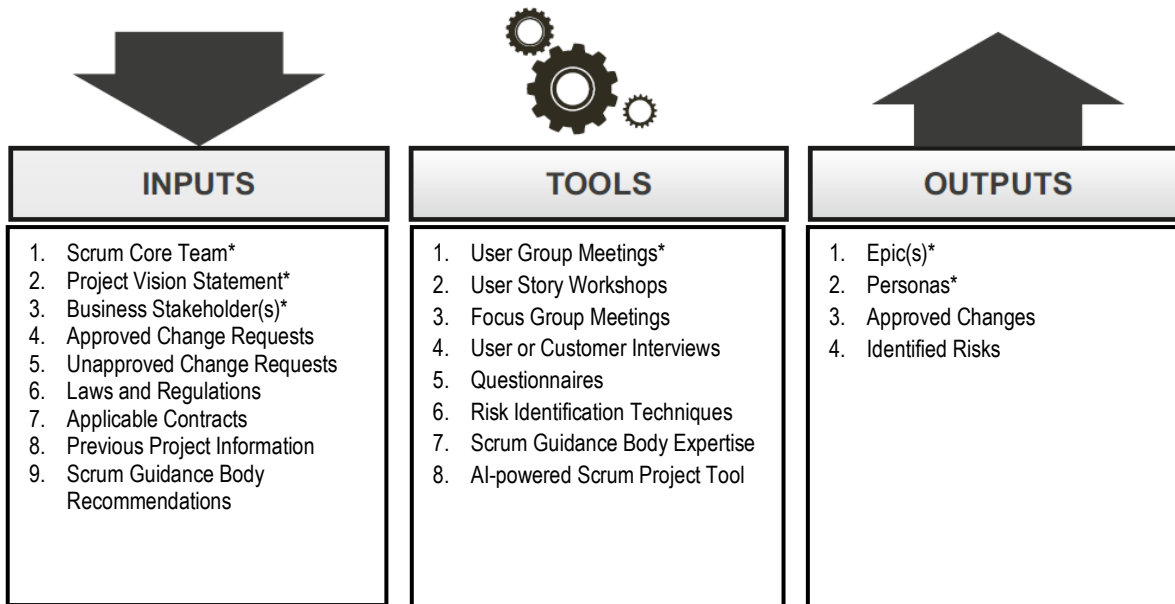


Figure 8-14: Develop Epic(s)—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 8-15 explains the data flow in Develop Epics. It visualizes the transformation of business inputs into actionable backlog components using Scrum techniques.

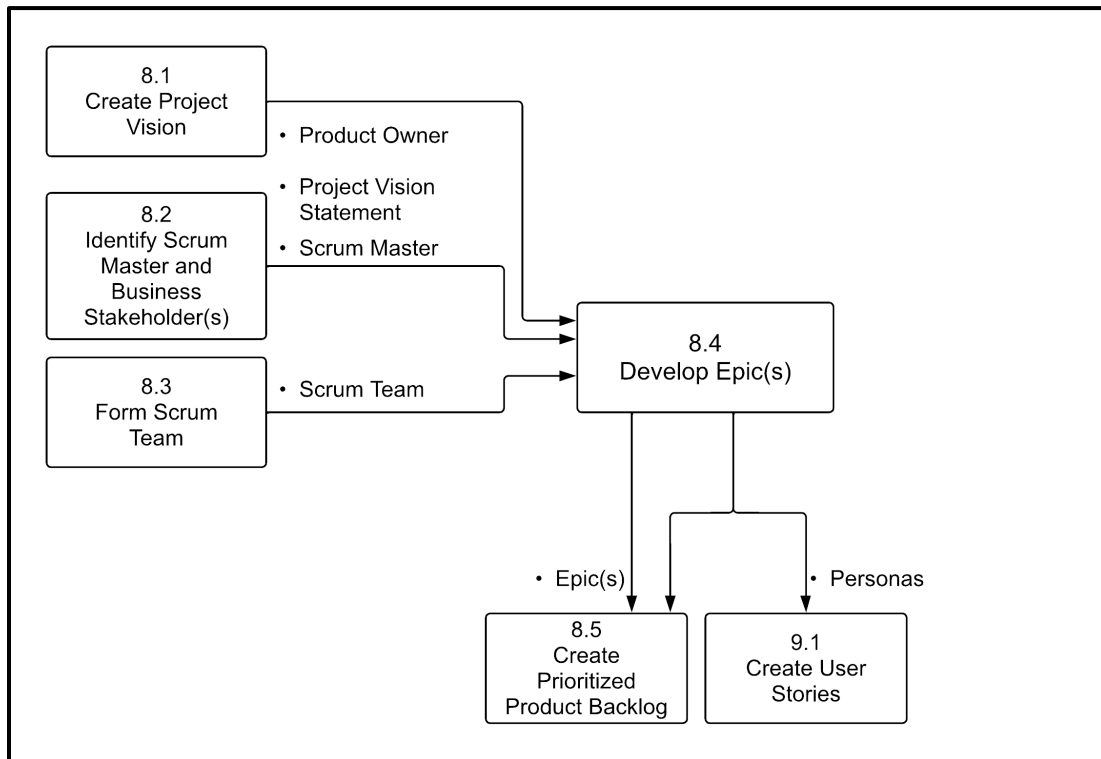


Figure 8-15: Develop Epic(s)—Data Flow Diagram

8.4.1 Inputs

8.4.1.1 Scrum Core Team*

The Scrum Core Team consists of the Scrum Team, the Scrum Master, and the Product Owner as described in section 3.2.1.

8.4.1.2 Project Vision Statement*

Described in section 8.1.3.2.

8.4.1.3 Business Stakeholder(s)*

Described in section 8.2.3.2.

8.4.1.4 Approved Change Requests

Approved Change Requests originating from the business, or more specifically a program or portfolio, are inputs to be added to the list of approved project changes for implementation in future Sprints. Each change can require its own Epic or User Story and could become an input to the *Develop Epic(s)* process. Approved Change Requests could also result from other Scrum processes, where they are initially considered unapproved Change Requests (until they are discussed and approved).

Change Requests and Approved Change Requests are discussed in sections 6.3.1, 6.4.2.1, and 6.6.

8.4.1.5 Unapproved Change Requests

Requests for changes are usually submitted as Change Requests and remain in an unapproved status until they are formally approved. Unapproved Change Requests in the *Develop Epic(s)* process could come from the *Create Deliverables*, *Conduct Daily Standup*, and sometimes other processes. Change Requests and Unapproved Change Requests are discussed in sections 6.3.1 and 6.4.2.1.

8.4.1.6 Laws and Regulations

Depending on the project, there may be applicable laws and regulations imposed by governing bodies, which impact the planning and execution of the project. Laws are external to the organization and imposed by a governmental entity. Regulations can either be internal or external. Internal regulations are those that are applicable within the company, typically based on organizational policies. Internal regulations often pertain to company procedures and processes related to quality management, financial management, staffing management, and so on. External regulations are those relating to government-established standards, norms, and/or requirements. Laws and regulations must be considered while developing Epics. Although Epics are based on business requirements, to meet those requirements, the project team must abide by both applicable internal and external laws and regulations.

At times, some of the laws and regulations impacting multiple Scrum projects may be included as part of the Scrum Guidance Body Recommendations, as discussed in section 8.1.1.7.

8.4.1.7 Applicable Contracts

If the entire project or portions of it are being completed under a contract, the contract defines the scope of work and the specific terms of the contract. The type of contract used influences project risk.

Some of the most common types of contracts used in Scrum projects are as follows:

- **Incremental Delivery Contract**—This contract mandates inspection points at regular intervals. It helps the customer and business stakeholders make decisions regarding product development periodically throughout the project at each inspection point.

The customer can either accept the current development of the product, decide to stop the development, or request product modifications.

- **Joint Venture Contract**—This contract is used when two or more parties partner to accomplish the work of a project. The parties involved in the project will both achieve some Return on Investment because the revenues or benefits generated will be shared between the parties.
- **Development in Phases Contract**—This contract makes funding available each month or each quarter after a release is successfully completed. It gives incentive to both customer and supplier and ensures that the monetary risk for the customer is limited to that particular time period since unsuccessful releases are not funded.
- **Incentive and Penalty Contract**—This contract is based on the agreement that the supplier will be rewarded with a financial incentive if the project's products are delivered on time but will incur financial penalties if the delivery is late.

Other popular contract types include paying by features contract, time and materials contract, fixed price and fixed scope contract, and fixed profit contract.

Epics should be developed keeping in mind the terms and conditions of the contract and the contract type being used.

8.4.1.8 Previous Project Information

Information and insights gained from previous similar projects within the organization are valuable inputs for developing Epics and assessing risk. Previous project information may be reflected in previous Scrum Core Team members' notes, lessons learned documents, and stakeholder feedback. Best practices and other useful information may also be available through the Scrum Guidance Body recommendations.

8.4.1.9 Scrum Guidance Body Recommendations

Scrum Guidance Body recommendations may include information on rules, regulations, standards, and best practices for developing Epics. For more information on the Scrum Guidance recommendations, see section 8.1.1.7.

8.4.2 Tools

8.4.2.1 User Group Meetings*

User Group Meetings provide the Product Owner with firsthand information about user expectations from relevant business stakeholders (primarily users and/or customers). Relevant members from the Scrum Core Team may also be present at User Group Meetings. These meetings help in formulating the Acceptance Criteria for the product and provide valuable insights for developing Epics. User Group meetings also promote buy-in for the project and create a mutual understanding among the Product Owner, Scrum Team, and relevant business stakeholders. User Group Meetings are vital in the prevention of expensive rework that may result from a lack of clarity regarding expectations and requirements.

8.4.2.2 User Story Workshops

User Story Workshops may be held as part of the *Develop Epic(s)* process. The Scrum Master facilitates these sessions with the entire Scrum Core Team, and at times, other business stakeholders. These workshops help the Product Owner to prioritize requirements and enable the Scrum Core Team to gain a shared perspective of the Acceptance Criteria. They ensure that the Epics and User Stories describe the functionality from the users' point of view, are easy to understand, and can be reliably estimated. User Story Workshops are useful in understanding user expectations for the deliverables and are excellent for team building. They also facilitate preparation for Sprint planning. A User Story Workshop is a good platform to discuss and clarify every element of a product and often delve into the smallest details to ensure clarity.

8.4.2.3 Focus Group Meetings

Focus groups assemble individuals in a guided session to provide their opinions, perceptions, or ratings of a product, service, or desired result. Focus group members have the freedom to ask questions to each other and to get clarifications on particular subjects or concepts. Through questioning, constructive criticism, and feedback, focus groups lead to a better and higher quality product and thereby contribute to meeting the expectations of the users. In these meetings, the focus group members sometimes reach consensus in certain areas, while in other areas their opinions may differ. Where group members have differing opinions or perspectives, every effort is made to resolve the differences in order to reach consensus.

Focus group sessions can help teams come up with innovative ideas, solve problems, and give suggestions for improvement. These meetings facilitate fact-finding and generate ideas and feedback from potential users and product developers. They are usually conducted to assist with planning, evaluating, or improving a product or service. Insights obtained from these meetings can also help develop Epics and User Stories. At times, Focus Group Meetings are conducted to resolve issues that may arise during the development of Epics.

8.4.2.4 User or Customer Interviews

Engaging business stakeholders, including the sponsor(s), users, and customers of the product, is important to gain the necessary context and insight required to develop Epics. Quality time spent interviewing users and/or customers can help to ensure that the requirements of Epics align with the overall project vision, thereby delivering greater value.

These interviews help to:

- Identify and understand business stakeholders' needs and expectations
- Gather opinions and facts
- Understand business stakeholders' perspective of the end product
- Gather feedback about the iterated or partially developed product
- Get buy-in and commitment from users and/or customers

8.4.2.5 Questionnaires

A cost-effective way to gain quantitative and qualitative statistical insight from a large number of users or customers is to use surveys or questionnaires. A questionnaire is a research or data collection tool that contains questions to be asked to a chosen set of individuals in order to collect information about a specific issue or topic. Questionnaires can be self-administered or administered by an interviewer.

Great care must be exercised in the design of questionnaires, selecting the right target audience, and determining an appropriate method of survey deployment to avoid errors and bias. While developing Epics, the Product Owner or the Scrum Master might conduct a survey to gather relevant information from business stakeholders or the Scrum Team.

8.4.2.6 Risk Identification Techniques

Described in section 7.4.1.1.

8.4.2.7 Scrum Guidance Body Expertise

While creating Epics, Scrum Guidance Body Expertise could refer to documented regulations, standards, and/or best practices for creating Epics. There may also be a team of subject matter experts who are available to assist the Product Owner in developing Epics. This team could include business analysts, lead architects, senior developers, Scrum experts, and other experienced people. This expert group is usually not the same team that will stay on and work on a particular project, as they tend to move from project to project during the 'selling phase' or 'phase zero' which involves customers and/or users. For more information on the Scrum Guidance Body, see section 3.2.2.

8.4.2.8 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 8-16 shows Vabro's interface for epic development. It includes features for creating, managing, and organizing epics aligned with Scrum project goals.

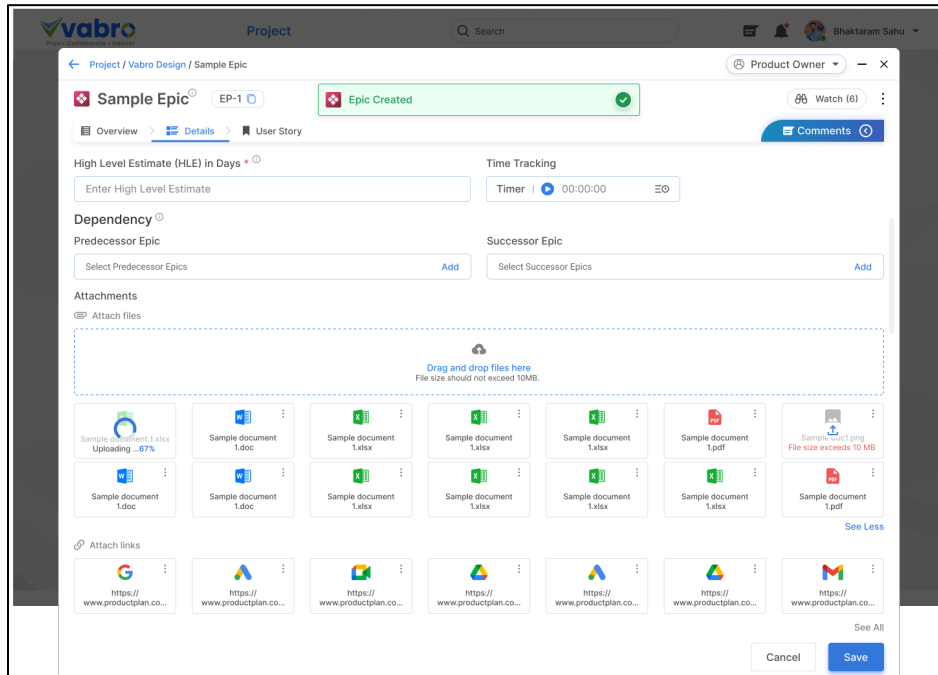


Figure 8-16: Sample Scrum Project Tool – Vabro

Figure 8-17 depicts the Jira tool used for managing epics. It supports backlog prioritization, user story linkage, and progress tracking within Scrum.

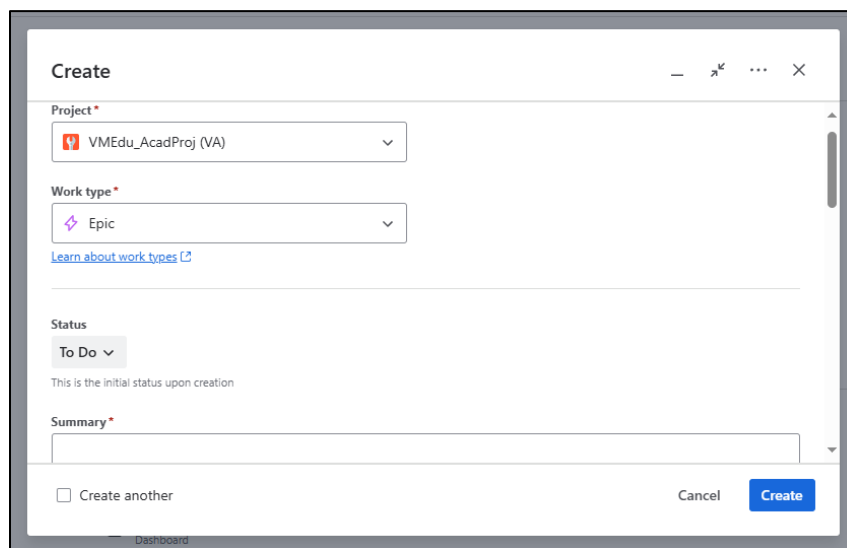


Figure 8-17: Sample Scrum Project Tool – Jira

8.4.3 Outputs

8.4.3.1 Epic(s)*

Epics are written in the initial stages of a project when most requirements are high-level functionalities or when product descriptions are broadly defined. Epics exist as large, unrefined User Stories in the Prioritized Product Backlog. Epics help the Product Owner and relevant business stakeholders in planning for releases, prioritizing high-level requirements, and establishing an overall roadmap for the project. As the Product Owner gets additional clarity on user requirements, these Epics are then broken down into smaller, more granular User Stories. So, Epics contain one or more User Stories. These smaller User Stories are simple, short, and easy-to-implement functionalities or blocks of tasks to be completed in a Sprint. User Stories should be defined by the Product Owner and need to satisfy the Definition of Ready, and should be estimated, before they come up in the Prioritized Product Backlog for addition in an upcoming Sprint.

8.4.3.2 Personas*

Personas are highly detailed fictional characters, representative of the majority of users and of other business stakeholders who may not directly use the end product. Personas are created to identify the needs of the target user base. Creating specific Personas can help the team better understand users and their requirements and goals. Based on a persona, the Product Owner can more effectively prioritize features to create the Prioritized Product Backlog.

Creating a Persona—This involves assigning a fictional name and preferably a picture, like a stock image, to the character. The Persona will include highly specific attributes such as age, gender, education, environment, interests, and goals. A quote illustrating the Persona's requirements can also be included. Below is an example of a Persona for a travel website.

Example: Vanessa is a 39-year-old resident of San Francisco. She is pursuing her passion for traveling after having a highly successful career as an attorney. Vanessa likes to have options while picking air travel and accommodations so that she can choose the best and the most affordable services. She gets easily frustrated with slow and cluttered websites.

8.4.3.3 Approved Changes

Unapproved Change Requests may be approved by the Product Owner during the *Develop Epic(s)* process, at times with suggestions and other input provided by relevant business stakeholders. Such changes are categorized as Approved Changes and can be prioritized and implemented in future Sprints. Change Requests and Approved Change Requests are discussed in sections 6.3.1 and 6.4.2.1.

8.4.3.4 Identified Risks

When creating Epics, new risks may be identified, and these identified risks form an important output of this process. Identified risks contribute to the development of the Prioritized Product Backlog (also be referred to as the Risk Adjusted Product Backlog). Risk Identification is described in section 7.4.1.

8.5 Create Prioritized Product Backlog

In this process, Epics are refined and elaborated, and most importantly, prioritized according to their respective business value to create a Prioritized Product Backlog for the project. Additionally, based on the Scrum Guidance Body recommendations, the Product Owner and the Scrum Team establish the Done Criteria for the project.

Figure 8-18 shows all the inputs, tools, and outputs for the *Create Prioritized Product Backlog* process.

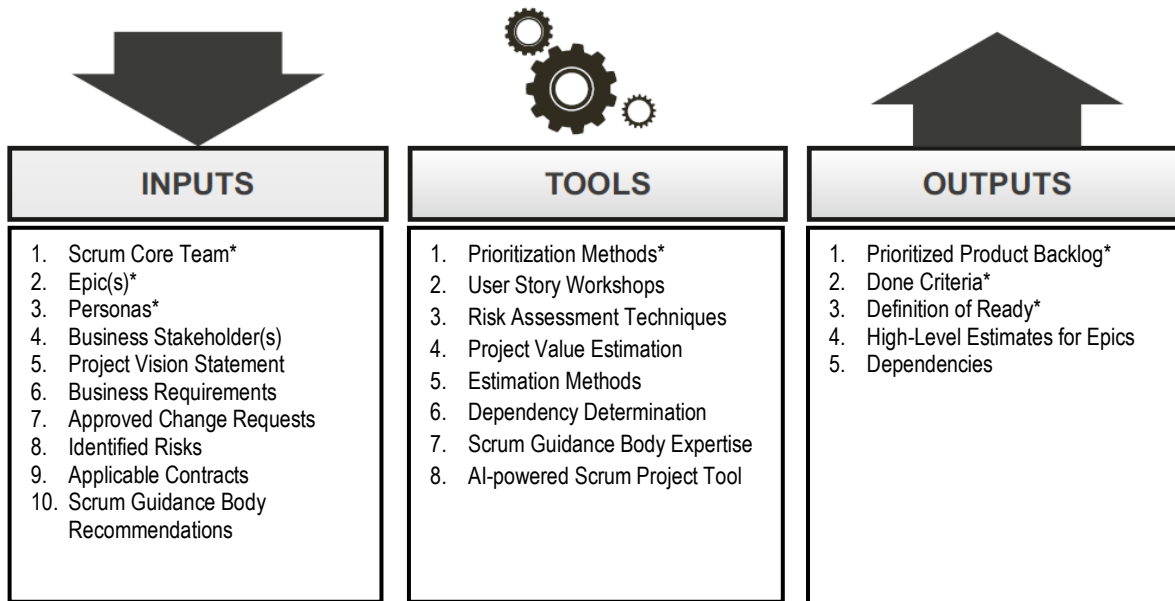


Figure 8-18: Create Prioritized Product Backlog—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 8-19 shows the data flow for product backlog creation. It maps how inputs are processed and refined into a structured backlog ready for release planning.

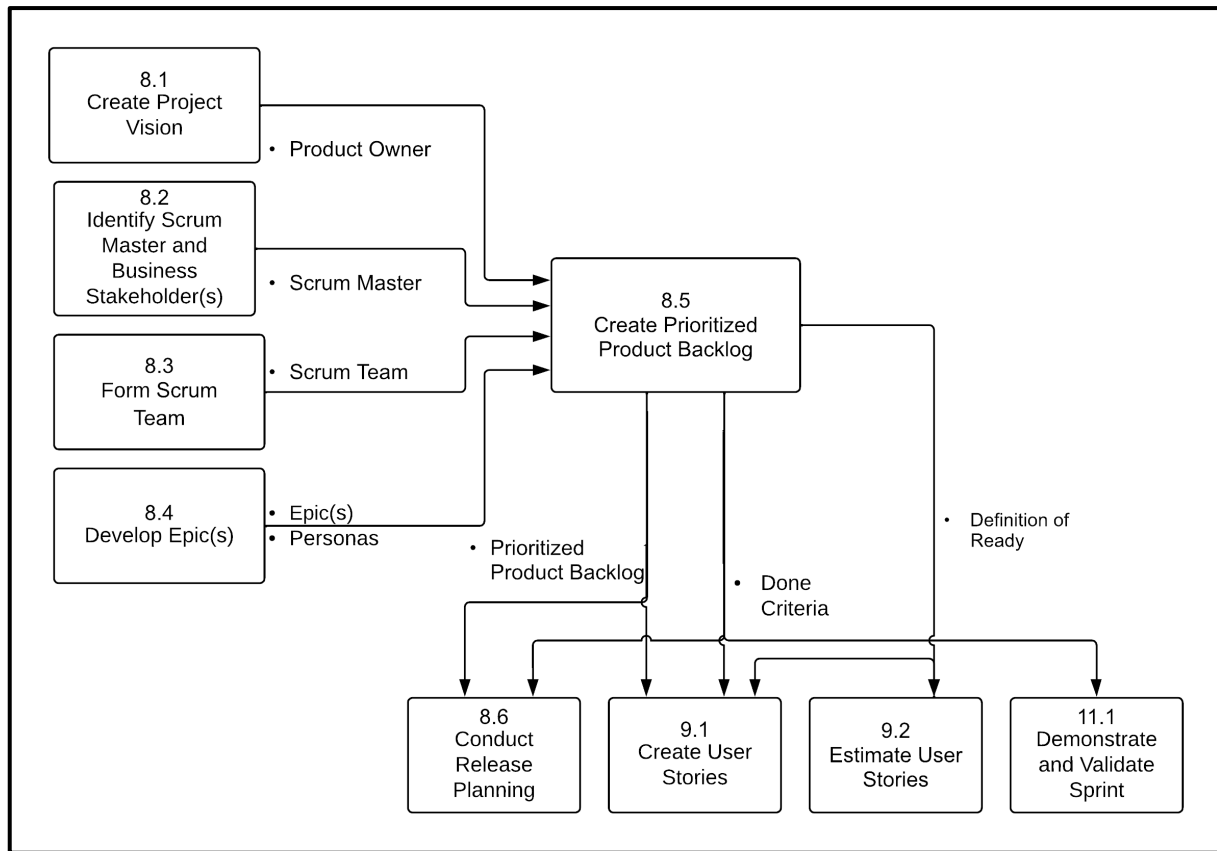


Figure 8-19: Create Prioritized Product Backlog—Data Flow Diagram

8.5.1 Inputs

8.5.1.1 Scrum Core Team*

Described in section 3.2.1.

8.5.1.2 Epic(s)*

Described in section 8.4.3.1.

8.5.1.3 Personas*

Described in section 8.4.3.2.

8.5.1.4 Business Stakeholder(s)

Described in section 8.2.3.2.

8.5.1.5 Project Vision Statement

Described in section 8.1.3.2.

8.5.1.6 Business Requirements

The insights gained through various tools such as user or customer interviews, questionnaires, JAD sessions, Gap Analysis, SWOT Analysis, and other meetings, help develop a better perspective about the business requirements and aids in creating the Prioritized Product Backlog.

8.5.1.7 Approved Change Requests

Described in section 8.4.3.3.

8.5.1.8 Identified Risks

Described in section 8.4.3.4.

8.5.1.9 Applicable Contracts

Described in section 8.4.1.7.

8.5.1.10 Scrum Guidance Body Recommendations

While creating the Prioritized Product Backlog, Scrum Guidance Body recommendations may include information on rules, regulations, standards, and best practices for developing the Prioritized Product Backlog. The Scrum Guidance Body may also provide recommendations for the Definition of Ready (the criteria that a User Story has to satisfy before being considered for estimation or inclusion into a Sprint) and the Definition of Done (the set of rules that are applicable to the User Stories in a given Sprint). Further details about Scrum Guidance Body recommendations can be found in section 8.1.1.7.

8.5.2 Tools

8.5.2.1 Prioritization Methods*

Some techniques used to prioritize the Epics, User Stories, or requirements in the Prioritized Product Backlog, based on business value are presented below:

- **MoSCoW Prioritization Scheme**—The MoSCoW prioritization scheme derives its name from the first letters of the phrases “Must have,” “Should have,” “Could have,” and “Won’t have.” This prioritization method is more effective than simple schemes. The labels are in decreasing order of priority with “Must have” features or functionalities being those without which the product will have no value and “Won’t have” functionalities being those that, although would be nice to have, are not necessary to be included.
- **Paired Comparison**—In this technique, a list of all the Epics/User Stories in the Prioritized Product Backlog is prepared. Next, each Epic/User Story is taken individually and compared with the others in the list, one at a time. Each time two Epics or User Stories are compared, a decision is made regarding which of the two is more important. Through this process, a prioritized list of Epics or User Stories can be generated.
- **100-Point Method**—Dean Leffingwell and Don Widrig (2003) developed the 100-Point Method. It involves giving the customers 100 points they can use to vote for the Epics or User Stories that are most important. The objective is to give more weight to the Epics/User Stories that are of higher priority when compared to the others. Each group member allocates points to the various User Stories, giving more points to those they feel are more important. On completion of the voting process, prioritization is determined by calculating the total points allocated to each Epic or User Story.
- **Kano Analysis**—Described in section 4.5.2.

8.5.2.2 User Story Workshops

Described in section 8.4.2.2.

8.5.2.3 Risk Assessment Techniques

Described in section 7.4.2.1.

8.5.2.4 Project Value Estimation

Described in section 4.5.1.

8.5.2.5 Estimation Methods

It might be useful to formulate some very high-level time estimates for Epics as these estimates can help the Product Owner plan appropriately for releases and can also aid in the prioritization process. However, since Epics planned at this stage are typically not yet broken down into well-defined User Stories, and there is also a possibility that some of the initially developed Epics may not be implemented, the team should not spend too much time at this point on estimating Epics.

This tool is used differently in the *Estimate User Stories* process (as described in section 9.2.2.1), where the Scrum Team is very involved with estimating well-defined User Stories. The Product Owner, Scrum Master, and relevant business stakeholders should understand that any high-level estimates provided in this process are used for guidance only, and that there may be significant deviations from the Scrum Team estimates for User Stories, once User Stories are properly defined and estimated.

Some specific tools that can be used to provide high-level estimates of Epics are:

- **Pre-existing Epic Estimates**—If similar Epics were developed in the past for the same project or other related projects, these prior estimates can be used to derive detailed estimates in this process.
- **Expert Opinion**—Experts who have implemented similar project functionalities in the past may be able to provide high-level effort estimates for Epics.
- **5-10-Minute Estimates**—Relevant team members can be provided with an overview of the general requirements for each Epic and asked to provide a rough estimate (e.g., in hours, days, or weeks) within five to ten minutes. This estimation method should be an option when only a small number of Epics need to be estimated, otherwise it could become time-consuming.
- **Affinity Estimation**—Affinity Estimation, such as T-shirt sizing, is a technique used to quickly create a time estimate for a large number of Epics. This is described in detail in section 9.2.2.1.

8.5.2.6 Dependency Determination

Properly documenting dependencies helps the Scrum Team to determine the relative order in which Epics (and User Stories) should be executed to create the Sprint deliverables. Dependencies highlight the relationship and interaction between Epics (and between User Stories), both within the Scrum Team working on a given Sprint and with other Scrum Teams working on the project. As the Prioritized Product Backlog is created, the Product Owner identifies any dependencies and relationships between Epics (and between User Stories), including any technical dependencies and dependencies related to the availability of people, as these dependencies will impact the order and priority of work to be done on the project.

Dependencies can be mandatory or discretionary, internal, or external, or some combination of these. For example, a dependency may be both mandatory and external to the project.

The following is a description of each type of dependency:

- **Mandatory Dependencies**—Mandatory dependencies are dependencies that dictate the sequence of the work to be completed in the project and are therefore important to consider when initially creating the Prioritized Product Backlog. These dependencies may be mandatory because of the nature of the work (such as a physical limitation on work sequence) or may exist due to contractual obligations or legal requirements. Mandatory dependencies are also commonly described as hard logic.
- **Discretionary Dependencies**—Discretionary dependencies are dependencies that are placed into the workflow based on past experiences or best practices in a particular field or domain. For example, the team may decide to complete one Epic (or User Story) before another because this flow of completing the work has worked well on past projects.
- **External Dependencies**—External dependencies are dependencies pertaining to activities or factors that are outside the scope of the work to be executed by the Scrum Team but are needed to complete a project task or create a project deliverable. External dependencies are usually outside the Scrum Team's control and therefore can produce greater risk to a project.
- **Internal Dependencies**—Internal dependencies are those factors within the project that affect the sequence of work to be done. These factors are usually under the control of the Scrum Team.

There are diverse ways to identify, define, and present project tasks and their dependencies.

8.5.2.7 Scrum Guidance Body Expertise

While creating the Prioritized Product Backlog, Scrum Guidance Body expertise could relate to documented rules and regulations or standards and best practices for prioritizing the product backlog. There may also be a team of subject matter experts who are available to assist the Product Owner in this process. This team could include business analysts, lead architects, senior developers, Scrum experts, and/or other experienced people. This expert group is usually not the same team that will stay on and work on this project, as they tend to move from project to project during the 'selling phase' or 'phase zero' which involves customers and/or users. For more information on the Scrum Guidance Body see section 8.4.2.7.

8.5.2.8 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 8-20 depicts the Vabro tool interface used to create a prioritized product backlog. It supports task hierarchy, backlog grooming, and estimation tracking.

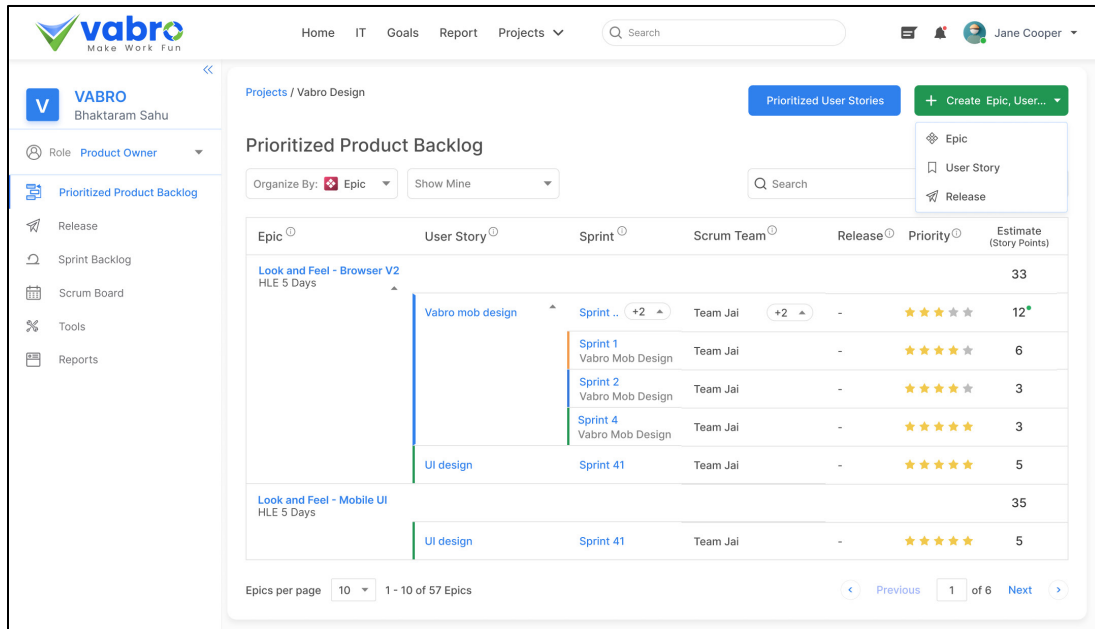


Figure 8-20: Creation of Prioritized Product Backlog (Source: Vabro)

Figure 8-21 shows how Jira manages the prioritized product backlog. It enables users to organize, prioritize, and visualize user stories and epics for development.

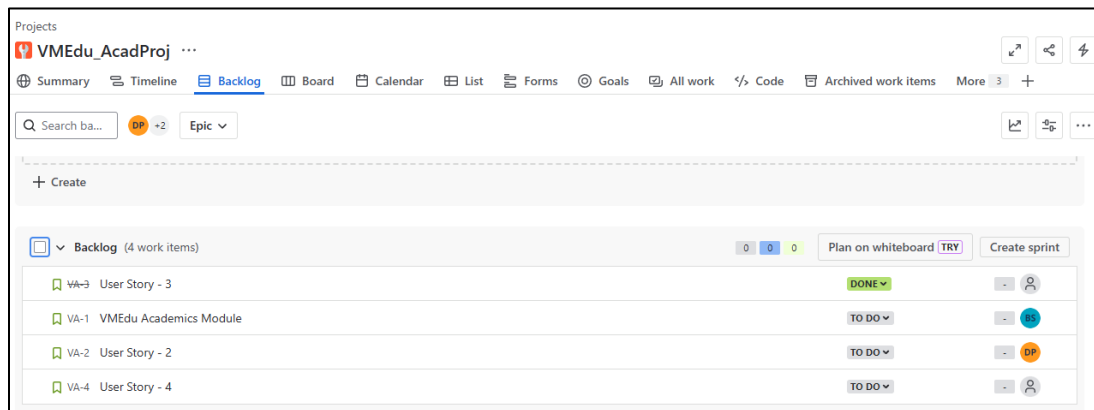


Figure 8-21: Use of Prioritized Product Backlog (Source: Jira)

8.5.3 Outputs

8.5.3.1 Prioritized Product Backlog*

The Product Owner develops a Prioritized Product Backlog which contains a prioritized list of business and project requirements written in the form of Epics, which are high-level User Stories. The Prioritized Product Backlog is based primarily on three factors—value, risk or uncertainty, and dependencies. It may also be referred to as the Risk Adjusted Product Backlog since it includes identified and assessed risks related to the project.

The Prioritized Product Backlog also encompasses all the approved changes that can be appropriately prioritized (as described in section 6.3.1).

Below is a description of the key factors that are considered when prioritizing items in the Prioritized Product Backlog.

- **Value**—The Product Owner is responsible to ensure the delivery of Epics or those product increments or functionalities that provide the highest level of business value first. Even an extremely valuable product increment may not be part of the first release if there are other product increments of even higher value that are sufficient for a first release.
- **Risk and Uncertainty**—The more uncertainty that exists, the riskier a project will be. Therefore, it is important that riskier Epics in the Prioritized Product Backlog are given higher priority. Requirements carrying a higher level of risk will require risk mitigation actions. When risks and their corresponding risk mitigation actions are considered and prioritized against other backlog items, the result is a Risk Adjusted Product Backlog. Dealing with risks early in the project does not guarantee that a project will be successful, but it does enhance the team's ability to successfully deal with those risks. The topic of risks is further described in section 7.4.3.
- **Dependencies**—Most projects will inherently have dependencies between some of the Epics or User Stories. These dependencies should be taken into consideration while creating the Prioritized Product Backlog. Functional requirements often depend on other functional and even non-functional requirements. These dependencies can impact how the Epics (and User Stories) in the Prioritized Product Backlog are prioritized. Two of the most common ways to resolve dependencies are to either split a single Epic (or User Story) into multiple Epics (or User Stories) or combine the interdependent portions.
- **Estimates**—High-level estimates for Epics generated from the estimation methods are also available in the Prioritized Product Backlog.

It is important to note that the prioritization of Epics may be different from that of its underlying User Stories. For example, even if an Epic is categorized as high priority, some User Stories contained in the Epic may be categorized as low priority while other User Stories in the Epic may be categorized as high priority.

8.5.3.2 Done Criteria*

The Done Criteria are a set of rules that are applicable to all User Stories. A clear definition of Done is critical because it removes ambiguity from requirements and helps the team adhere to mandatory quality norms. This definition is used to create the agreed-upon Done Criteria that will be used to determine when User Stories are complete. A User Story is considered Done when it is demonstrated to and approved by the Product Owner who judges it on the basis of the Done Criteria and the User Story Acceptance Criteria.

The Done Criteria may initially be determined and documented by the Scrum Guidance Body. However, there are typically project-specific Done Criteria that need to be incorporated during this process. Further details on the Done Criteria can be found in section 5.4.3.

8.5.3.3 Definition of Ready*

The Definition of Ready defines the criteria that a User Story must satisfy before being considered for estimation and inclusion into a Sprint. The Definition of Ready puts the onus on the Product Owner to properly define requirements for each User Story. Without properly defined requirements, it will be impossible to get reliable estimates, and the Scrum Team will not be able to effectively complete the required project work.

The Definition of Ready may initially be determined and documented by the Scrum Guidance Body. However, there are typically project-specific criteria that need to be incorporated during this process. Further details on the Definition of Ready can be found in section 5.4.2.

8.5.3.4 High-Level Estimates for Epics

High-level estimates for Epics may be initiated by the Product Owner using various estimation methods (as described in section 8.5.2.5). These estimates will help the Product Owner get an approximate idea of how much time and effort it will take to complete each Epic, which in turn will help with the prioritization of Epics in the Prioritized Product Backlog and plan project releases.

8.5.3.5 Dependencies

Dependencies describe the relationship and interaction between different Epics (or User Stories) in a project. Dependencies can be classified as mandatory or discretionary; internal or external; or some combination of these (as discussed in section 8.5.2.6). Dependencies will affect the relative order in which Epics (and User Stories) will be executed to create the Sprint deliverables, and will therefore affect their priority, as documented in the Prioritized Product Backlog.

8.6 Conduct Release Planning

In this process, the Product Owner, with inputs from business stakeholders and members of the Scrum Team, develops the initial Release Planning Schedule, which is communicated to, and shared with, all business stakeholders and Scrum Team Members. It is understood that the iterative nature of Scrum may necessitate future adjustments to the release schedule. The length of each Sprint is also determined in this process.

Figure 8-22 shows all the inputs, tools, and outputs for the *Conduct Release Planning* process.

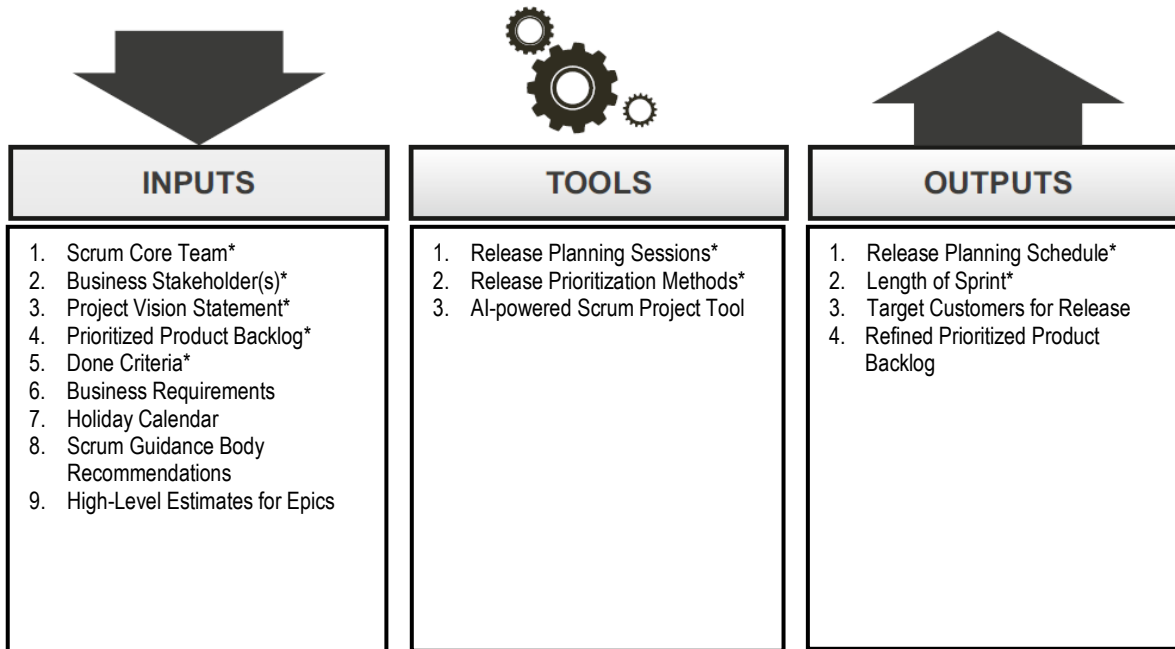


Figure 8-22: Conduct Release Planning—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 8-23 illustrates the data flow during release planning. It shows how backlog items and estimates are translated into detailed release schedules and goals.

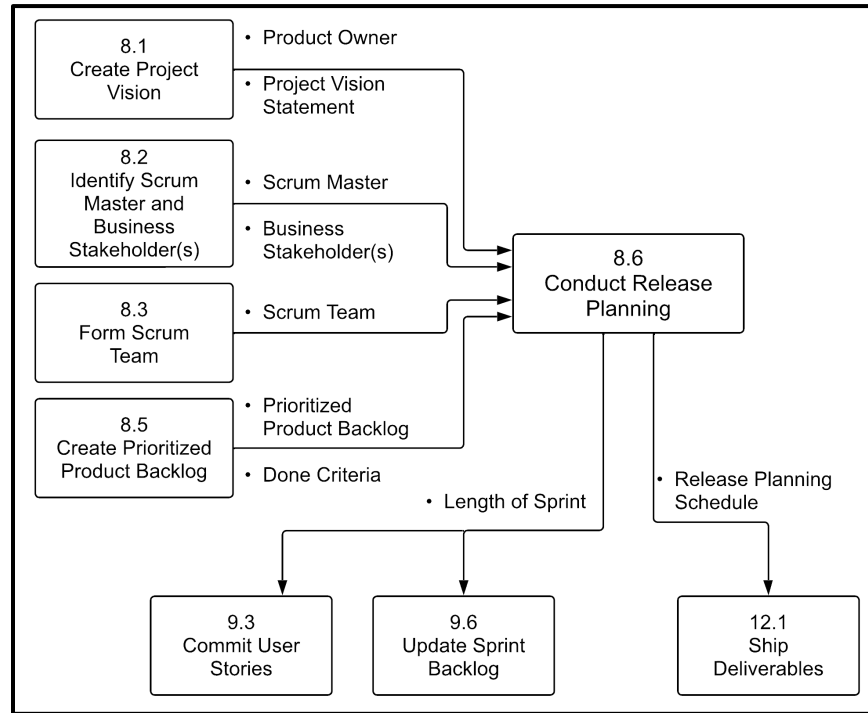


Figure 8-23: Conduct Release Planning—Data Flow Diagram

8.6.1 Inputs

8.6.1.1 Scrum Core Team*

Described in section 3.2.1.

8.6.1.2 Business Stakeholder(s)*

Described in section 8.2.3.2.

8.6.1.3 Project Vision Statement*

Described in section 8.1.3.2.

8.6.1.4 Prioritized Product Backlog*

Described in section 8.5.3.1.

8.6.1.5 Done Criteria*

Described in section 8.5.3.2.

8.6.1.6 Business Requirements

Described in section 8.5.1.6.

8.6.1.7 Holiday Calendar

It is important for the Scrum Team to keep track of key dates and availability of each team member. This can be accomplished through the use of a shared calendar that provides information on official holidays, leaves, travel plans, events, and so on that could impact project scheduling. This calendar will help the team in planning and executing Sprints.

8

8.6.1.8 Scrum Guidance Body Recommendations

In the *Conduct Release Planning* process, Scrum Guidance Body recommendations may pertain to rules, regulations, standards, and best practices for developing the Release Planning Schedule. The Scrum Guidance Body may be the best authority for defining guidelines related to business value, release expectations, deployment strategies, quality, and security. For more information on the Scrum Guidance Body see section 3.2.2.

8.6.1.9 High-Level Estimates for Epics

In release planning sessions, high-level estimates for Epics are used to plan releases using either a schedule-driven approach or a feature-driven approach. For more information on these estimates, see sections 8.5.2.5 and 8.5.3.4.

8.6.2 Tools

8.6.2.1 Release Planning Sessions*

Release planning sessions are conducted with the goal of developing a Release Planning Schedule for the project. This schedule defines when various sets of usable functionality or product increments will be delivered to the customer. A major objective of a Release Planning Meeting or session is to enable the Scrum Team to have an overview of the planned releases and delivery schedule for the product they are developing. This needs to align with the expectations of the Product Owner and relevant business stakeholders (primarily the project sponsor). Many organizations have a strategy regarding the release of projects. Some organizations prefer continuous deployment, where there is a release after the creation of specified usable product functionality.

Other organizations prefer phased deployment, where releases are made at predefined intervals. Depending on the organization's strategy, these planning sessions may be driven by functionality—in which the objective is to deliver working deliverables (of product increments) once a set of predetermined functionalities has been developed—or by date, where releases occur on predefined dates. Since the Scrum framework promotes information-based iterative-decision-making over the detailed upfront planning practiced in traditional waterfall-style project management, release planning sessions need not produce a detailed Release Planning Schedule for the entire project. The plan is expected to be updated continually as relevant and more detailed information becomes available.

8.6.2.2 Release Prioritization Methods*

Release prioritization methods are used to develop the Release Planning Schedule. These methods are usually industry- or organization-specific and are usually determined by the organization's senior management.

8.6.2.3 AI-powered Scrum Project Tool

Described in section 2.4.4.

Figure 8-24 shows release planning in Vabro. It provides visual tools for scheduling, tracking dependencies, and aligning releases with business priorities.

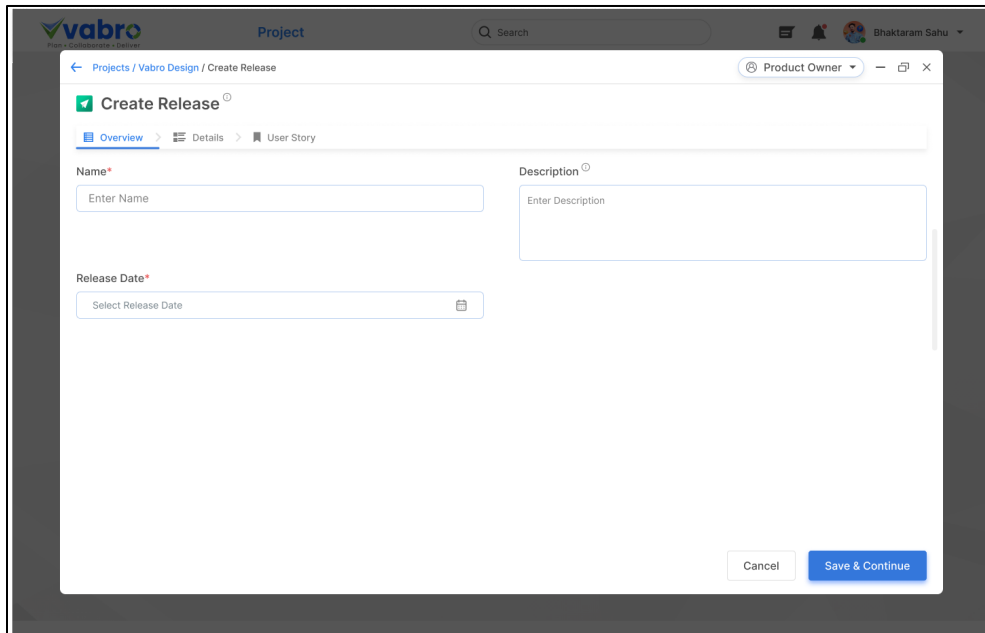
The screenshot shows a web application interface for Vabro. At the top, there's a navigation bar with the Vabro logo, a 'Project' tab, a search bar, and a user profile 'Bhaktaram Sahu'. Below this, a breadcrumb trail reads 'Projects / Vabro Design / Create Release'. The main content area is titled 'Create Release' with a green checkmark icon. It has three tabs: 'Overview' (selected), 'Details', and 'User Story'. The 'Overview' tab contains three input fields: 'Name*' with a placeholder 'Enter Name', 'Description*' with a placeholder 'Enter Description', and 'Release Date*' with a placeholder 'Select Release Date' and a calendar icon. At the bottom right of the form are two buttons: 'Cancel' and 'Save & Continue'.

Figure 8-24: Release Planning in Scrum (Source: Vabro)

8

Figure 8-25 depicts release and configuration management in Jira. It highlights features for managing software versions, release readiness, and deployment tracking.

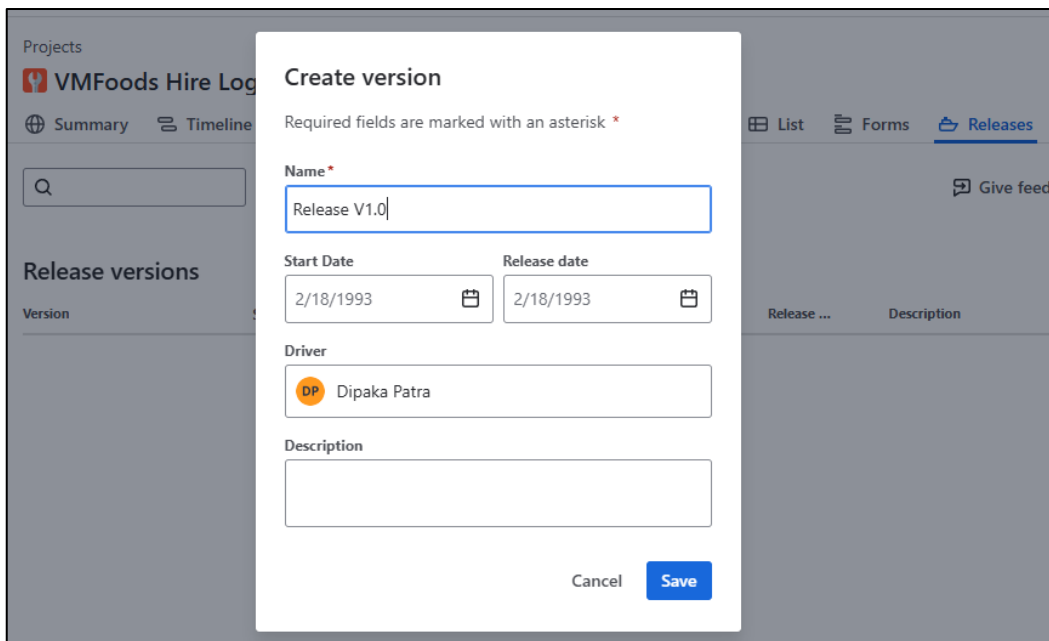
The screenshot shows a Jira interface with a 'Create version' modal open. The modal has a title 'Create version' and a note 'Required fields are marked with an asterisk *'. It contains several fields: 'Name*' with the value 'Release V1.0', 'Start Date' with the value '2/18/1993' and a calendar icon, 'Release date' with the value '2/18/1993' and a calendar icon, 'Driver' with a dropdown showing 'DP Dipaka Patra', and 'Description' with an empty text area. At the bottom of the modal are 'Cancel' and 'Save' buttons. In the background, the Jira interface shows a sidebar with 'Projects' and 'VMFoods Hire Log', a search bar, and a 'Release versions' section. The main content area has tabs for 'List', 'Forms', and 'Releases' (selected), and a 'Give feed' button.

Figure 8-25: Release and Configuration Management in Scrum (Source: Jira)

8.6.3 Outputs

8.6.3.1 Release Planning Schedule*

A Release Planning Schedule is a key output of the *Conduct Release Planning* process. The Release Planning Schedule states which deliverables are to be released to the customers, along with planned intervals, and release dates. A release typically includes a group of User Stories in the Prioritized Product Backlog, which should be completed and shipped together as part of the release. It is important to note, however, that User Stories in a release do not always provide the complete functionality of Epics. Some Epics may only be partially completed since not all User Stories in an Epic are necessarily of a high enough priority to be part of that particular release. There may not be a release scheduled at the end of every Sprint iteration. At times, a release may be planned after a group of Sprint iterations are completed.

Depending on the organization's strategy, release planning sessions may be driven by functionality—in which the objective is to deliver working deliverables once a set of predetermined functionalities has been developed—or by date, where releases occur on predefined dates. Each deliverable should only be released when it offers sufficient business value to the customer. During the *Conduct Release Planning* process, it may be useful to consider the Sprint Length and the assumptions for Team Velocity in order to derive a better Release Planning Schedule.

8.6.3.2 Length of Sprint*

Based on the various inputs, the business requirements, and details of the Release Planning Schedule, the Product Owner and the Scrum Team decide on the appropriate Sprint length for the project. Once determined, the length of each Sprint typically remains the same throughout the entire project. However, the duration of Sprints can be changed if the Product Owner and the Scrum Team have justification for the change. For example, early in the project, the team may still be experimenting to find the best Sprint length. Later in the project, a reduction in the Sprint length can occur due to improvements in the project environment. To get maximum benefits from a Scrum project and to provide maximum flexibility for change, the length of a Sprint should be as short as possible. A Sprint is typically Time-boxed with a duration of one to four weeks.

Most Scrum projects typically have Sprints Time-boxed at a duration of two or three weeks. However, for projects with very stable requirements, Sprints can extend to six weeks, if justified. Because changes are not allowed during a Sprint, the impact and frequency of any changes expected may have an impact on the decision related to the Length of Sprint. The impact of expected change on the Length of Sprint is described in section 6.5.1.

8.6.3.3 Target Customers for Release

Not every release will target all business stakeholders or users. The business stakeholder(s) may choose to limit certain releases to a subset of customers and/or users. The Release Planning Schedule should specify the target customers and/or users for each release.

8.6.3.4 Refined Prioritized Product Backlog

The Prioritized Product Backlog, initially developed in the *Create Prioritized Product Backlog* process, may be refined in this process. For example, there may be additional clarity about the User Stories after the Scrum Core Team conducts release planning sessions with business stakeholders.

8.7 Initiate Phase Data Flow Diagram

Figure 8-26 illustrates the data flow for the Initiate phase. It outlines how stakeholder inputs and tools combine to produce critical outputs for initiating a Scrum project.

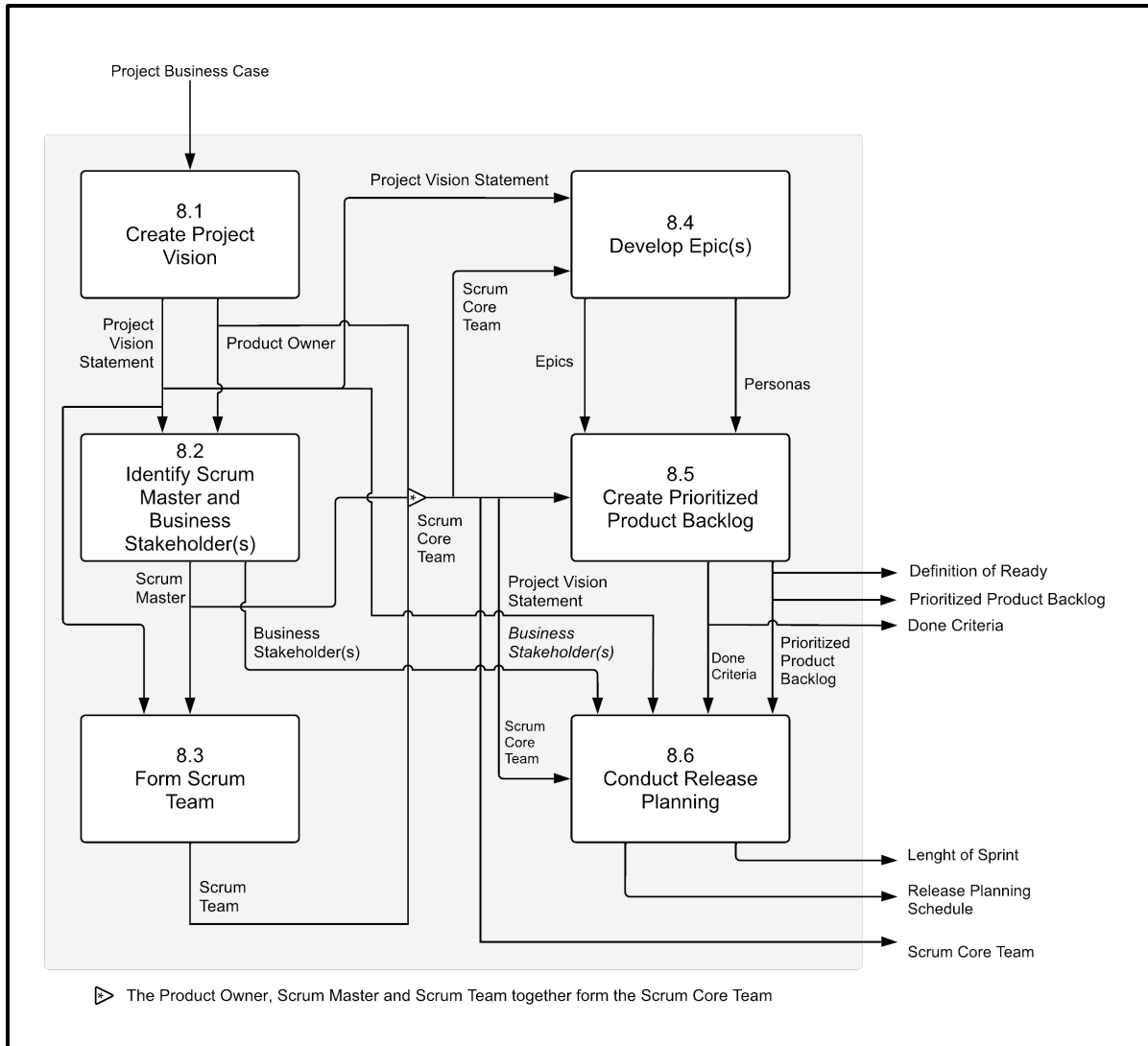


Figure 8-26: Initiate Phase—Data Flow Diagram

9. PLAN AND ESTIMATE

The Plan and Estimate phase consists of processes related to planning and estimating User Stories and associated tasks, which include *Create User Stories*, *Estimate User Stories*, *Commit User Stories*, *Identify Tasks*, *Estimate Tasks*, and *Update Sprint Backlog*.

Plan and Estimate, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Product Owners, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter.

This chapter is written from the perspective of one Scrum Team working on one Sprint to produce potentially shippable deliverables, which could be part of a larger project, program, or portfolio. Additional information pertaining to Scaling Scrum for Large Projects is available in chapter 13. Additional information pertaining to Scaling Scrum for the Enterprise can be found in chapter 14.

After the Initiate Phase is completed, the iterative Sprint cycles can commence. Plan and Estimate is the first of three phases that are done repetitively, in every Sprint cycle.

At the beginning of a Sprint, the Product Owner and Scrum Team, facilitated by the Scrum Master, plan the Sprint. The Product Owner refines the highest-priority Epics into a set of well written and estimated User Stories, which the Scrum Team commits to completing in the upcoming Sprint based on team velocity assumptions. The Scrum Master and the Scrum Team create and updates the Sprint Backlog with the list of User Stories committed to be delivered as part of the Sprint.

The Scrum Team then plans its work in more detail by identifying and optionally estimating the tasks it must complete in order to deliver the User Stories for the Sprint. As a final planning step for the Sprint, the team completes the Sprint Backlog with details of tasks and, if available, their estimates. The Sprint Backlog will be used in the Implement phase to track the team's progress during the Sprint.

It is also important to realize that although all phases and processes are defined uniquely in the SBOK® Guide, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to overlap some phases and/or processes, depending on the specific requirements of each project.

Figure 9-1 provides an overview of the Plan and Estimate phase processes, which are as follows:

9.1 Create User Stories—In this process, User Stories and their related Acceptance Criteria are created by the Product Owner (elaborated from the previously-defined Epics) and incorporated into the Prioritized Product Backlog. User Stories are designed to ensure that the customer's requirements are clearly depicted and can be fully understood by the business stakeholders. User Stories need to be tangible enough and must satisfy the Definition of Ready before they can be estimated and developed by the Scrum Team.

9.2 Estimate User Stories—In this process, the Scrum Team, supported by the Scrum Master, estimates the User Stories and identifies the effort required to develop the functionality described in each User Story. Only User Stories that satisfy the Definition of Ready and are properly defined by the Product Owner are estimated by the team.

9.3 Commit User Stories—In this process, the Scrum Team commits to delivering a set of User Stories for the Sprint. The decision on which User Stories will be committed to is based on the relative value-based priority of the User Stories and the estimated effort and team velocity for one Sprint. As part of this process, the Scrum Team starts the creation of the Sprint Backlog, which contains the committed User Stories that are assigned to a particular Sprint. The backlog is refined further with task-level details as Sprint Planning continues. With this commitment from the Scrum Team given at the beginning of a Sprint as part of Sprint Planning, the content of the Sprint is defined and cannot be changed once the Sprint implementation phase has begun.

9.4 Identify Tasks—In this process, the committed User Stories are decomposed into specific tasks and compiled into a task list. Identifying tasks could either be done at the beginning of the Sprint for all committed User Stories or before the team starts working on the tasks required for each User Story.

9.5 Estimate Tasks—This is an optional process which involves creating task estimates if the Scrum Team sees value in doing so. In this process, the Scrum Team estimates the effort required to accomplish each task in the Task List. Task estimates could either be determined at the beginning of the Sprint for all User Stories/tasks relevant to that Sprint, or for each task just before the team starts working on the particular User Story/task. The estimation can be done using the same methods that were used for the *Estimate User Stories* process.

9.6 Update Sprint Backlog—In this process, the Scrum Core Team updates the Sprint Backlog with task details and if available, the task estimates. The updated Sprint Backlog will be used in the Implement phase to track the team's progress during the upcoming Sprint.

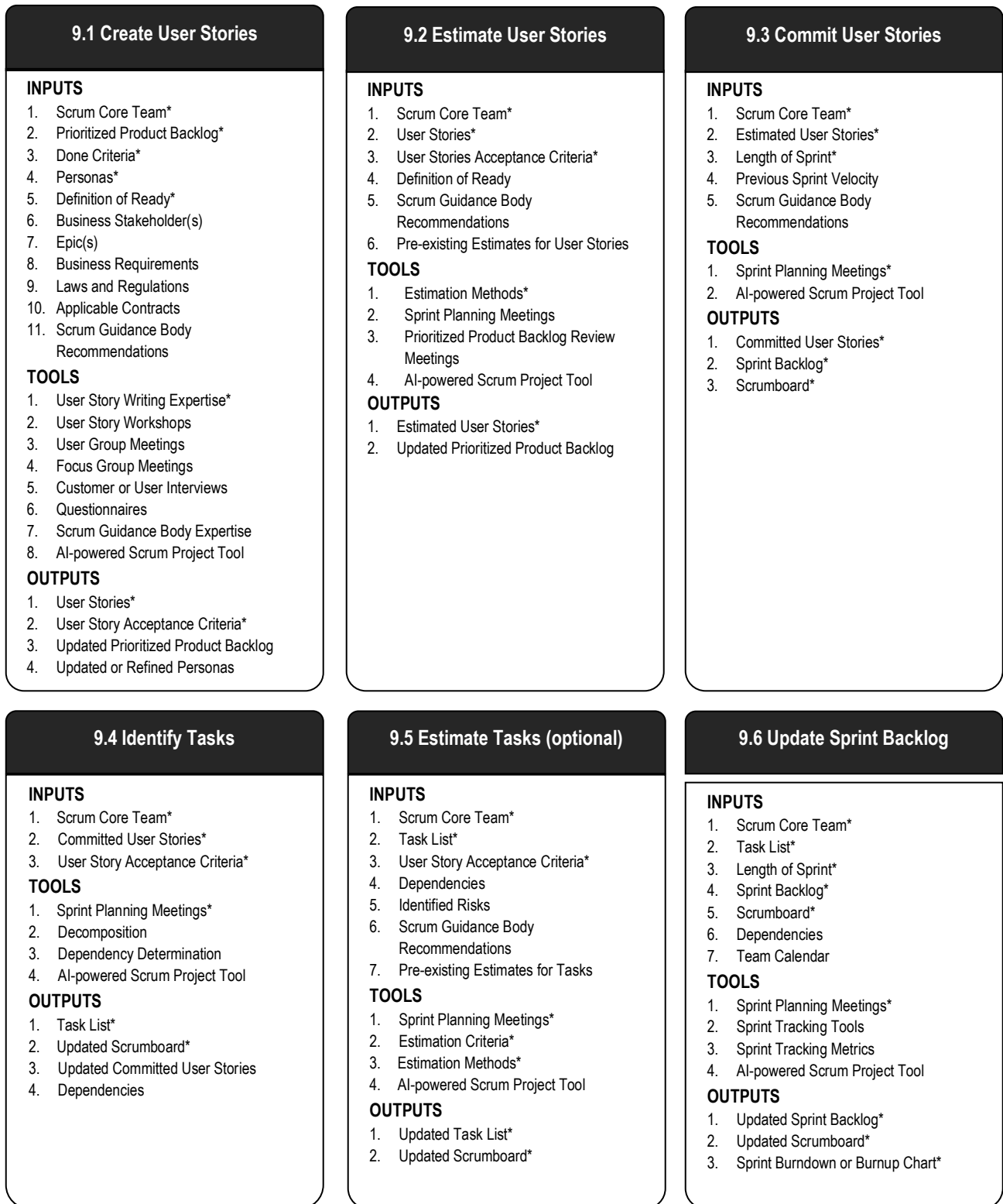


Figure 9-1: Plan and Estimate Overview

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-2 below shows the mandatory inputs, tools, and outputs for the processes in the Plan and Estimate phase.

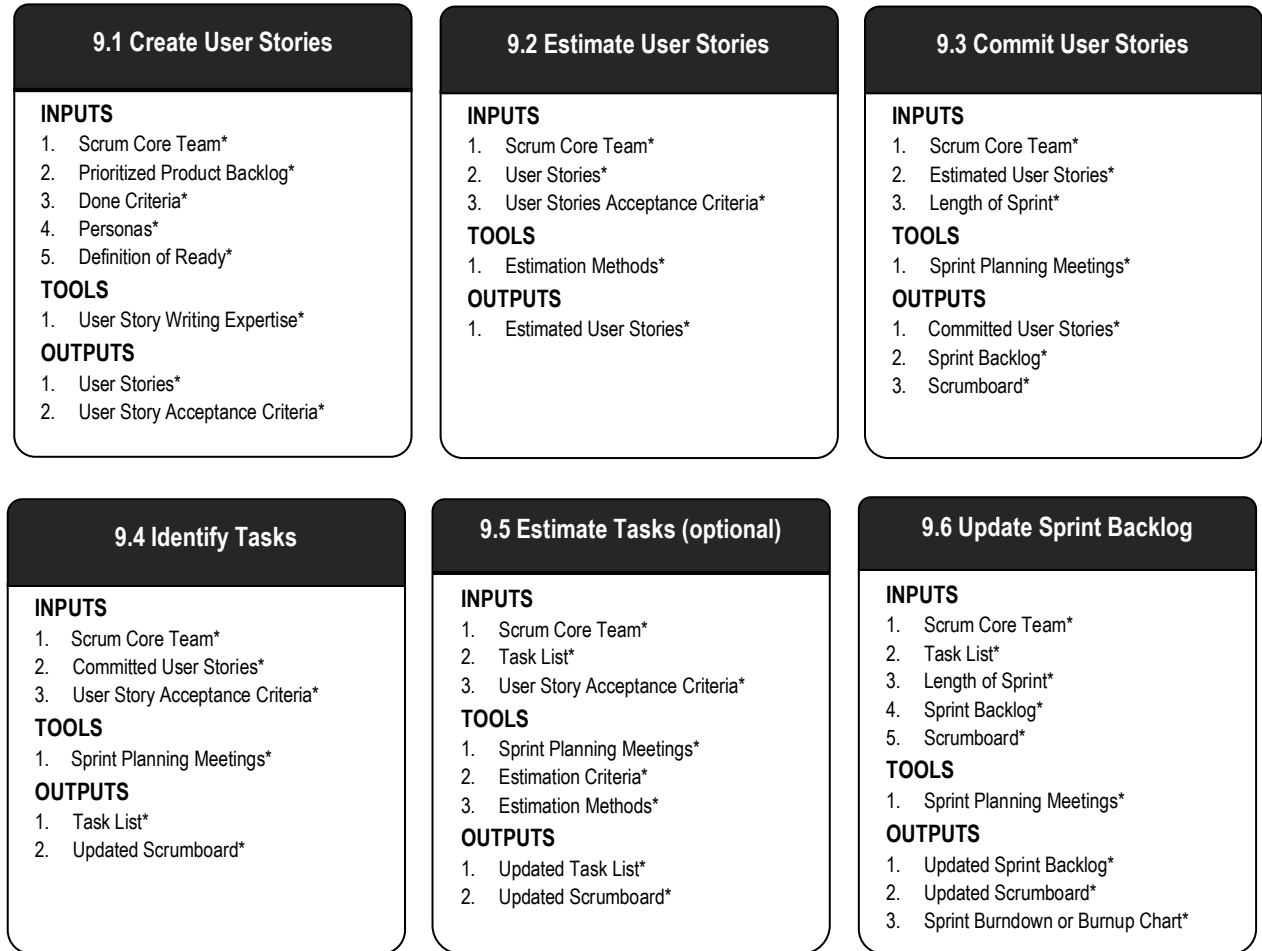


Figure 9-2: Plan and Estimate Overview (Essentials)

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

9.1 Create User Stories

In this process, User Stories and their related Acceptance Criteria are created by the Product Owner (elaborated from the previously defined Epics) and incorporated into the Prioritized Product Backlog. User Stories are designed to ensure that the customer's requirements are clearly depicted and can be fully understood by all the project stakeholders. User Stories need to be tangible enough and must satisfy the Definition of Ready before they can be estimated and developed by the Scrum Team.

User Story Workshops may be used to help the Scrum Team members better understand the User Stories created by the Product Owner.

Figure 9-3 shows all the inputs, tools, and outputs for the *Create User Stories* process.

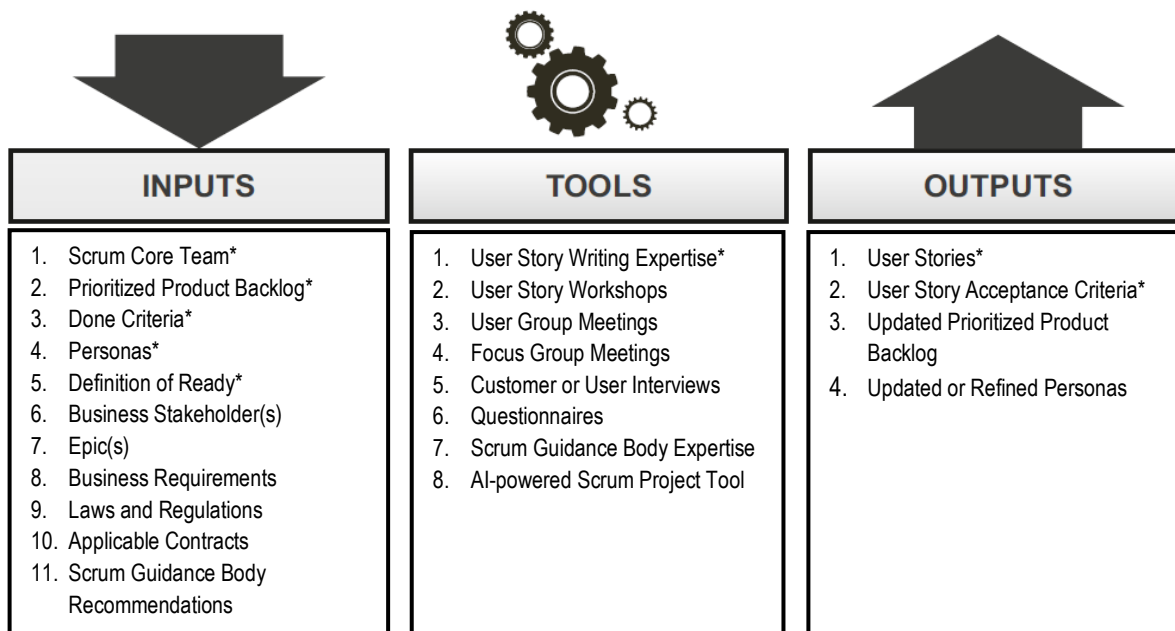


Figure 9-3: Create User Stories—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-4 illustrates the data flow for Create User Stories process. It maps how inputs such as business needs and personas are transformed into user stories and acceptance criteria.

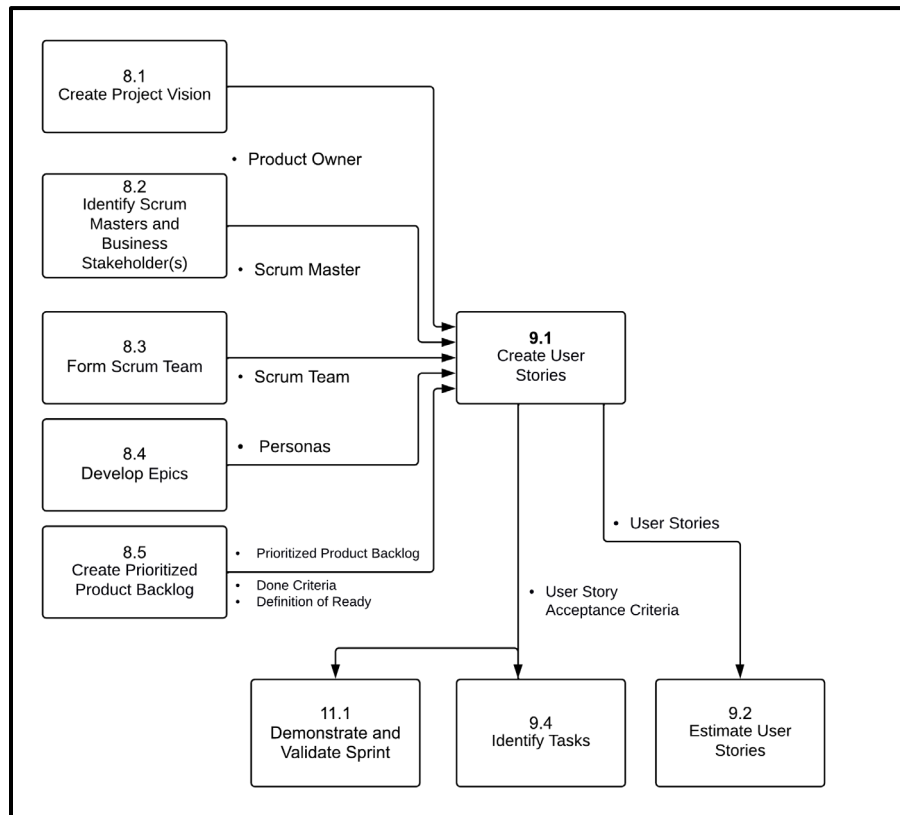


Figure 9-4: Create User Stories—Data Flow Diagram

9.1.1 Inputs

9.1.1.1 Scrum Core Team*

Described in section 3.2.1.

9.1.1.2 Prioritized Product Backlog*

Described in section 8.5.3.1.

9.1.1.3 Done Criteria*

Described in section 8.5.3.2.

9.1.1.4 Personas*

Described in section 8.4.3.2.

9.1.1.5 Definition of Ready*

The Definition of Ready defines the criteria that a User Story will have to satisfy before being considered for estimation or inclusion into a Sprint. For more information on the Definition of Ready see sections 5.3.2 and 8.5.3.3.

9.1.1.6 Business Stakeholder(s)

Described in section 8.2.3.2.

9.1.1.7 Epic(s)

Described in section 8.4.3.1.

9.1.1.8 Business Requirements

Described in section 8.5.1.6.

9.1.1.9 Laws and Regulations

Described in section 8.4.1.6.

9.1.1.10 Applicable Contracts

Described in section 8.4.1.7.

9.1.1.11 Scrum Guidance Body Recommendations

In the *Create User Stories* process, Scrum Guidance Body Recommendations may include information on rules, regulations, standards, and best practices required to create effective User Stories. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

9.1.2 Tools

9.1.2.1 User Story Writing Expertise*

The Product Owner, based on his or her interaction with the business stakeholders, business knowledge and expertise, and inputs from the team, develops the User Stories that will form the initial Prioritized Product Backlog for the project. The Prioritized Product Backlog represents the total sum of all the requirements that must be completed for the project. The objective of this exercise is to create elaborated and refined User Stories that can be estimated and committed to by the Scrum Team. At times, the Product Owner may bring a business analyst to assist with writing the User Stories. Although the Product Owner has the primary responsibility for writing User Stories and often carries out this exercise on his or her own, a User Story Writing Workshop can be held if desired.

9.1.2.2 User Story Workshops

Described in section 8.4.2.2.

9.1.2.3 User Group Meetings

Described in section 8.4.2.1.

9.1.2.4 Focus Group Meetings

Focus Group Meetings are a qualitative technique used to gauge and understand user needs and expectations about a proposed product. A small group of users are selected to form the focus group. This group may be selected randomly from a large pool of users or can be selected specifically to represent all the major personas being targeted. Focus Group Meetings normally adhere to a certain format in which the group is asked questions that they then discuss among themselves. Each Focus Group session can have its own rules for discussion as decided by the organizers. These meetings are usually held in the presence of a moderator.

9.1.2.5 Customer or User Interviews

Described in section 8.4.2.4.

9.1.2.6 Questionnaires

Described in section 8.4.2.5.

9.1.2.7 Scrum Guidance Body Expertise

While creating User Stories, the Scrum Guidance Body Expertise could refer to regulations, standards, and/or best practices for creating User Stories. There may also be a team of subject matter experts who are available to assist the Product Owner or provide guidance on how to create the User Stories. This team could include business analysts, lead architects, senior developers, Scrum experts, and other experienced people. This expert group is usually not the same team that will stay on and work on the project, as they tend to move from project to project and provide guidance to Scrum Teams when required. For more information on the Scrum Guidance Body see section 8.4.2.7.

9.1.2.8 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 9-5 displays the Vabro interface for adding user stories to the prioritized product backlog. It supports team collaboration and helps organize stories based on business value.

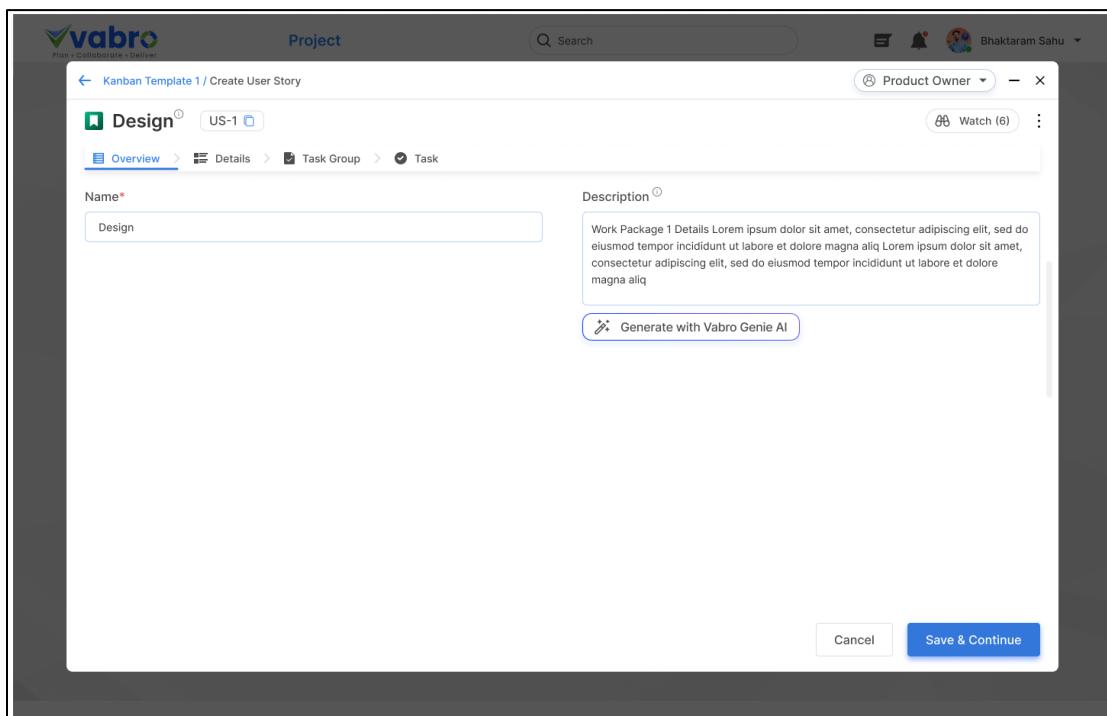


Figure 9-5: Adding User Stories in Prioritized Product Backlog (Source: Vabro)

Figure 9-6 shows Jira's interface for managing product backlog. It allows teams to add, prioritize, and track user stories within an agile framework.

The screenshot shows the 'Create' dialog in Jira. At the top, it says 'Create' with window control icons. Below that, a note states 'Required fields are marked with an asterisk *'. The 'Project' field is a dropdown menu showing 'VMedu_AcadProj (VA)'. The 'Work type' field is a dropdown menu showing 'Story', with a link 'Learn about work types' below it. The 'Status' section shows 'To Do' as the selected status, with a note 'This is the initial status upon creation'. The 'Summary' field is empty. At the bottom, there is a checkbox 'Create another' and two buttons: 'Cancel' and 'Create'.

Figure 9-6: Adding User Stories in Product Backlog (Source: Jira)

9.1.3 Outputs

9.1.3.1 User Stories*

User Stories adhere to a specific, predefined structure and are a simplistic way of documenting the requirements and desired end-user functionality of a product. A User Story describes three things about a requirement—Who, What, and Why. The requirements expressed in User Stories are short, simple, and easy-to-understand statements. The predefined standard format results in enhanced communication among the business stakeholders and better estimations by the team. Some Epics/User Stories may initially be too large to handle within a single Sprint. Once Epics move up in the Prioritized Product Backlog to be completed in an upcoming Sprint, they are further decomposed into smaller User Stories.

The Prioritized Product Backlog is a dynamic list that is continuously updated because of reprioritization and new, updated, refined, and sometimes, deleted User Stories. These updates to the Product Backlog are typically the result of changing business requirements. For more information on the Prioritized Product Backlog see section 8.5.3.1.

User Story Format:

As a <role/persona>, I should be able to <requirement> so that <benefit>.

User Story Example:

As a Database Administrator, I should be able to revert a selected number of database updates so that the desired version of the database is restored.

9.1.3.2 User Story Acceptance Criteria*

Every User Story has an associated Acceptance Criteria that is defined by the Product Owner and communicated to the Scrum Team. User Stories are subjective, so the Acceptance Criteria provide the objectivity required for the User Story to be considered as Done or not Done (i.e., accepted or rejected) during the *Demonstrate and Validate Sprint* process. Acceptance Criteria provide clarity to the team on what is expected of a User Story, remove ambiguity from requirements, and help in aligning expectations. During the Sprint Review Meeting, the Acceptance Criteria provide the context for the Product Owner to decide if a User Story has been completed satisfactorily. It is important and the responsibility of the Scrum Master to ensure that the Product Owner does not change the Acceptance Criteria of a committed User Story in the middle of a Sprint.

9.1.3.3 Updated Prioritized Product Backlog

The Prioritized Product Backlog created in the *Create Prioritized Product Backlog* process is updated with information on User Stories, Epics, estimates for User Stories, and the User Story Acceptance Criteria. For more information on the Prioritized Product Backlog, see section 8.5.3.1.

9.1.3.4 Updated or Refined Personas

Personas are initially created in the *Develop Epic(s)* process. While writing User Stories, the Scrum Team may come to a collective decision that some of those initial personas created are inadequate and need refinement. If refining personas is required, it is normally done near the end of the *Create User Stories* process. For more information on personas see section 8.4.3.2.

9.2 Estimate User Stories

In this process, the Scrum Team, supported by the Scrum Master, estimates the User Stories and identifies the effort required to develop the functionality described in each User Story. Only User Stories that satisfy the Definition of Ready and are properly defined by the Product Owner are estimated by the team.

The Product Owner does not play an active role in estimating User Stories, but may be consulted to clarify any questions the Scrum Team might have related to the User Stories being estimated.

Figure 9-7 shows all the inputs, tools, and outputs for the *Estimate User Stories* process.

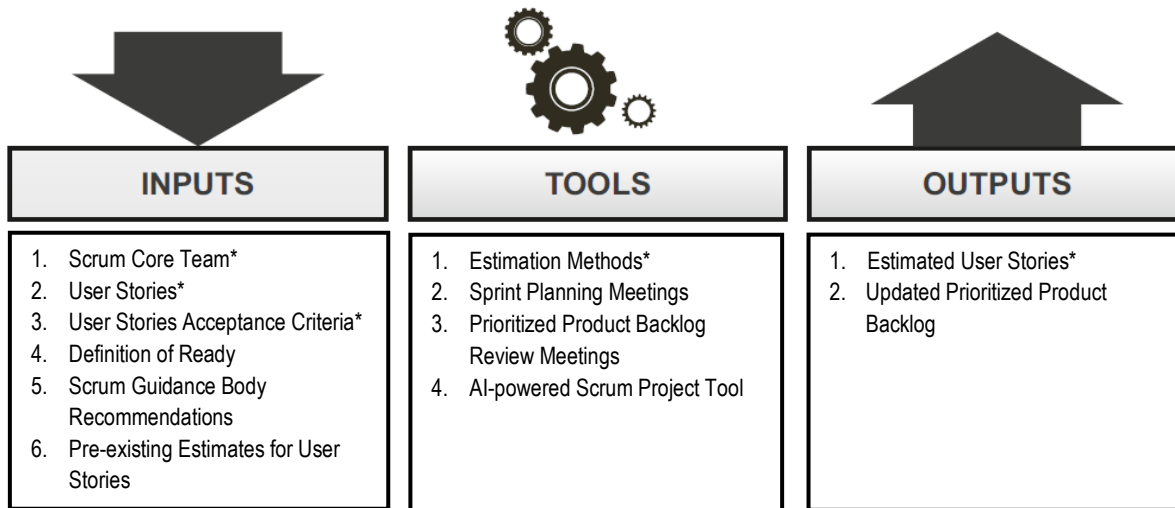


Figure 9-7: Estimate User Stories—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 9-8 explains the data flow for estimating user stories. It connects backlog items with planning tools to produce updated, estimated stories ready for commitment.

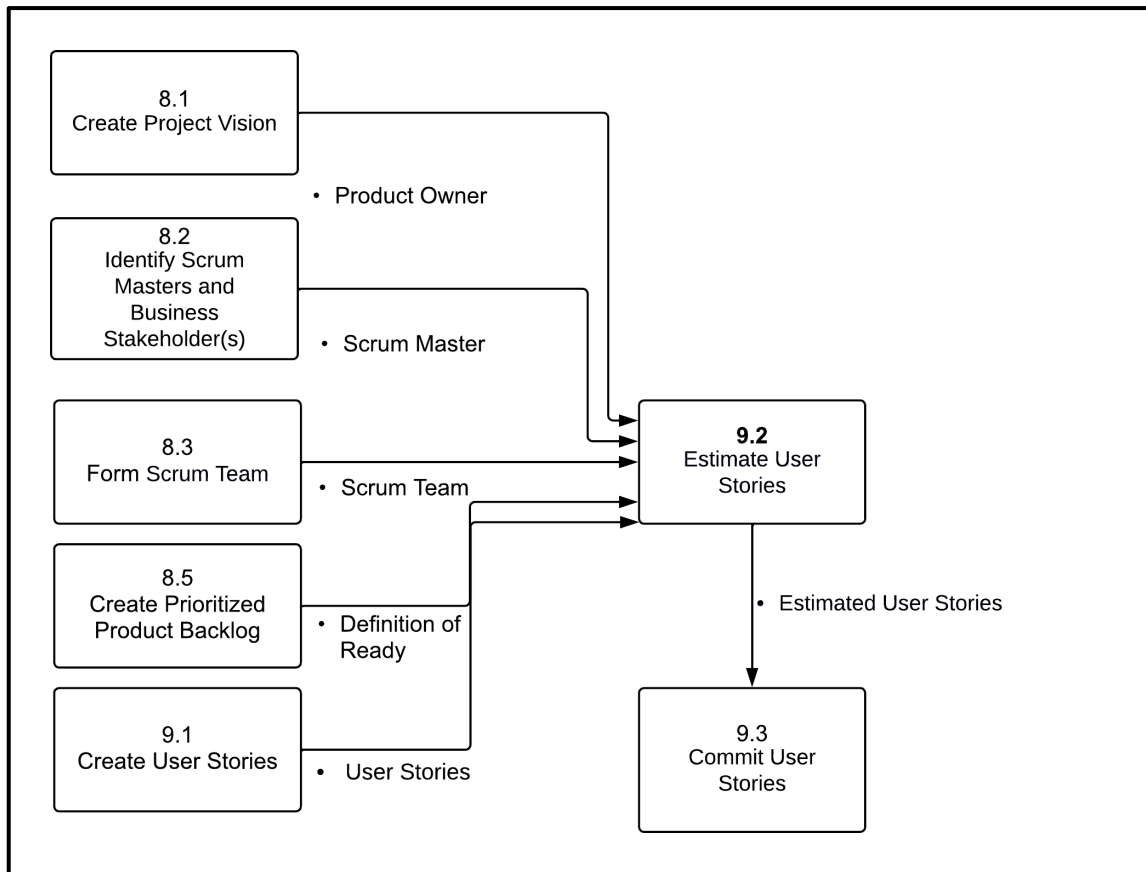


Figure 9-8: Estimate User Stories—Data Flow Diagram

9.2.1 Inputs

9.2.1.1 Scrum Core Team*

Described in section 3.2.1.

9.2.1.2 User Stories*

Described in section 9.1.3.1.

9.2.1.3 User Stories Acceptance Criteria*

Described in section 9.1.3.2.

9.2.1.4 Definition of Ready

The Definition of Ready defines the criteria that a User Story will have to satisfy before being considered for estimation or inclusion into a Sprint. For more information on the Definition of Ready see sections 5.3.2 and 8.5.3.3.

9.2.1.5 Scrum Guidance Body Recommendations

In the *Estimate User Stories* process, Scrum Guidance Body Recommendations may include information on the rules, regulations, standards, and best practices required to effectively estimate User Stories. For more information on the Scrum Guidance Body Recommendations, see section 8.1.1.7.

9.2.1.6 Pre-Existing Estimates for User Stories

Some pre-existing estimates of the effort needed to complete User Stories may already exist, particularly if similar User Stories were already completed in the same project or related past projects. The effort and time taken to complete similar User Stories can be used to derive estimates for the time needed to complete existing User Stories. Experts who have implemented similar project requirements in the past may also be able to provide some effort estimates for the User Stories. These pre-existing estimates will help the Scrum Team to thoroughly estimate and commit User Stories for the Sprint. It is important to ensure that the Scrum Team creates their own estimates of User Stories before the User Stories are committed as a part of the Sprint, instead of solely relying on any pre-existing estimates available for User Stories.

9.2.2 Tools

9.2.2.1 Estimation Methods*

As new or updated User Stories are refined in the Product Backlog, the Scrum Team will assign or update any pre-existing estimates for each User Story. Relative sizing or story points can be used for estimating the overall size of a User Story or feature. This approach assigns a story point value based on an overall assessment of the size of a User Story with consideration given to its associated risk, the amount of effort required, and the level of complexity. This assessment will be conducted by the Scrum Team and a story point value will be assigned. Once an evaluation is done on one User Story in the Prioritized Product Backlog, the Scrum Team can then evaluate other User Stories relative to that first story. It should be noted that the story point calibration for each team is different so the number of User Story points completed cannot be used for comparison across teams. Also, the estimation method selected depends on the level of estimation detail required by the team.

Some techniques that can be used to estimate User Stories are as follows:

1. Wideband Delphi

Wideband Delphi is a group-based estimation technique for determining how much work is involved and how long it will take to complete the work. Individuals within the team anonymously provide estimations for each item and the initial estimates are plotted on a chart. The team then discusses the factors that influenced their own estimates and proceed to a second round of estimation. This process is repeated until the individual estimates are close to each other and a consensus for the final estimate can be reached.

2. Planning Poker

Planning Poker, also called Estimation Poker, is a derivative of the Wideband Delphi technique that uses consensus to estimate the relative sizes of User Stories, or the effort required to create them.

In Planning Poker, each team member is assigned a deck of cards. Each card is numbered in sequence with each number representing the complexity of the User Story (or task) in terms of time or effort. The Scrum Team members assess the User Story (or task) to better understand it. Each member then picks a card from his/her deck that represents his/her estimate for the time or effort required to complete the User Story (or task). If the majority or all team members select the same card, then the estimate indicated by that card value will be the estimate recorded for that item. If there is no consensus, then the team members discuss reasons for selecting different cards or estimates. After this discussion, each member picks a card again. This sequence continues until all the assumptions are understood, misunderstandings are resolved, and a majority or consensus is reached. Planning Poker advocates greater interaction and enhanced communication among Scrum Team members. It also facilitates independent thinking by participants, thus avoiding the phenomenon of group think.

3. Fist of Five

Fist of Five is a simple and fast mechanism that can be used as an estimation tool, as well as a general group consensus-building technique. After an initial discussion about a particular User Story (or task) being estimated, Scrum Team members are each asked to vote on a scale of one to five using their fingers, with the number of fingers indicating the relative estimate value. Team members with outlier estimates (i.e., the highest and lowest values) explain their rationale for their estimates to the group and these are discussed. After this discussion, another Fist of Five round is conducted, or a collective decision is made.

The value in using this technique is not only consensus building but also driving discussion because team members are asked to explain the reasons for their estimates. They are also given the opportunity to express any issues or concerns. Used as a general consensus building technique, the proposal or pending decision under consideration is initially discussed, then the team members vote based on their level of agreement and desire for discussion as follows:

One finger: I disagree with the group's conclusion and have major concerns.

Two fingers: I disagree with the group's conclusion and would like to discuss some minor issues.

Three fingers: I am not sure and would like to go with the group's consensus conclusion.

Four fingers: I agree with the group's conclusion and would like to discuss some minor issues.

Five fingers: I wholeheartedly agree with the group's conclusion.

4. Affinity Estimation

Affinity Estimation, such as T-shirt sizing, is a technique used to quickly estimate a large number of User Stories. Using sticky notes or index cards and tape, the team places User Stories on a wall or other surface, in order from small to large. Each team member begins with a subset of User Stories from the Prioritized Product Backlog to place in order by their relative size. This initial placement is done in silence. Once everyone has placed their User Stories on the wall, the team reviews all of the placements and can move User Stories around as appropriate. This second part of the exercise involves a discussion about the placements. Finally, the Product Owner will indicate some sizing categories on the wall. These categories can be small, medium, or large, or they may be numbered using story point values to indicate relative size. The team will then move the User Stories into these categories as the last step in the process. Some key benefits of this approach are that the process is very transparent, visible to everyone, and is easy to conduct.

9.2.2.2 Sprint Planning Meetings

During Sprint Planning Meetings, the User Stories are discussed by the Scrum Core Team. If it is not already done during the creation or the refining of the Product Backlog, each User Story is evaluated and assigned a high-level estimate based on relative story points. See also sections 9.3.2.1, 9.4.2.1, 9.5.2.1, and 9.6.2.1.

9.2.2.3 Prioritized Product Backlog Review Meetings

Prioritized Product Backlog Review Meetings are held as part of the *Refine Prioritized Product Backlog* process. Information from these meetings provides additional clarity about the User Stories and helps in determining their estimates.

9.2.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 9-9 shows Vabro's interface used for estimating user stories. It provides tools for assigning effort values and organizing sprint-ready tasks.

Project

Search

Bhaktaram Sahu

Kanban Template 1 / Design

Product Owner

DesignUS-1User Story Created

OverviewDetailsTask GroupTaskComments

Category

Design fileVabro design

Planned Start Date & End Date

Select Date Range

Priority

☆☆☆Reset

Acceptance Criteria

Enter Acceptance Criteria

High Level Estimate (HLE) in Days

Enter High Level EstimatePoker Game

Estimate (Story Points)

24 Kanban Board-wise breakup

Board NameTeamKanban ManagerEstimate

Board 34Team BbsrBharat kathopali8

Board 24Team CskBharat kathopali8

Board 12Team ChennaiBharat kathopali8

Dependency

Predecessor User StoriesSelect Predecessor User Stories

Successor User StoriesSelect Successor User Stories

Attachments

Attach files

Drag and drop files hereFile size should not exceed 10MB

Sample document 1.xlsxUploading - 47%

Sample document 1.doc

Sample document 1.xlsx

Sample document 1.xlsx

Sample document 1.xlsx

Sample document 1.pdf

Sample docx(1).pngFile size exceeds 10 MB

Sample document 1.doc

Sample document 1.doc

Sample document 1.xlsx

Sample document 1.xlsx

Sample document 1.xlsx

Sample document 1.pdf

Show Less

Attach links

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

https://www.productplan.co...

Show All

CancelSave

Figure 9-9: User Stories Estimation (Source: Vabro)

Figure 9-10 depicts Jira's view of user story estimation. It enables teams to assign story points and align user stories with sprint planning workflows.

Projects / VMEdU_AcadProj / Add epic / VA-1

VMEdU Academics Module

+ Add
 Apps

Description

Add a description...

Confluence content

Product requirements
 TRY TEMPLATE

Activity

All
Comments
History
Work log

Add a comment...

To Do ⚡

Details

Assignee	Brijesh SB Assign to me
Labels	None
Sprint	None
Story point estimate	2
Development	Create branch Create commit

Figure 9-10: Estimating User Stories in Scrum (Source: Jira)

9.2.3 Outputs

9.2.3.1 Estimated User Stories*

After the User Stories are estimated by the Scrum Team using the various estimation techniques discussed in this section, they are considered to be Estimated User Stories.

9.2.3.2 Updated Prioritized Product Backlog

Described in section 9.1.3.3.

9.3 Commit User Stories

In this process, the Scrum Team commits to delivering a set of User Stories for the Sprint. The decision on which User Stories will be committed to is based on the relative value-based priority of the User Stories and the estimated effort and team velocity for one Sprint. As part of this process, the Scrum Team starts the creation of the Sprint Backlog, which contains the committed User Stories that are assigned to a particular Sprint. The backlog is refined further with task-level details as Sprint Planning continues.

With this commitment from the Scrum Team given at the beginning of a Sprint as part of Sprint Planning, the content of the Sprint is defined and cannot be changed once the Sprint implementation phase has begun.

Figure 9-11 shows all the inputs, tools, and outputs for the *Commit User Stories* process.

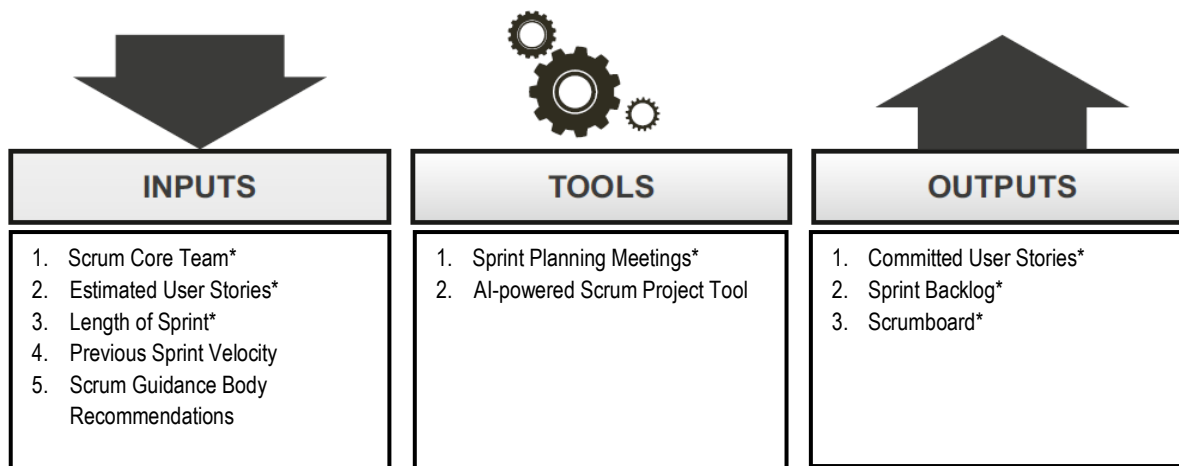


Figure 9-11: Commit User Stories—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-12 illustrates the data flow of the commit user stories process. It maps how selected and estimated stories become sprint tasks with clear acceptance criteria.

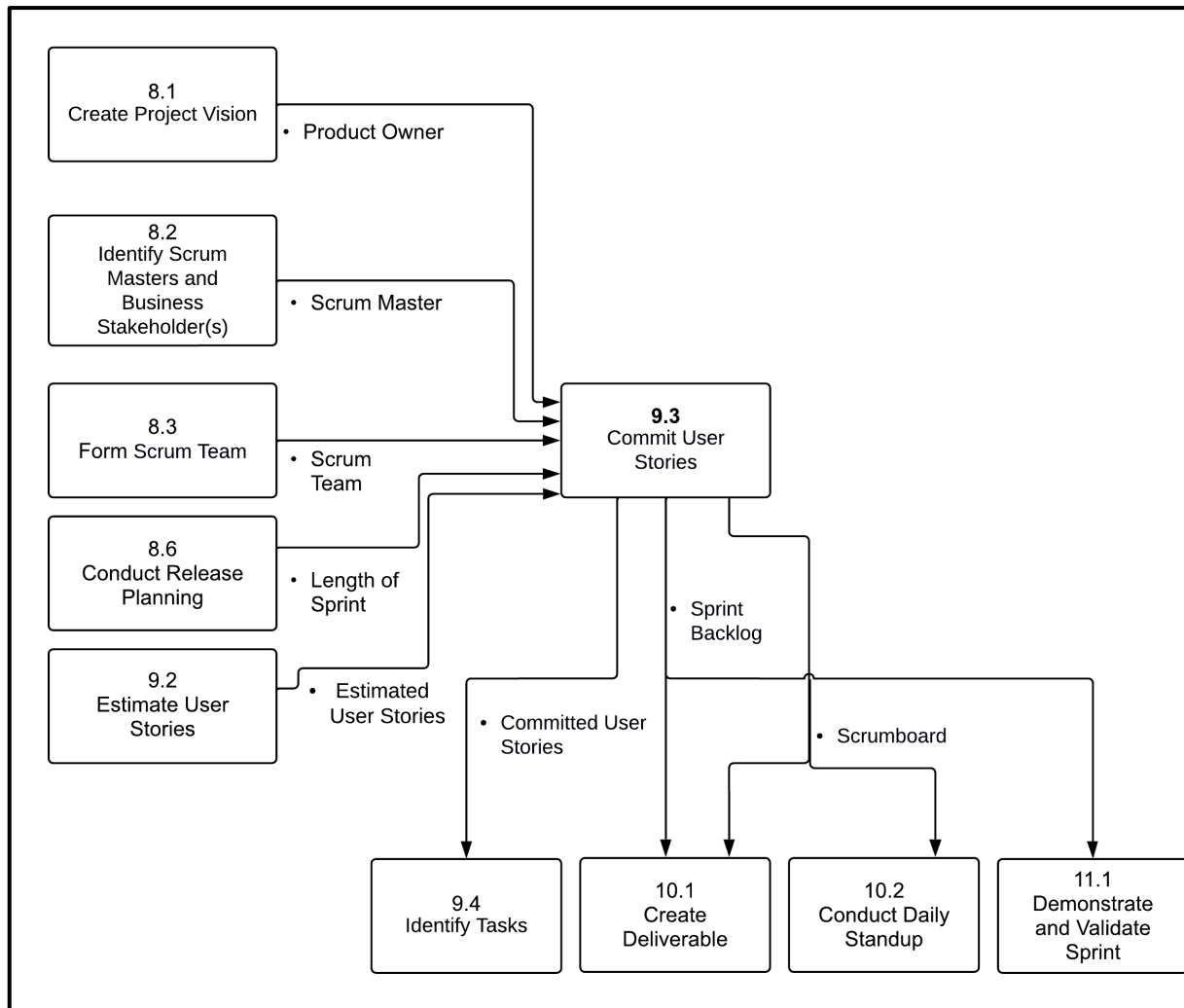


Figure 9-12: Commit User Stories—Data Flow Diagram

9.3.1 Inputs

9.3.1.1 Scrum Core Team*

Described in section 3.2.1.

9.3.1.2 Estimated User Stories*

Described in section 9.2.3.1.

9.3.1.3 Length of Sprint*

Described in section 8.6.3.2.

9.3.1.4 Previous Sprint Velocity

Sprint Velocity is the rate at which the team can complete the work in a Sprint. It is usually expressed in the same units as those used for estimation, normally story points or ideal time. A record of the team's velocity for each Sprint is maintained and used as a reference point for future Sprints. The previous Sprint Velocity becomes the most important factor in determining the amount of work the team can commit to in a subsequent Sprint. Any changes in the situation or conditions since the last Sprint are accounted for to ensure a better estimation of the Sprint Velocity for the upcoming Sprint.

9.3.1.5 Scrum Guidance Body Recommendations

In the *Commit User Stories* process, recommendations from the Scrum Guidance Body may include information on the rules, regulations, standards, and best practices required for the team to effectively commit User Stories and incorporate them into the Sprint Backlog. For more information on Scrum Guidance Body recommendations, see 8.1.1.7.

9.3.2 Tools

9.3.2.1 Sprint Planning Meetings*

In the Sprint Planning Meeting, the Scrum Team comes together to plan the work to be done in the upcoming Sprint. The Product Owner is present during this meeting in case any clarifications of the User Stories or priorities are needed. The team reviews the estimates for those User Stories that are at the top of the Prioritized Product Backlog. To help ensure that the group stays on topic, this meeting should be Time-boxed, with the standard length limited to two hours per week of Sprint duration (e.g., four hours for two-week Sprints). This assists in preventing the tendency to stray into discussions that should occur at other meetings (such as the Release Planning Meeting or the Sprint Review Meeting). As part of this meeting, the entire Scrum Team will commit to delivering a subset of User Stories from the Prioritized Product Backlog in the upcoming Sprint. These committed User Stories are then incorporated into the Sprint Backlog. For more information on Sprint Planning Meetings, see sections 9.2.2.2, 9.4.2.1, 9.5.2.1, and 9.6.2.1.

9.3.2.2 AI-powered Scrum Project Tool

Described in section 2.4.4

9.3.3 Outputs

9.3.3.1 Committed User Stories*

The Scrum Team commits to a subset of the estimated User Stories that they believe they can complete in the upcoming Sprint based upon team velocity. The committed User Stories should always be selected based on the priorities defined by the Product Owner (as incorporated into the Prioritized Product Backlog).

9.3.3.2 Sprint Backlog*

The list of the User Stories to be executed by the Scrum Team in the Sprint is called the Sprint Backlog. This is a subset of the Prioritized Product Backlog that contains the committed User Stories that are assigned to a particular Sprint. The Sprint Backlog will be further refined with task-level details as Sprint Planning continues.

It is common practice for the Sprint Backlog User Stories (and associated tasks) to be represented on a Scrumboard or similar task board, which provides a constantly visible depiction of the current status of the User Stories in the Product Backlog.

Once the Sprint Backlog is finalized and committed to by the Scrum Team, new User Stories should not be added. If new requirements arise during a Sprint, they should be added to the Prioritized Product Backlog to be considered for a future Sprint.

9.3.3.3 Scrumboard*

It is important to track the progress of a Sprint and to know where the Scrum Team stands in terms of completing the User Stories (and tasks) in the Sprint Backlog. A variety of tools can be used to track the work in a Sprint, but one of the most common is a Scrumboard, also known as a task board or a progress chart. Scrum's transparency comes from openly viewable information tools like the Scrumboard, which shows the ongoing progress of the team. The team uses a Scrumboard to plan and track progress during each Sprint.

The most basic version of a Scrumboard has the board divided into three sections— “Work Not Started” (also referred to as “To Do”), “Work In Progress” (also referred to as “In Progress”), and “Completed Work” (also referred to as “Complete.” Sticky notes representing each User Story and their associated tasks are placed in the appropriate category to reflect the status of the work to be complete in the current Sprint. The task notes are moved forward to the next category as the work progresses.

A typical Scrumboard is shown in Figure 9-13. The Scrumboard shows all the User Stories in the left column and has three columns labelled “To Do,” “In Progress,” and “Complete.” As tasks associated with the User Stories are identified and worked on in later processes, those tasks would be depicted under their respective columns.

Figure 9-13 displays a typical Scrumboard in Vabro. It helps visualize sprint progress by showing committed stories, their statuses, and related tasks.

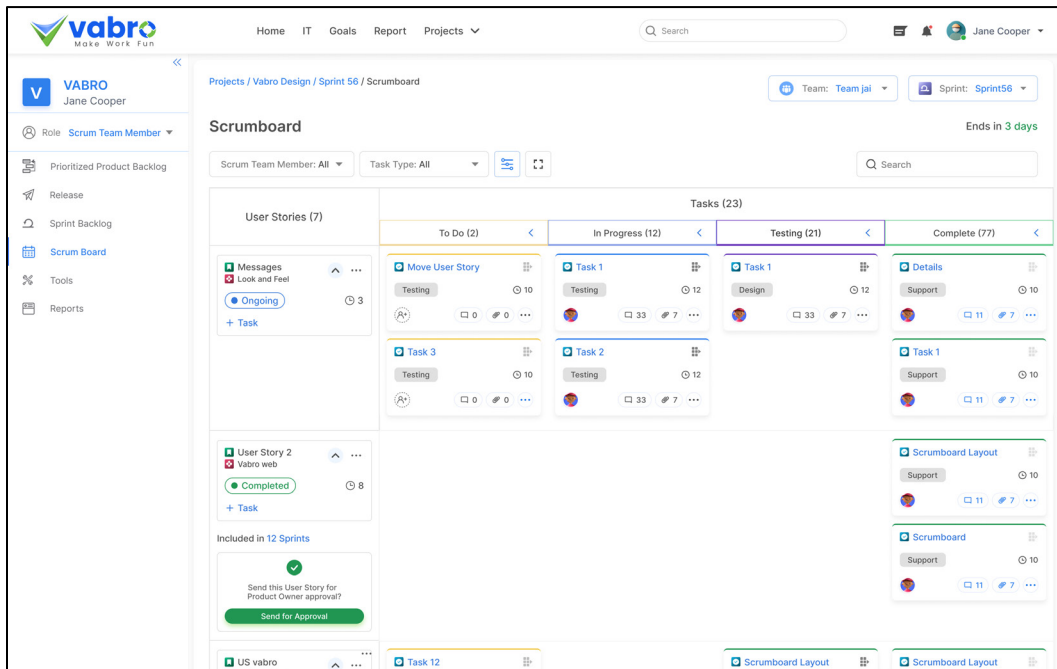


Figure 9-13: Typical Scrumboard (Source: Vabro)

Variations of the typical Scrumboard may be used to depict the status of the team's work more accurately. For example, one variation contains an additional "Testing" column to indicate that the task is complete, but the work result is currently being tested. The "Complete" column in this case is used to represent those tasks that are fully complete and also successfully tested. Instead of a Testing column, the team may include any other column on the Scrumboard that the team thinks will be helpful to track progress.

The Scrumboard can be maintained manually on paper or on a large whiteboard, or it can be maintained electronically in a spreadsheet or using an AI-powered Scrum Project Tool. The Scrum Team should change or add to the Scrumboard as required so that the Scrumboard continues to provide accurate visual information and control about the status of the work being done (as agreed to and committed by the team).

9.4 Identify Tasks

In this process, the committed User Stories are decomposed into specific tasks and compiled into a task list. Identifying tasks could either be done at the beginning of the Sprint for all committed User Stories or before the team starts working on the tasks required for each User Story.

The Product Owner does not play an active role in identifying tasks but needs to be available to answer any questions from the Scrum Team that may arise while decomposing the User Stories into tasks.

Figure 9-14 shows all the inputs, tools, and outputs for the *Identify Tasks* process.

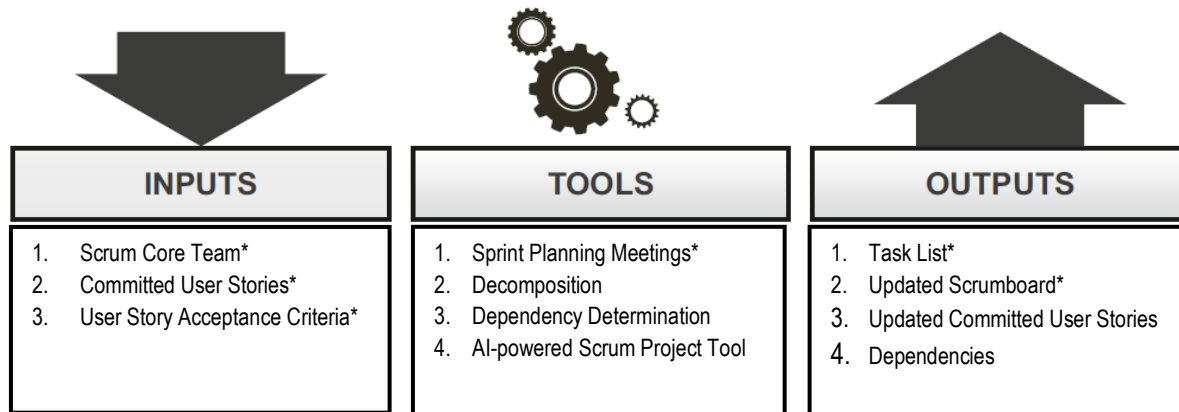


Figure 9-14: Identify Tasks—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-15 depicts the data flow for identifying tasks. It illustrates how stories are broken into tasks and arranged in the Scrumboard to support sprint planning.

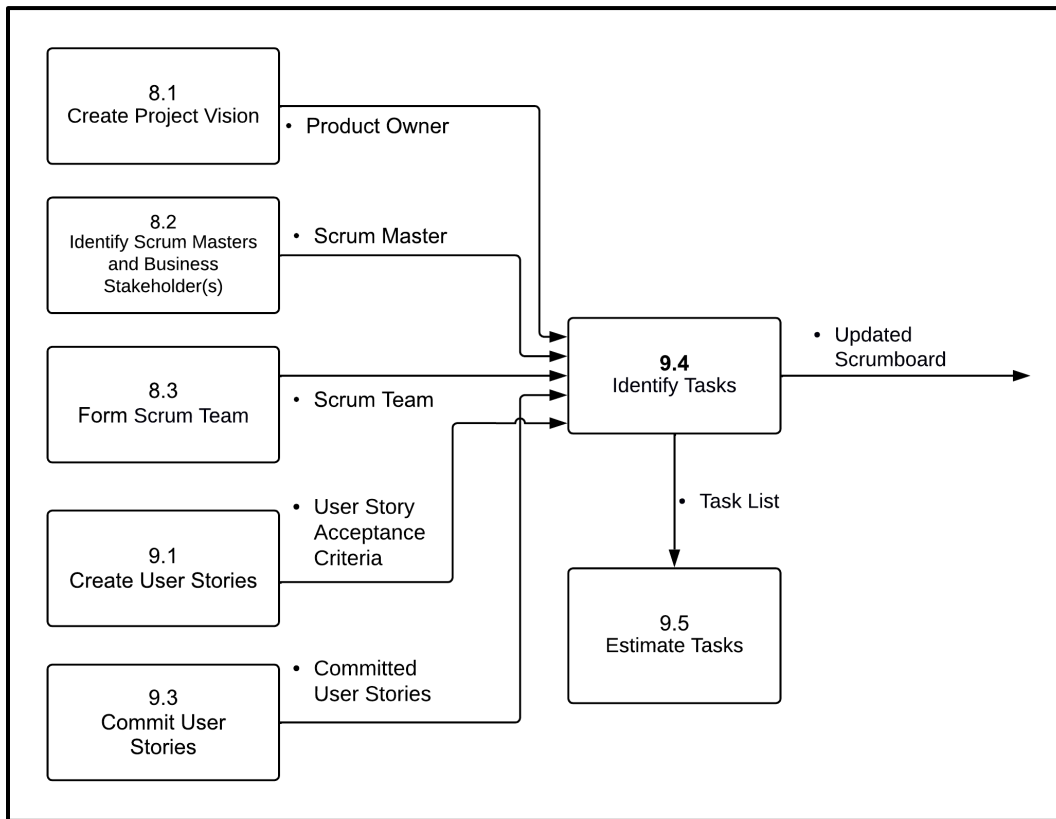


Figure 9-15: Identify Tasks—Data Flow Diagram

9.4.1 Inputs

9.4.1.1 Scrum Core Team*

Described in section 3.2.1.

9.4.1.2 Committed User Stories*

Described in section 9.3.3.1.

9.4.1.3 User Story Acceptance Criteria*

Described in section 9.1.3.2.

The Product Owner must ensure that the defined Acceptance Criteria are appropriate for the User Stories and must also provide clarity regarding the requirements for the Scrum Team. Understanding Acceptance Criteria by the Scrum Team helps them determine which tasks are needed to satisfy the requirements of the User Story.

Acceptance testing refers to the assessment of the ability of the completed deliverable to meet its Acceptance Criteria. This provides information to the Product Owner to help decide about approving or rejecting the deliverables. The Acceptance Criteria should be crisp, unambiguous, and specific. They should be defined to ensure that the team is able to verify that the outcomes are in alignment with the sponsor organization's goals and objectives.

9.4.2 Tools

9.4.2.1 Sprint Planning Meetings*

In the Sprint Planning Meetings, the Scrum Team convenes to plan the work to be done in the upcoming Sprint. The team reviews each committed User Story for the Sprint and identifies actionable activities, or tasks required to implement the deliverables necessary to fulfill the User Story and meet the Acceptance Criteria. The Product Owner is present at this meeting in case clarification is required related to the committed User Stories to help the team make design decisions. For more information on Sprint Planning Meetings, see sections 9.2.2.2, 9.3.2.1, 9.5.2.1, and 9.6.2.1.

9.4.2.2 Decomposition

The Scrum Team uses decomposition to break down the User Stories in the upcoming Sprint into detailed tasks. The User Stories should be sufficiently decomposed to a level that provides the Scrum Team with adequate information needed to create the deliverables using the tasks captured in the Task List.

9.4.2.3 Dependency Determination

Once the Scrum Team has selected the User Stories for the upcoming Sprint, the team should then consider any dependencies, including those related to the availability of people, as well as any technical dependencies. Properly documenting dependencies helps the Scrum Team determine the relative order in which tasks should be executed to create the Sprint deliverables. Dependencies also highlight the relationship and interaction between tasks both within the Scrum Team working on a given Sprint and across other Scrum Teams in the project. For more information on determining dependencies, see section 8.5.2.6.

9.4.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 9-16 shows the Vabro interface used to create tasks from user stories. It supports task assignments, estimates, and dependency tracking.

Figure 9-16: Task Creation in Scrum Project Tool (Source: Vabro)

Figure 9-17 displays Jira's task creation interface, enabling teams to generate and manage sprint tasks linked to specific user stories.

Figure 9-17: Task Creation in Scrum Project Tool (Source: Jira)

9.4.3 Outputs

9.4.3.1 Task List*

The Task List is a comprehensive list that contains all the tasks which the Scrum Team has committed to for the current Sprint and their corresponding descriptions. The level of granularity to which the tasks are decomposed is decided by the Scrum Team. The Task List must include any testing and integration efforts so that the product increment from the Sprint can be successfully integrated into the deliverables from previous Sprints. The Task List is used by the Scrum Team during Sprint Planning Meetings to update the Sprint Backlog and to create the Sprint Burndown Chart. It is also used to determine if the team needs to reduce its commitment, or if they can take on additional User Stories during Sprint Planning for the next Sprint.

9.4.3.2 Updated Scrumboard*

As tasks are identified, the Scrumboard is updated to show the tasks associated with each User Story. Tasks are typically shown on sticky notes placed on a physical Scrumboard or as entries under the applicable User Stories when using an electronic AI-powered Scrum Project Tool. During implementation, as the team adds, assigns, and updates tasks being worked on, the Scrumboard keeps getting updated with the additional tasks and the status of each task. If the team has estimated the tasks, the task estimates are also depicted on the Scrumboard. In the example in Figure 9-18, the Scrumboard shows four User Stories. User Story 1, 2, and 3 have been decomposed into tasks, but User Story 4 has not yet been decomposed into tasks. At the beginning of a Sprint, all tasks for that Sprint are placed in the 'To Do' column and are subsequently moved forward according to their progress. For example, User Story 1 has 7 tasks, all of which are categorized as "To Do," which indicates that the Scrum Team has not started working on any of those tasks.

For more information on the Scrumboard, see sections 9.3.3.3 and 10.1.1.3.

Figure 9-18 depicts a scrumboard with identified tasks for multiple user stories. Shows decomposition status, with tasks in 'to do' column, and differentiates user stories that have or have not been broken into tasks yet. Useful for tracking sprint readiness.

USER STORIES	TASKS		
	<i>To Do</i>	<i>In Progress</i>	<i>Complete</i>
1			
2			
3			
4			

Figure 9-18: Scrumboard with Identified Tasks

9.4.3.3 Updated Committed User Stories

The User Stories are updated during this process. Updates can include revisions to the original User Story estimates based on task creation and complexity factors discussed during the Sprint Planning Meeting. Committed User Stories are described in section 9.3.3.1.

9.4.3.4 Dependencies

Dependencies describe the relationship and interaction between different tasks in a project and can be classified as mandatory, discretionary, internal, or external, as discussed in section 8.5.2.6.

9.5 Estimate Tasks

This is an optional process which involves creating task estimates if the Scrum Team sees value in doing so. In this process, the Scrum Team estimates the effort required to accomplish each task in the Task List. Task estimates could either be determined at the beginning of the Sprint for all User Stories/tasks relevant to that Sprint, or for each task just before the team starts working on the particular User Story/task. The estimation can be done using the same methods that were used for the *Estimate User Stories* process.

Figure 9-19 shows all the inputs, tools, and outputs for the *Estimate Tasks* process.

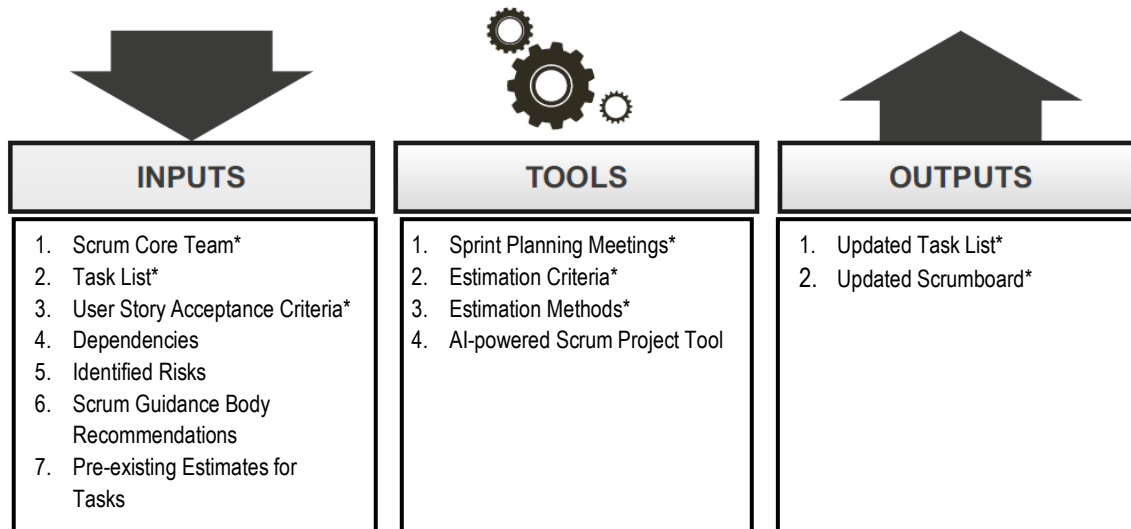


Figure 9-19: Estimate Tasks—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-20 is a data flow diagram representing the “estimate tasks” process. Shows the flow from task inputs to estimation activities and the resulting outputs such as updated scrumboard and task lists.

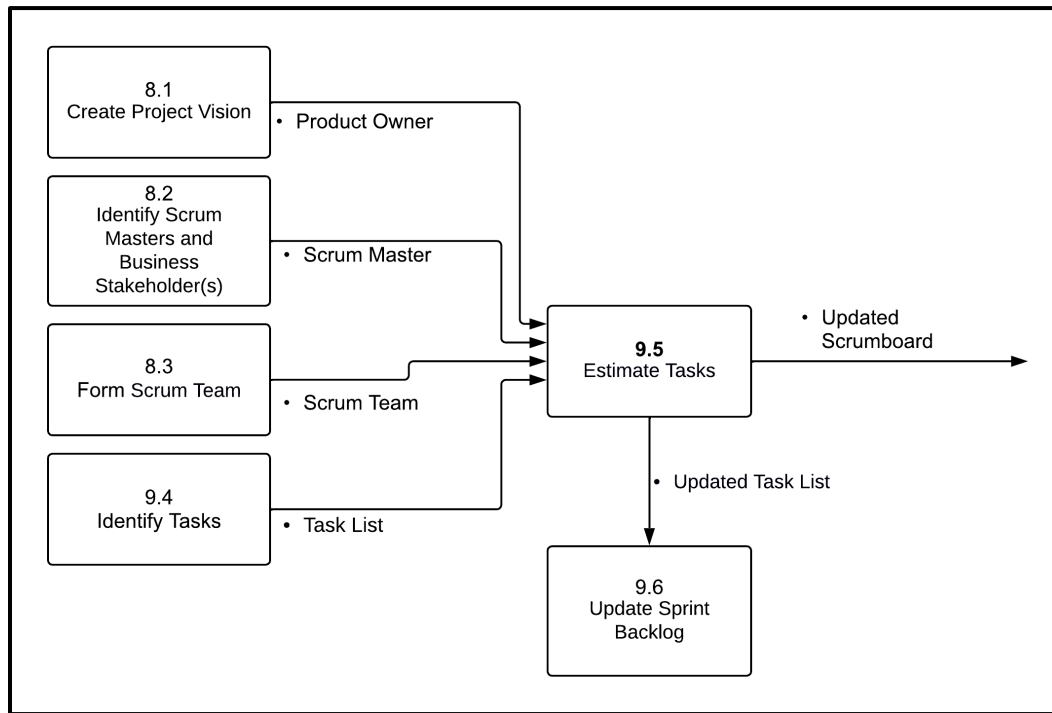


Figure 9-20: Estimate Tasks—Data Flow Diagram

9.5.1 Inputs

9

9.5.1.1 Scrum Core Team*

Described in section 3.2.1.

9.5.1.2 Task List*

Described in section 9.4.3.1.

9.5.1.3 User Story Acceptance Criteria*

Every User Story has associated Acceptance Criteria. User Stories are subjective, so the Acceptance Criteria provide the objectivity required for the User Story to be considered as Done or not Done during the Sprint Review (that occurs during the *Demonstrate and Validate Sprint* process). For more information on the User Story Acceptance Criteria, see section 9.1.3.2.

9.5.1.4 Dependencies

Described in section 9.4.3.4.

9.5.1.5 Identified Risks

Described in section 8.4.3.4.

9.5.1.6 Scrum Guidance Body Recommendations

In the *Estimate Tasks* process, Scrum Guidance Body Recommendations may include information on rules, regulations, standards, and best practices required to effectively estimate tasks in the Task List. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

9.5.1.7 Pre-existing Estimates for Tasks

Some pre-existing task estimates may be considered by the Scrum Team members when estimating tasks. Similar tasks may have been previously completed in the same project, or in other past projects, and the effort and time taken to complete those similar tasks can help the Scrum Team to create better task estimates during this process. Experts who have worked on similar tasks in the past may also be able to provide some effort estimates for tasks. However, it is important to ensure that the Scrum Team creates their own task estimates, instead of solely relying on any pre-existing estimates. Pre-existing task estimates may also help the Scrum Team reevaluate the commitment made to the Product Owner at the User Story level.

9.5.2 Tools

9.5.2.1 Sprint Planning Meetings*

As part of the Sprint Planning Meetings, the Scrum Team estimates the effort required to complete a task or set of tasks and to estimate the people effort and other resources required to carry out the tasks within a given Sprint. The Scrum Team members use the Task List to estimate the effort for the User Stories to be completed in the Sprint. One of the key benefits of this technique is that it enables the team to have a shared perspective of the User Stories and requirements so that they can reliably estimate the effort required. For more information on Sprint Planning Meetings, see sections 9.2.2.2, 9.3.2.1, 9.4.2.1, and 9.6.2.1.

9.5.2.2 Estimation Criteria*

Estimation criteria can be expressed in numerous ways, with two common examples being story points and ideal time. Story point values are used to represent relative or comparative effort to complete tasks. Whereas ideal time normally describes the number of hours a Scrum Team member works exclusively on developing the project's deliverables, without including any time spent on other activities or work that is outside the project. Estimation criteria make it easier for the Scrum Team to estimate effort and enable them to evaluate and address inefficiencies when necessary.

9.5.2.3 Estimation Methods*

The same estimation methods used to estimate User Stories can be applied to tasks as well. For more information on estimation methods, see section 9.2.2.1.

9.5.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4.

Figure 9-21 shows how Vabro facilitates task estimation in scrum using ai-powered features. Includes automated suggestion tools and effort estimation inputs for scrum teams to plan more effectively.

The screenshot displays the Vabro project management tool interface for a task titled 'Task1'. The interface includes a navigation bar with 'Project' and a search bar. The task details section shows 'Task1' with a status of 'ST-1' and a 'Task Created' notification. Below this, there are input fields for 'Type' (a dropdown menu), 'Estimate (Hours)' (a text input), 'Time Tracking' (a timer set to 00:00:00), 'Planned Start Date' (a date picker), and 'Planned Target Date' (a date picker). The 'Attachments' section features a 'Drag and drop files here' area with a 10MB limit, followed by a row of file thumbnails including 'Sample document 1.xlsx' (uploading at 67%), 'Sample document 3.xlsx', 'Sample document 4.xlsx', 'Sample document 5.xlsx', 'Sample document 6.xlsx', and 'Sample document 1.xlsx'. Below the attachments are 'Attach links' and a 'Save & Continue' button. The user 'Bhaktaram Sahu' is logged in, and there are 6 watchers for the task.

Figure 9-21: Task Estimation in Scrum (Source: Vabro)

Figure 9-22 demonstrates Jira's task estimation interface. Includes visuals for inputting task data, viewing estimates, and tracking sprint effort. It helps streamline the planning process using real-time collaboration.

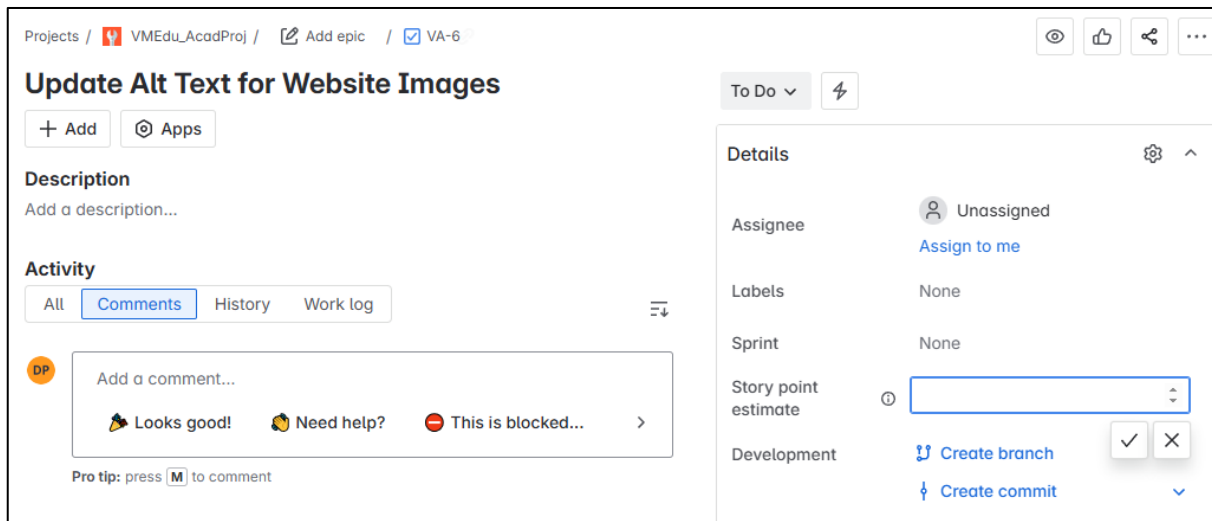


Figure 9-22: Use of Scrum Tool for Task Estimation (Source: Jira)

9.5.3 Outputs

9.5.3.1 Updated Task List*

The Task List is updated to include the estimated efforts that were determined using the detailed estimation activities undertaken in the *Estimate Tasks* process. There may also be re-estimations resulting from changes in the Scrum Team's collective understanding of User Stories and requirements. Estimated effort is expressed in terms of the estimation criteria agreed on by the team. Typically, the accuracy of the estimates varies with team skills. The updated Task List is used by the Scrum Team during Sprint Planning Meetings to update the Sprint Backlog and to create the Sprint Burndown Chart. It is also used to determine if the team needs to reduce its commitment, or if they can take on additional User Stories during Sprint Planning for the next Sprint.

9.5.3.2 Updated Scrumboard*

As tasks are estimated, the effort estimates are then updated in the Scrumboard. For more information on the Scrumboard, see sections 9.3.3.3 and 9.4.3.2.

9.6 Update Sprint Backlog

In this process, the Scrum Core Team updates the Sprint Backlog with task details and if available, the task estimates. The updated Sprint Backlog will be used in the Implement phase to track the team's progress during the upcoming Sprint.

Figure 9-23 shows all the inputs, tools, and outputs for the *Update Sprint Backlog* process.

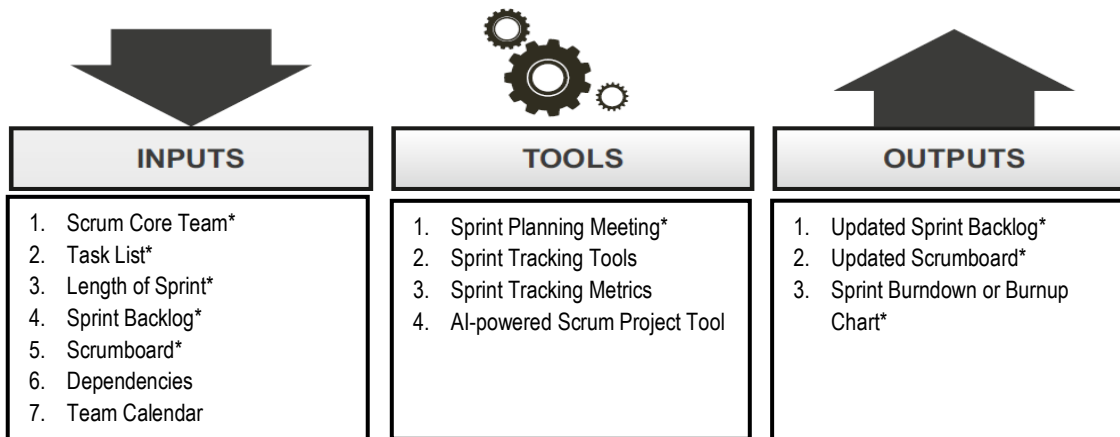


Figure 9-23: Update Sprint Backlog—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 9-24 shows data flow diagram for the “update sprint backlog” process. Visualizes how various tools and inputs lead to an updated sprint backlog and scrumboard.

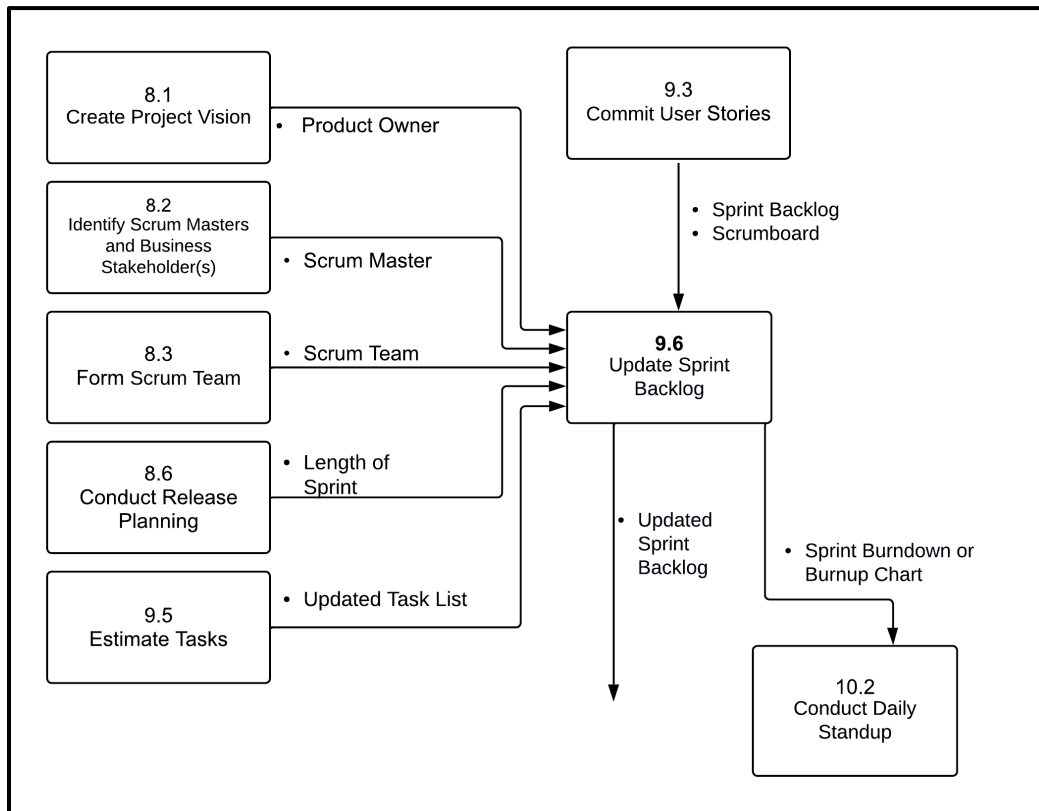


Figure 9-24: Update Sprint Backlog—Data Flow Diagram

9.6.1 Inputs

9.6.1.1 Scrum Core Team*

Described in section 3.2.1.

9.6.1.2 Task List*

Described in section 9.5.3.1.

9.6.1.3 Length of Sprint*

Described in section 8.6.3.2.

9.6.1.4 Sprint Backlog*

Described in section 9.3.3.2.

9.6.1.5 Scrumboard*

Described in section 9.3.3.3.

9.6.1.6 Dependencies

Described in section 9.4.3.4.

9.6.1.7 Team Calendar

A team calendar contains information regarding the availability of team members, including information related to employee vacation, leaves, major events, and holidays. One of the major objectives of using a Team Calendar is to track what each team member is working on throughout the project. It helps the team not only in planning and executing the Sprints efficiently but also in aligning the Sprints with release dates.

9.6.2 Tools

9.6.2.1 Sprint Planning Meetings*

During Sprint Planning Meetings, User Stories are committed for a Sprint, and Tasks are identified and estimated by the Scrum Team. Each Scrum Team member also uses the Task List to select the tasks they plan to work on in the Sprint, based on their skills and experience. The Scrum Team also updates the Sprint Backlog and creates Sprint Burndown Chart using the User Stories and the Task List during the Sprint Planning Meetings. For more information on Sprint Planning Meetings, see sections 9.2.2.2, 9.3.2.1, 9.4.2.1, and 9.5.2.1.

9.6.2.2 Sprint Tracking Tools

It is important to track the progress of a Sprint and to know where the Scrum Team stands in terms of completing the tasks in the Sprint Backlog. A variety of tools can be used to track the work in a Sprint. One of the most common tools is a Scrumboard, also known as a task board or a progress chart. For more information on the Scrumboard, see sections 9.3.3.3 and 9.4.3.2.

9.6.2.3 Sprint Tracking Metrics

Tracking metrics used in Scrum projects include velocity, business value delivered, and number of stories.

- **Velocity**—represents the number of User Stories or functionalities delivered in a single Sprint.
- **Business value delivered**—measures the value of the User Stories delivered from the business perspective.
- **Number of stories**—refers to how many User Stories are delivered as part of a single Sprint. It can be expressed in terms of simple count or weighted count.

9.6.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 9-25 displays how to set up the sprint backlog in Vabro. Shows visual categorization of tasks, priority levels, and assignment options to prepare for sprint execution.

The screenshot shows the 'Create Sprint Backlog' form in the Vabro application. The form is organized into several sections:

- Scrum Team:** A dropdown menu to select the team.
- Duration:** Two dropdown menus for 'Select Weeks' and 'Select Days'.
- Start Date:** A date picker field.
- End Date:** A date picker field.
- Name:** A text input field.
- Working Days:** A text input field.
- Sprint Goal:** A text input field.
- Definition of Done (DoD):** A list of criteria with checkboxes, including 'Reviewed by other team members', 'Completed unit testing of the User Story', 'Completion of quality assurance tests', 'Completion of all documentation related to the User Story', 'All issues are fixed', and 'Successful demonstration to stakeholders and/or business representatives'.
- Generate with Vabro Genie:** A button with a magic wand icon.
- Scrumboard Preview:** A small preview of a Kanban board with columns like 'To Do', 'In Progress', and 'Testing'.
- Automatically start Sprint:** A checkbox.
- Automatically renew Sprint:** A checked checkbox.
- Sprint Name Format:** A dropdown menu showing a format like '{Start_Date}_{Team_Name}'.
- Add all Recurring User Stories of this Sprint to the new Sprint:** A checked checkbox.
- Buttons:** 'Cancel' and 'Save & Continue' buttons at the bottom right.

Figure 9-25: Setting up Sprint Backlog in Scrum (Source: Vabro)

Figure 9-26 illustrates Jira's interface for using the sprint backlog. Includes filtering, prioritizing, and linking tasks to user stories for smooth backlog management.

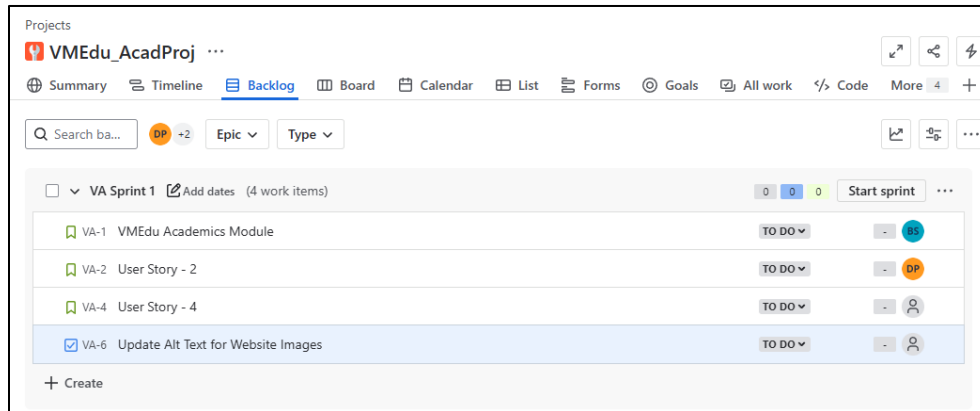


Figure 9-26: Use of Sprint Backlog in Scrum Project Tool (Source: Jira)

9.6.3 Outputs

9.6.3.1 Updated Sprint Backlog*

The Scrum Core Team updates the Sprint Backlog with details of the tasks associated with the committed User Stories in the Sprint Backlog. If available, task estimates are also updated in the Sprint Backlog. The Sprint Backlog is used in the Implement phase to track the team's progress during the Sprint. Once the Sprint Backlog is finalized and committed to by the Scrum Team, new User Stories should not be added; however, tasks that might have been missed or overlooked from the committed User Stories may need to be added. If new requirements arise during a Sprint, they will be added to the Prioritized Product Backlog and included for consideration in a future Sprint.

9.6.3.2 Updated Scrumboard*

The Scrumboard is updated to reflect the information in the updated Sprint Backlog including any updates to the tasks, task statuses, and task estimates, if available. For more information on the Scrumboard, see sections 9.3.3.3 and 9.4.3.2.

9.6.3.3 Sprint Burndown or Burnup Chart*

Burn Charts (Burndown or Burnup) are used to track progress in a Scrum project. A Burndown Chart is a graph that depicts the amount of work remaining in relation to the remaining time. Unlike the Burndown Chart, a Burnup Chart depicts what has been completed in relation to the remaining time.

For more details, please refer to Section 4.6.1.

9.7 Plan and Estimate Phase Data Flow Diagram

Figure 9-27 is a comprehensive data flow diagram for the entire 'plan and estimate' phase. Maps interdependencies between tasks, sprints, and backlog elements to align planning activities.

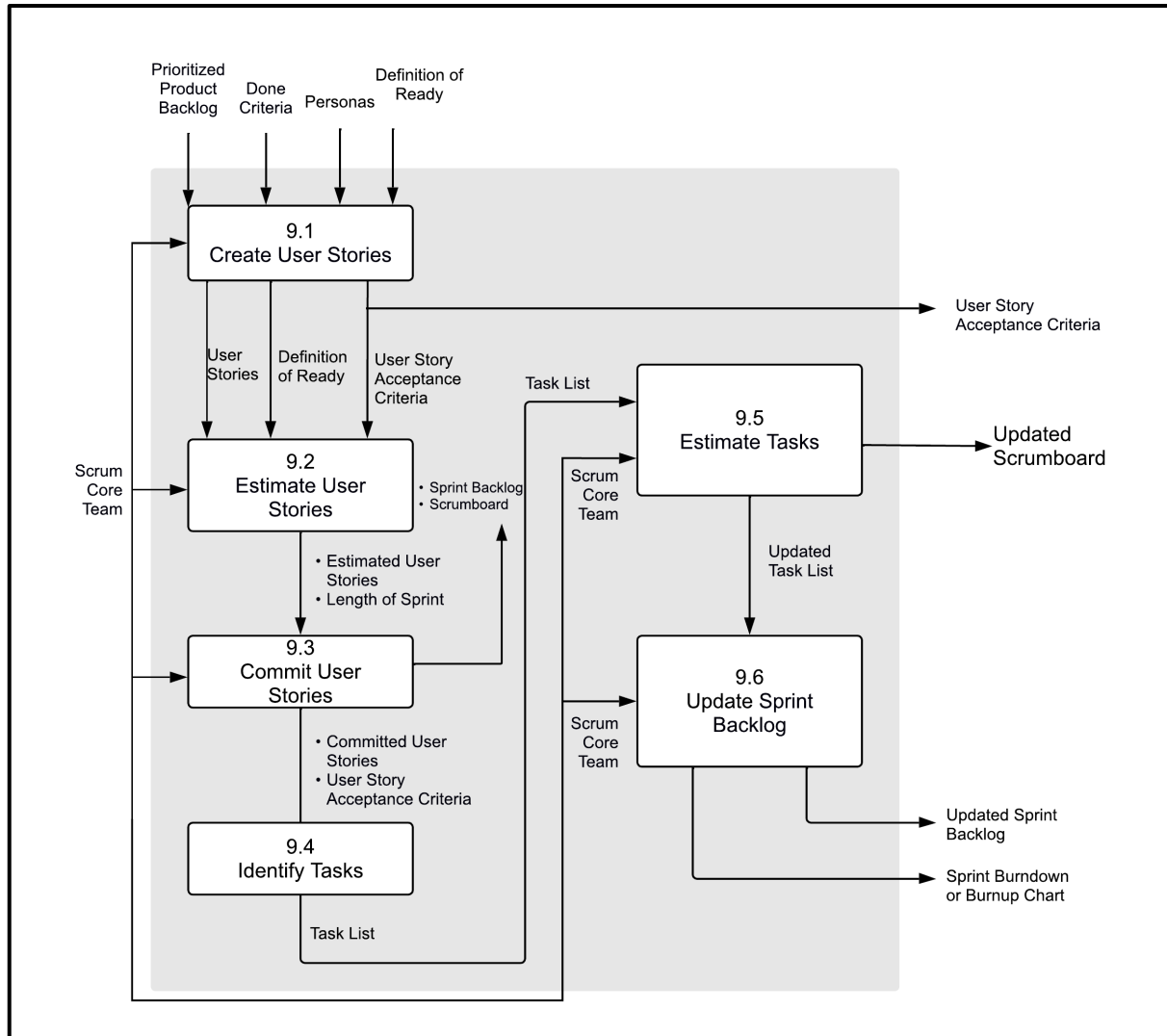


Figure 9-27: Plan and Estimate Phase—Data Flow Diagram

10.IMPLEMENT

The Implement phase is related to the execution of the tasks and activities to create a project's product. These activities include creating various deliverables, conducting Daily Standup Meetings, and refining (i.e., reviewing, fine-tuning, and regularly updating) the Product Backlog at regular intervals.

Implement, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Product Owners, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter.

This chapter is written from the perspective of one Scrum Team working on one Sprint to produce potentially shippable deliverables, which could be part of a larger project, program, or portfolio. Additional information pertaining to Scaling Scrum for Large Projects is available in chapter 13. Additional information pertaining to Scaling Scrum for the Enterprise can be found in chapter 14.

Implement is the second of the three phases that are done repetitively in every Sprint. This phase begins after Sprint planning is complete. It is the core of every Scrum project where the bulk of the work is done.

The Scrum Team, facilitated by the Scrum Master, creates the deliverables that are associated with the committed User Stories by working on and completing the tasks the team identified in the previous phase.

While the Scrum Team is creating the deliverables of the Sprint, the Product Owner updates and refines the Prioritized Product Backlog to keep it up to date with any changes in requirements and/or priorities and to ensure that the set of User Stories the Product Owner would like the team to commit to in the next Sprint will be ready for commitment.

It is also important to realize that although all phases and processes are defined uniquely in the SBOK® Guide, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to overlap some phases and/or processes, depending on the specific requirements of each project.

Figure 10-1 provides an overview of the “implement” phase in scrum. Summarizes key processes like creating deliverables, conducting daily standups, and refining the backlog, along with associated roles and expected outputs.

10.1 Create Deliverables—In this process, the Scrum Team creates the Sprint deliverables by working on the tasks in the Sprint Backlog. This is the process where the Scrum Team and the Scrum Master spend most of their time. The team is supported by the Scrum Master, who facilitates meetings for the team, addresses impediments the team faces, and does whatever he/she can do to allow the Scrum Team members to focus on the creation of the Sprint Deliverables.

The Scrum Team uses a Scrumboard to track its progress during the Sprint. The Scrum Team uses the information about its progress to get a good indication of its ability to deliver according to its commitment and, if necessary, to take action to secure the most valuable outcome of the Sprint that is possible under the given circumstances.

10.2 Conduct Daily Standup—In this process, a highly focused Daily Standup Meeting is conducted. This Time-boxed meeting is the forum for the Scrum Team to update each other on their progress and any impediments they may be facing.

10.3 Refine Prioritized Product Backlog—In this process, the Product Owner continuously updates and maintains the Prioritized Product Backlog. A Prioritized Product Backlog Review Meeting may be held, during which any changes or updates to the Product Backlog are discussed and incorporated into the Prioritized Product Backlog as appropriate.

In order to keep the Prioritized Product Backlog up to date with any change in requirements and/or priorities, the Product Owner continually works with the customer and other business stakeholders to capture and understand any changes in their needs.

To ensure that the set of User Stories the Product Owner would like the team to commit to in the next Sprint will be ready for commitment, the Product Owner refines existing Epics and User Stories in the Prioritized Product Backlog, and ensures that the User Stories satisfy the Definition of Ready.

As part of refining the Prioritized Product Backlog, the Product Owner also works with the Scrum Team to get feedback and questions related to the updates in the Prioritized Product Backlog, potentially including estimates.

If changes in requirements and/or the overall progress of the Scrum Team require changes to the Release Schedule and/or the business justification, the Product Owner will also make these changes during this process.

This is the process where the Product Owner will spend most of his/her time.

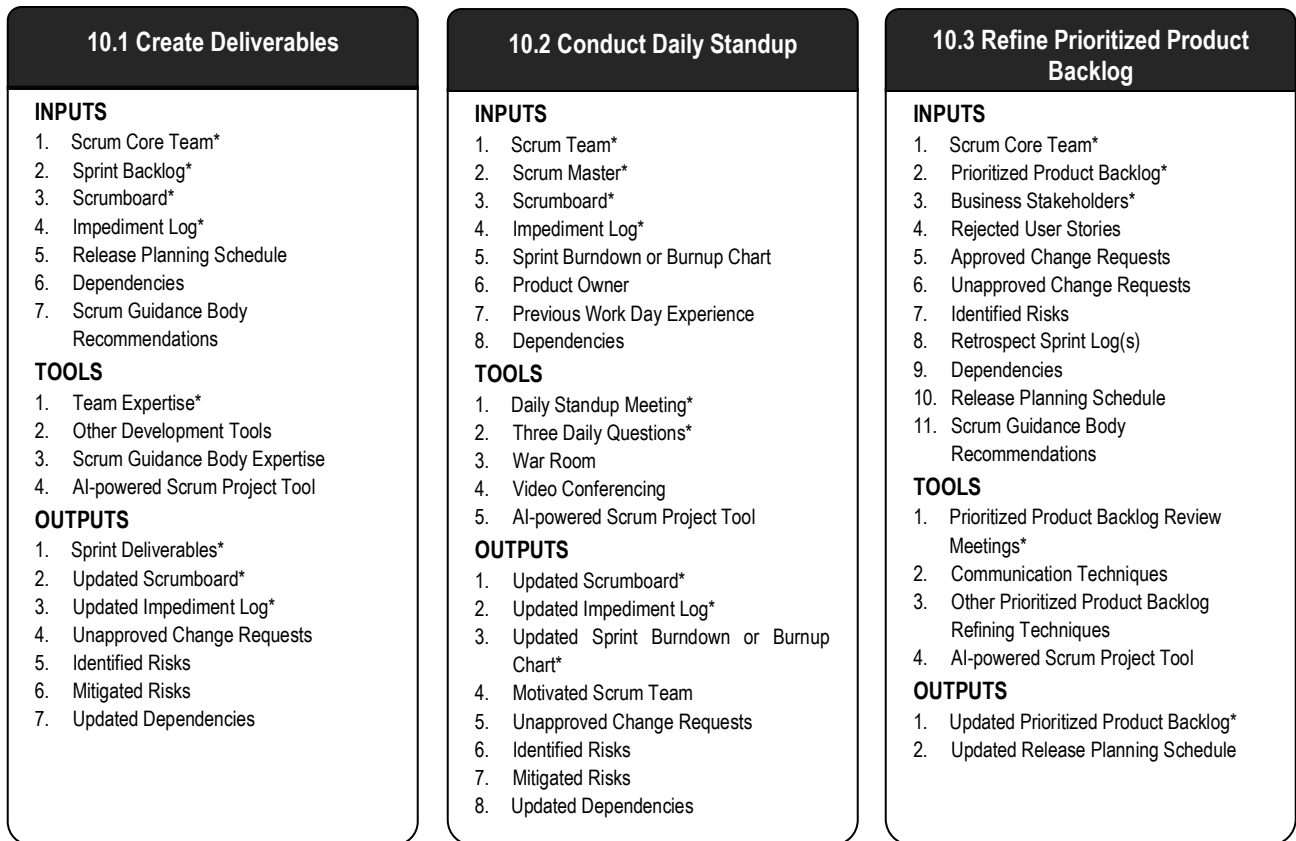


Figure 10-1: Implement Overview

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 10-2 below shows the mandatory inputs, tools, and outputs for processes in the Implement phase.

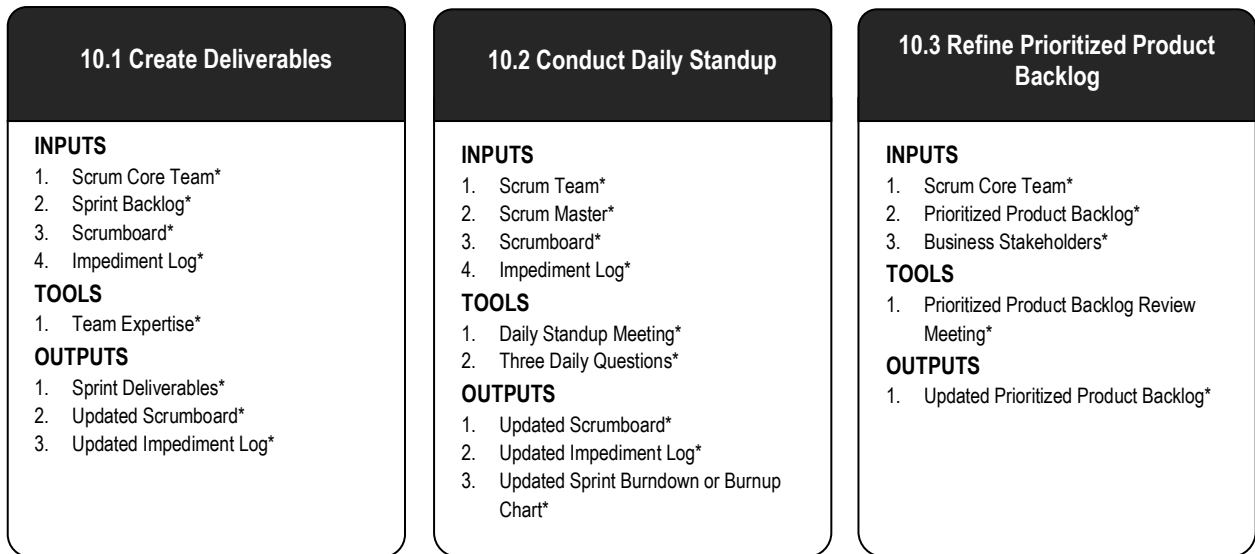


Figure 10-2: Implement Overview (Essentials)

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

10.1 Create Deliverables

In this process, the Scrum Team creates the Sprint deliverables by working on the tasks in the Sprint Backlog. This is the process where the Scrum Team and the Scrum Master spend most of their time. The team is supported by the Scrum Master, who facilitates meetings for the team, addresses impediments the team faces, and does whatever he/she can do to allow the Scrum Team members to focus on the creation of the Sprint Deliverables.

The Scrum Team uses a Scrumboard to track its progress during the Sprint. The Scrum Team uses the information about its progress to get a good indication of its ability to deliver according to its commitment and, if necessary, to take action to secure the most valuable outcome of the Sprint that is possible under the given circumstances.

Figure 10-3 shows all the inputs, tools, and outputs for the *Create Deliverables* process.

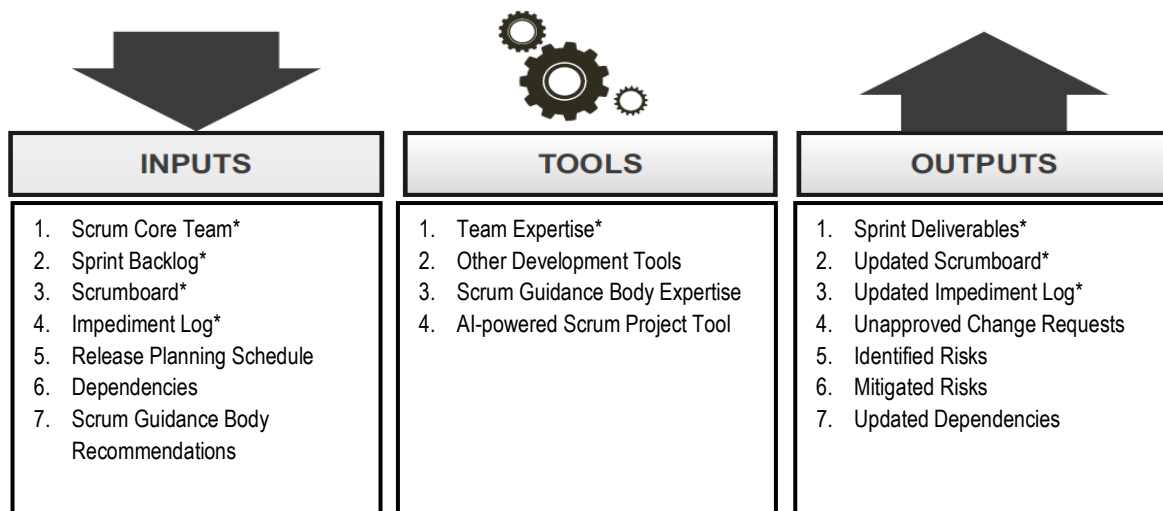


Figure 10-3: Create Deliverables—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 10-4 a data flow diagram for “create deliverables.” It illustrates how scrum team input and planning artifacts flow into tools and processes to result in deliverables and related updates.

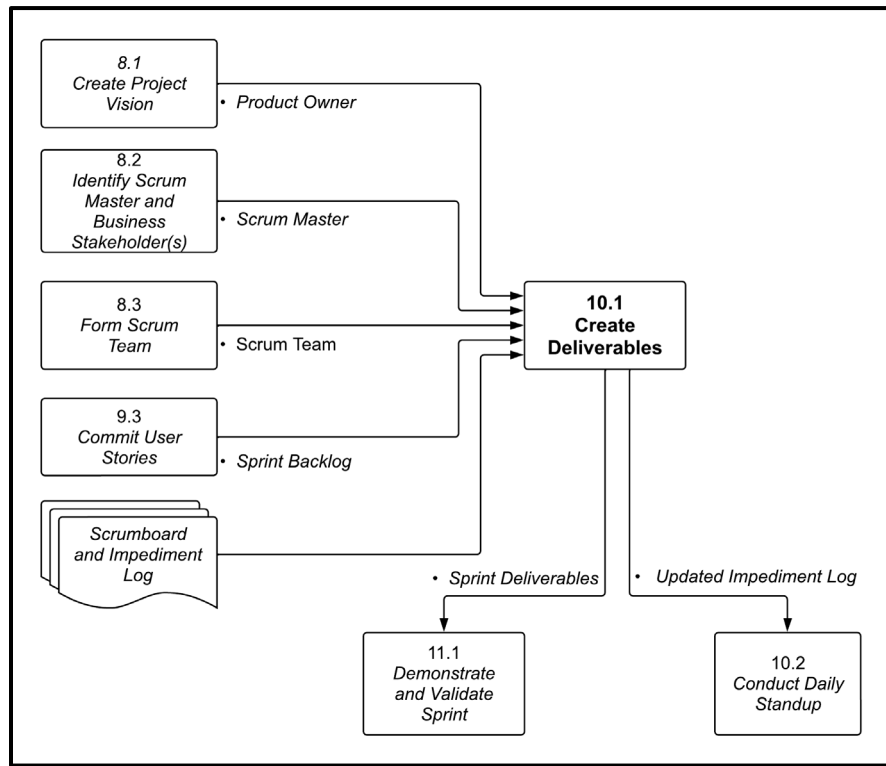


Figure 10-4: Create Deliverables—Data Flow Diagram

10.1.1 Inputs

10.1.1.1 Scrum Core Team*

Described in section 3.2.1.

10.1.1.2 Sprint Backlog*

Described in sections 9.3.3.2 and 9.6.3.1.

10.1.1.3 Scrumboard*

Described in sections 9.3.3.3 and 9.4.3.2.

Scrum's transparency comes from openly viewable information tools like the Scrumboard, which shows the team's progress. The team uses a Scrumboard to plan and track progress during each Sprint. The Scrum Team should update the Scrumboard as required so that the Scrumboard provides accurate visual information and control about the work going on as agreed and committed by the team.

The team uses the Scrumboard created from the Sprint Backlog in the Plan and Estimate Phase, which would initially have all the tasks in the "To Do" column at the beginning of the Sprint. Scrum Team members review the User Stories and Tasks in the Scrumboard on a daily basis and keep moving tasks to "In Progress" and "Complete" columns as the work progresses. An additional column (such as "Testing") may be added to the Scrumboard depending on the workflow of the Scrum Team as they create deliverables.

As the team keeps adding/updating tasks, and assigning tasks to work on, the Scrumboard keeps getting updated with the additional tasks, and the status of the tasks. For example, Figure 10-5 shows that all the tasks for User Story 1 are complete; but the team is currently working on some tasks for User Stories 2 and 3. User Story 4 has been decomposed into tasks, but the Scrum Team has not yet started working on the tasks for this User Story.

Figure 10-5 shows a scrumboard with columns for to do, in progress, and complete. Reflects real-time team progress and helps track task status throughout the sprint cycle.

USER STORIES	TASKS		
	<i>To Do</i>	<i>In Progress</i>	<i>Complete</i>
1			
2			
3			
4			

Figure 10-5: Scrumboard with Tasks To Do, In Progress, and Complete

To ensure that Scrum Team members take ownership for their work, it is recommended that the Scrum Team member working on a particular task moves the task from "To Do" to "In Progress" and puts his/her name on the task, so that the task is self-assigned to the person responsible for completing it. Also, only one Scrum Team member should be responsible for completing each task. So, User Stories should be broken down into tasks such that only one person can be responsible for one task until its completion.

As the Scrum Team starts working on a User Story, they may have gained a better understanding of the tasks required to complete the User Story. This may necessitate tasks being added, updated, or deleted from the Scrumboard as decided by the Scrum Team.

When all the tasks for a User Story are completed (e.g., as in User Story 1 in the above figure), the User Story is considered completed by the Scrum Team. Completed User Stories are then available for the Product Owner to review and either approve or reject. The review of User Stories by the Product Owner can be done by the Product Owner either, after the User Stories are completed, or during the *Demonstrate and Validate Sprint* process. If a User Story is approved by the Product Owner, then that User Story is considered as “Done” by the Scrum Team (and no more work needs to be done by the Scrum Team on that User Story).

If the Product Owner rejects a User Story, the Product Owner needs to provide his or her inputs about why the User Story was rejected (i.e., which elements of the Acceptance Criteria and/or Done Criteria were not met). Depending on the time remaining in the Sprint, after a User Story is rejected, and the reasons of the Product Owner for rejecting the User Story are provided, there will be two options available to the Scrum Team:

- Work on the rejected User Story in the current Sprint (based on inputs provided by the Product Owner) and then when all tasks required for the User Story are completed; the Scrum Team can re-submit the User Story to the Product Owner for approval during the same Sprint.
- Do not work on the rejected User Story in the same Sprint. In this case the User Story goes back into the Prioritized Product Backlog so that it can be assigned to another Sprint. The User Story may be assigned again to the same Scrum Team, or another Scrum Team may become responsible for that User Story in a future Sprint.

The Scrumboard can be maintained manually on paper or on a large whiteboard, but it can also be maintained electronically in a spreadsheet or using an AI-powered Scrum Project Tool. One Scrumboard is valid for the duration of one Sprint. The Scrum Team will create a new Scrumboard in the next Sprint.

10.1.1.4 Impediment Log*

An impediment is any hindrance or hurdle that reduces the productivity of the Scrum Team. Impediments must be identified, resolved, and removed if the team is to continue working effectively. Impediments can be internal to the team, such as inefficient workflow or lack of communication, or they can be external. Examples of external impediments might include software license issues or unnecessary documentation requirements. The Scrum framework, with its inherent transparency, facilitates the swift and easy identification of impediments. Failure to identify or deal with impediments can be very costly. Impediments should be formally recorded by the Scrum Master in an Impediment Log and should be discussed during Daily Standup Meetings and Sprint Review Meetings as appropriate.

10.1.1.5 Release Planning Schedule

Described in section 8.6.3.1.

10.1.1.6 Dependencies

Described in section 9.4.3.4.

10.1.1.7 Scrum Guidance Body Recommendations

In the *Create Deliverables* process, Scrum Guidance Body Recommendations may include best practices to effectively create deliverables, including preferred methods to conduct reviews, perform testing, create documentation, and so on. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

10.1.2 Tools

10.1.2.1 Team Expertise*

This refers to the collective expertise of the Scrum Team members in understanding the User Stories and tasks in the Sprint Backlog in order to create the final deliverables. Team Expertise is used to assess the inputs needed to execute the planned work of the project. This judgment and expertise are applied to all technical and management aspects of the project during the *Create Deliverables* process. Scrum Team members have the authority and responsibility to determine the best means for converting the Prioritized Product Backlog Items into finished deliverables or increments, without requiring the involvement of any business stakeholders outside the team. Additional expertise is available from the Scrum Guidance Body, as required.

10.1.2.2 Other Development Tools

Based on the specific requirements of the project and on industry specifications, other development tools can be used accordingly. Some examples are as follows:

1. Refactoring

Refactoring is a technique specific to software projects. The aim of this technique is to improve the maintainability of the existing code and make it simpler, more concise, and more flexible. Refactoring means improving the design of the present code without changing how the code behaves. It involves the following:

- Eliminating repetitive and redundant code
- Breaking methods and functions into smaller routines
- Clearly defining variables and method names
- Simplifying the code design
- Making the code easier to understand and modify

Regular refactoring optimizes code design a little at a time, over a period of time. Refactoring results in cleaner, more maintainable code, while preserving all functionalities.

2. Design Patterns

Design Patterns provide a formal way of recording a resolution to a design problem in a specific field of expertise. These patterns record both the process used and the actual resolution, which can later be reused to improve decision making and productivity.

10.1.2.3 Scrum Guidance Body Expertise

In *Create Deliverables* processes, Scrum Guidance Body Expertise could relate to documented rules and regulations development guidelines; and/or standards and best practices for creating the deliverables (e.g., guidance on how to conduct reviews or testing). There may also be a team of subject matter experts available who can provide guidance to the Scrum Team when creating the deliverables. This team could include lead architects, senior developers, security experts, or other experienced people. For more information on Scrum Guidance Body Expertise, see section 8.4.2.7.

10.1.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

10.1.3 Outputs

10.1.3.1 Sprint Deliverables*

At the end of each Sprint, a product increment or deliverable is completed. The deliverable should possess all features and functionality defined in the User Stories included in the Sprint and should have been tested successfully.

10.1.3.2 Updated Scrumboard*

The Scrumboard is updated regularly as the team completes tasks. However, at the end of the Sprint, the Scrumboard will be reset or wiped off and a new Scrumboard is created for the next Sprint. For more information on the Scrumboard, see sections 9.3.3.3 and 9.4.3.2.

10.1.3.3 Updated Impediment Log*

Described in section 10.1.1.4.

10.1.3.4 Unapproved Change Requests

Described in section 8.4.1.5.

10.1.3.5 Identified Risks

Described in section 8.4.3.4.

10.1.3.6 Mitigated Risks

As the Scrum Team executes the work of creating deliverables (according to the User Stories in the Prioritized Product Backlog), they carry out the mitigation actions that were defined to address any previously identified risks. Throughout the *Create Deliverables* process, the team documents any newly identified risks and mitigating actions taken. The record of project risks is a living document, continuously updated throughout the project by the team to reflect the current status of all risks. Additional information about managing risks is described in section 7.4.4.

10.1.3.7 Updated Dependencies

Described in section 8.5.2.6.

10.2 Conduct Daily Standup

In this process, a highly focused Daily Standup Meeting is conducted. This Time-boxed meeting is the forum for the Scrum Team to update each other on their progress and any impediments they may be facing.

Figure 10-6 shows all the inputs, tools, and outputs for the *Conduct Daily Standup* process.

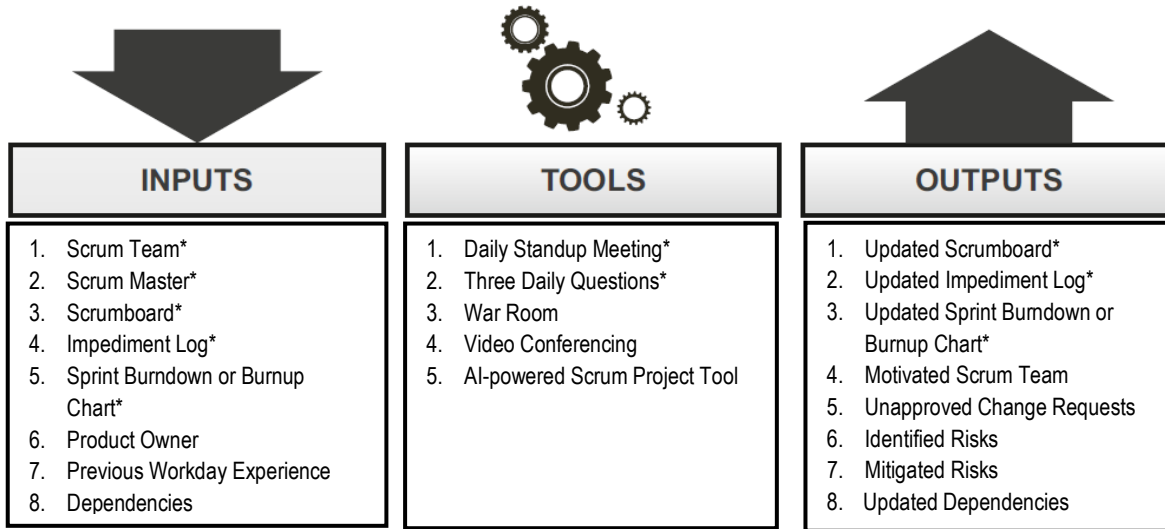


Figure 10-6: Conduct Daily Standup—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 10-7 is a data flow diagram for the “conduct daily standup” process. Displays how daily inputs feed into tools like standup meetings and lead to updated logs, charts, and team motivation.

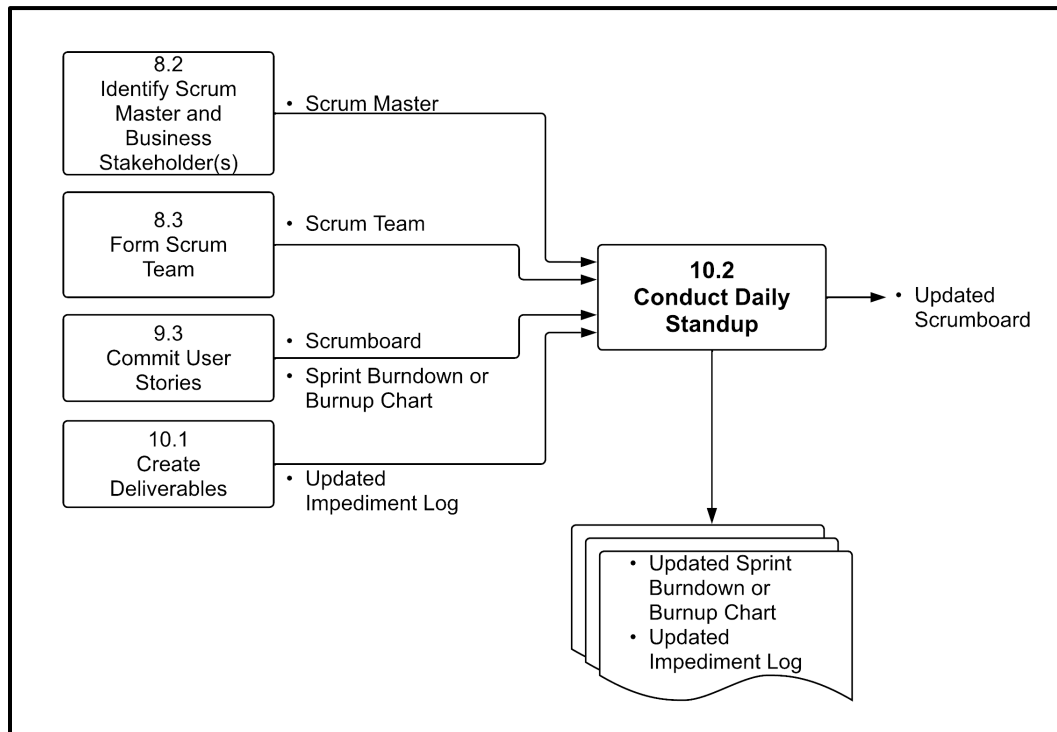


Figure 10-7: Conduct Daily Standup—Data Flow Diagram

10.2.1 Inputs

10

10.2.1.1 Scrum Team*

Described in section 8.3.3.1.

10.2.1.2 Scrum Master*

Described in section 8.2.3.1.

10.2.1.3 Scrumboard*

Described in sections 9.3.3.3, 9.4.3.2, and 10.1.1.3.

10.2.1.4 Impediment Log*

Described in section 10.1.1.4.

10.2.1.5 Sprint Burndown or Burnup Chart

Described in section 9.6.3.3.

10.2.1.6 Product Owner

Described in section 8.1.3.1.

10.2.1.7 Previous Work Day Experience

The Scrum Team members give status updates to fellow team members in the Daily Standup Meeting. This session is called standup because members stand throughout the meeting. Team members formulate their achievements and experiences from the previous workday. This experience is an important input to the Daily Standup Meeting.

10.2.1.8 Dependencies

Described in section 9.4.3.4.

10.2.2 Tools

10.2.2.1 Daily Standup Meeting*

The Daily Standup Meeting is a short daily meeting, Time-boxed to 15 minutes. Team members assemble to report their previous day's progress in the Sprint and plan the current day's activities. The meeting duration is intentionally short (which standing up at the meetings helps to reiterate) and all members of the Scrum Team are expected to attend. The meeting should not be cancelled or delayed if one or more team members are not able to attend. The Scrum Team manages the Daily Standup Meeting while the Scrum Master facilitates the meeting, as needed.

In the meeting, each Scrum Team member provides answers to the three daily questions (see section 10.2.2.2). Discussions between the Scrum Master and the team or between some Scrum Team members are encouraged, but such discussions happen after the meeting to ensure that the Daily Standup Meeting is kept short.

10.2.2.2 Three Daily Questions*

In the Daily Standup Meeting facilitated by the Scrum Master, each Scrum Team member provides information in the form of answers to the following three specific questions:

1. What have I done since the last meeting?
2. What do I plan to do before the next meeting?
3. What impediments or obstacles (if any) am I currently facing?

By focusing on these three questions, the entire team can have a clear understanding of the work status for the current Sprint. Occasionally, other items may be discussed, but this is kept to a minimum in light of the Time-boxed nature of the meeting.

It is highly recommended that the first two questions should be answered by team members in a quantifiable manner when possible, instead of qualitative lengthy answers. Team members can organize additional meetings after the Daily Standup Meeting to address items that need additional discussion.

10.2.2.3 War Room

When applying Scrum practices on a project, it is preferable for the team to be colocated, with all team members working at the same location. The term commonly used to describe this place is the War Room. Normally, the room is designed in such a way that team members can move around freely, work, and communicate easily because they are located in close proximity to each other. Typically index cards, sticky notes, and other low-tech, high-touch tools are made available in the room to facilitate workflow, collaboration, and problem solving.

The room is sometimes noisy due to team conversations, but these conversations contribute to the team's progress. A good War Room is cubicle free and allows the entire team to sit together ensuring face-to-face communication, which leads to team building and openness. The War Room is also ideal for conducting Daily Standup Meetings. Business stakeholder members from other Scrum Teams could also attend the War Room and discuss relevant issues, as needed.

10

10.2.2.4 Video Conferencing

In real-life situations, it may not always be possible for the entire Scrum Team to be colocated. In such cases, it becomes imperative to use video conferencing tools to enable face-to-face communication. For more information on effective collaboration in distributed teams, see section 2.5.3.

10.2.2.5 AI-powered Scrum Project Tool

Described in section 2.4.4

10.2.3 Outputs

10.2.3.1 Updated Scrumboard*

The Scrumboard continues to be updated regularly as the team completes tasks. For more information on the Scrumboard, see sections 9.3.3.3 and 9.4.3.2.

Figure 10-8 is a Vabro interface showing scrumboard usage in implementation. Visualizes real-time progress tracking, team assignments, and task movement across sprint phases using ai-powered features.

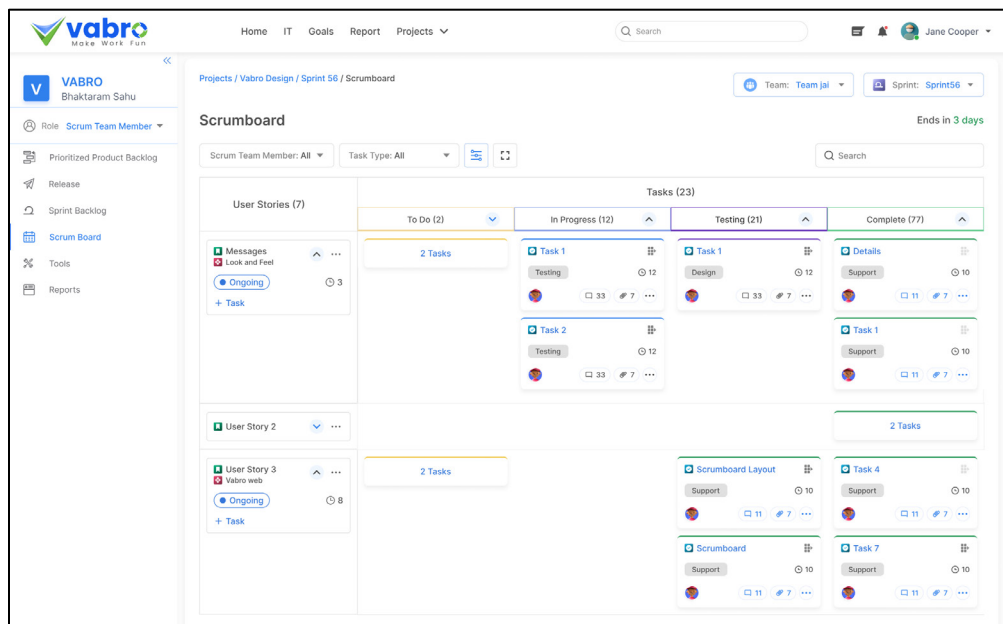


Figure 10-8: Use of Scrumboard (Source: Vabro)

Figure 10-9 is a Jira-based scrumboard showcasing its role in managing a Scrum initiative. Features task status columns, progress tracking, and backlog integration for sprint execution.

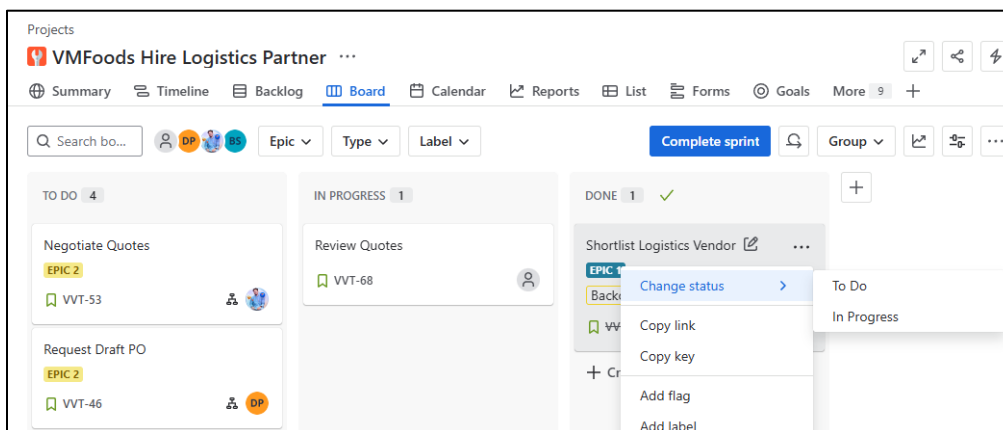


Figure 10-9: Role of Scrumboard in Scrum (Source: Jira)

10.2.3.2 Updated Impediment Log*

Described in section 10.1.1.4.

10.2.3.3 Updated Sprint Burndown or Burnup Chart

The Sprint Burndown Chart should be updated daily to show the progress that has been made by the Scrum Team and to also allow for the detection of estimates that may have been incorrect. If the Sprint Burndown Chart shows that the Scrum Team is not on track to finish the tasks in the Sprint on time, the Scrum Master should identify any obstacles or impediments to successful completion and try to remove them. For more information on the Sprint Burndown Chart, see section 9.6.3.3.

10.2.3.4 Motivated Scrum Team

The Daily Standup Meetings propagate the idea that each member of the team is important, and that he/she is a major contributor to the project. This can help improve individual and team morale. The practice of self-organizing teams can also help improve overall motivation, lead to enhanced performance of the team, and improve the quality of the deliverables being produced. For more information on the Scrum Team, see section 8.3.3.1.

10.2.3.5 Unapproved Change Requests

Described in section 8.4.1.5.

10.2.3.6 Identified Risks

Described in section 8.4.3.4.

10.2.3.7 Mitigated Risks

Described in section 10.1.3.6.

10.2.3.8 Updated Dependencies

Described in section 8.5.2.6.

10.3 Refine Prioritized Product Backlog

In this process, the Product Owner continuously updates and maintains the Prioritized Product Backlog. A Prioritized Product Backlog Review Meeting may be held, during which any changes or updates to the Product Backlog are discussed and incorporated into the Prioritized Product Backlog as appropriate.

In order to keep the Prioritized Product Backlog up to date with any change in requirements and/or priorities, the Product Owner continually works with the customer and other business stakeholders to capture and understand any changes in their needs.

To ensure that the set of User Stories the Product Owner would like the team to commit to in the next Sprint will be ready for commitment, the Product Owner refines existing Epics and User Stories in the Prioritized Product Backlog, and ensures that the User Stories satisfy the Definition of Ready.

As part of refining the Prioritized Product Backlog, the Product Owner also works with the Scrum Team to get feedback and questions related to the updates in the Prioritized Product Backlog, potentially including estimates. If changes in requirements and/or the overall progress of the Scrum Team require changes to the Release Schedule and/or the business justification, the Product Owner will also make these changes during this process. This is the process where the Product Owner will spend most of his/her time.

Figure 10-10 shows all the inputs, tools, and outputs for the *Refine Prioritized Product Backlog* process.

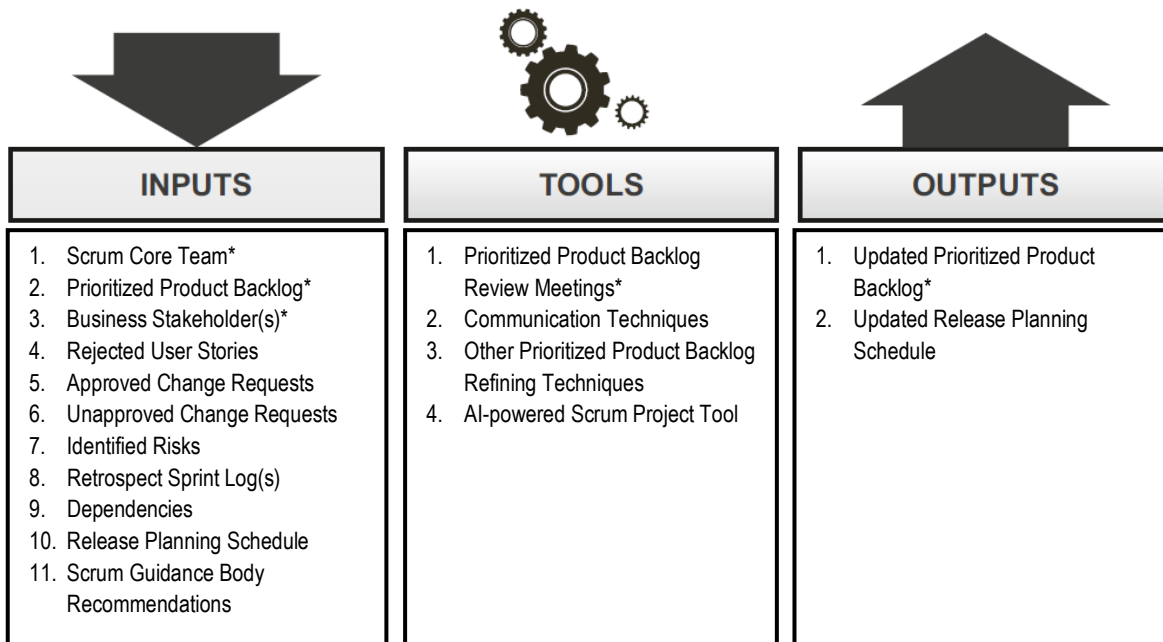


Figure 10-10: Refine Prioritized Product Backlog—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 10-11 is a data flow diagram for the “Refine Prioritized Product Backlog” process. Tracks how changes in user stories and backlog items flow through refinement to updated backlog versions.

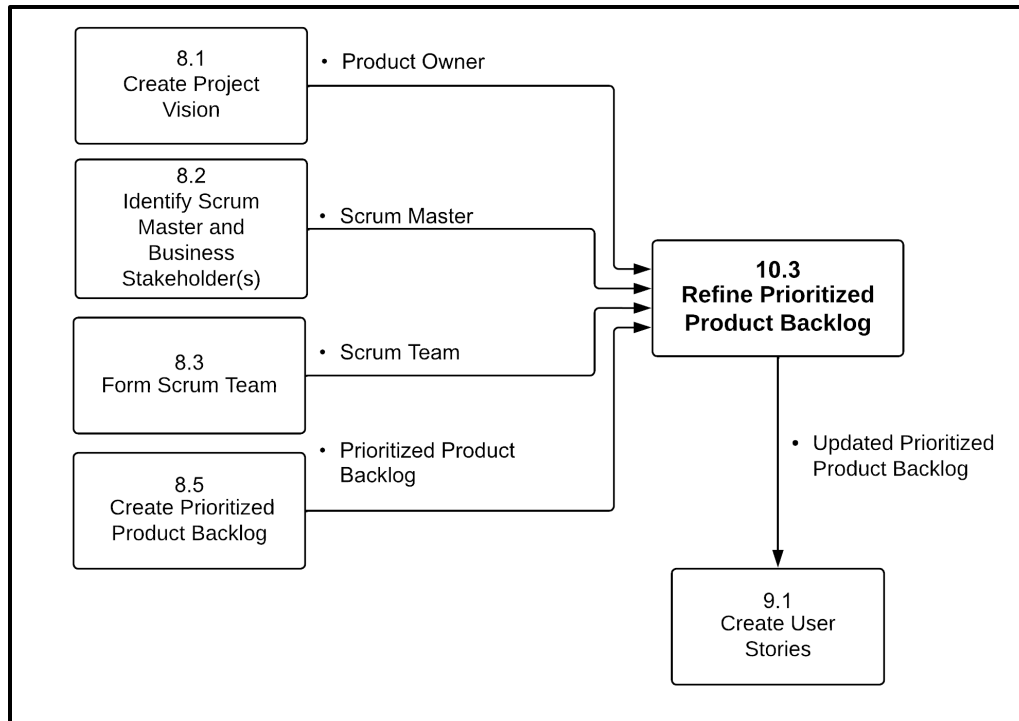


Figure 10-11: Refine Prioritized Product Backlog—Data Flow Diagram

10.3.1 Inputs

10

10.3.1.1 Scrum Core Team*

Described in section 3.2.1.

10.3.1.2 Prioritized Product Backlog*

Described in section 8.5.3.1.

10.3.1.3 Business Stakeholder(s)*

In order to keep the Prioritized Product Backlog up to date with any change in requirements and/or priorities, the Product Owner continually works with the customer and other business stakeholders to capture and understand any changes in their needs. For more information on business stakeholders, see section 3.2.2.

10.3.1.4 Rejected User Stories

In cases where a User Story does not meet the Acceptance Criteria, it is considered a rejected user story. The rejected user stories are normally not kept in a separate list. They simply remain in the Prioritized Product Backlog similar to other user stories in the backlog, and do not get marked as done so that they can be reprioritized in the *Refine Prioritized Product Backlog* process and be considered for development in the next or other future Sprint.

10.3.1.5 Approved Change Requests

Described in section 8.4.1.4.

10.3.1.6 Unapproved Change Requests

Described in section 8.4.1.5.

10.3.1.7 Identified Risks

Described in section 8.4.3.4.

10.3.1.8 Retrospect Sprint Log(s)

Described in section 11.2.3.4.

10.3.1.9 Dependencies

Described in section 9.4.3.4.

10.3.1.10 Release Planning Schedule

Described in section 8.6.3.1.

10.3.1.11 Scrum Guidance Body Recommendations

In the *Refine Prioritized Product Backlog* process, recommendations from the Scrum Guidance Body may include best practices on how to systematically understand and collate requirements from business stakeholder(s) and Scrum Teams and then properly prioritize the Product Backlog and finally communicate updates to all relevant persons involved with the Scrum project. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

10.3.2 Tools

10.3.2.1 Prioritized Product Backlog Review Meetings*

The Product Owner may have separate meetings with relevant business stakeholder(s), the Scrum Master, and the Scrum Team to ensure that he or she has enough information to appropriately update the Prioritized Product Backlog during the *Refine Prioritized Product Backlog* process. The intent of these Prioritized Product Backlog Review Meetings is to ensure that the User Stories and their corresponding Acceptance Criteria are understood, and are written properly by the Product Owner so that they reflect the stated customer requirements and priorities; User Stories are understood by everyone in the Scrum Team; and that high-priority User Stories are well-refined so that the Scrum Team can properly estimate and commit to such User Stories. The Prioritized Product Backlog Review Meetings also ensure that irrelevant User Stories are removed and any Approved Change Requests or identified risks are incorporated into the Prioritized Product Backlog.

10.3.2.2 Communication Techniques

10

Scrum principles and practices promote accurate and effective communication primarily through colocation of the Scrum Team. Scrum also favors informal, face-to-face interactions over formal written communications. When a Scrum Team needs to be distributed, the Scrum Master should ensure that effective communication techniques and perhaps an AI-powered Scrum Project Tool are available for use so that distributed teams can self-organize and work effectively. For more information on distributed teams, see section 2.5.3.

10.3.2.3 Other Prioritized Product Backlog Refining Techniques

Some other Prioritized Product Backlog refining tools include many of the same tools used for the following processes:

- *Develop Epic(s)*—Described in section 8.4.2.
- *Create Prioritized Product Backlog*—Described in section 8.5.2.
- *Conduct Release Planning*—Described in section 8.6.2.
- *Create User Stories*—Described in section 9.1.2.

- *Estimate User Stories*—Described in section 9.2.2.
- *Commit User Stories*—Described in section 9.3.2.
- *Identify Tasks*—Described in section 9.4.2.
- *Estimate Tasks*—Described in section 9.5.2.

10.3.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 10-12 shows Vabro interface for product backlog refinement. Features include item scoring, stakeholder feedback, and ai-generated suggestions to improve backlog prioritization.

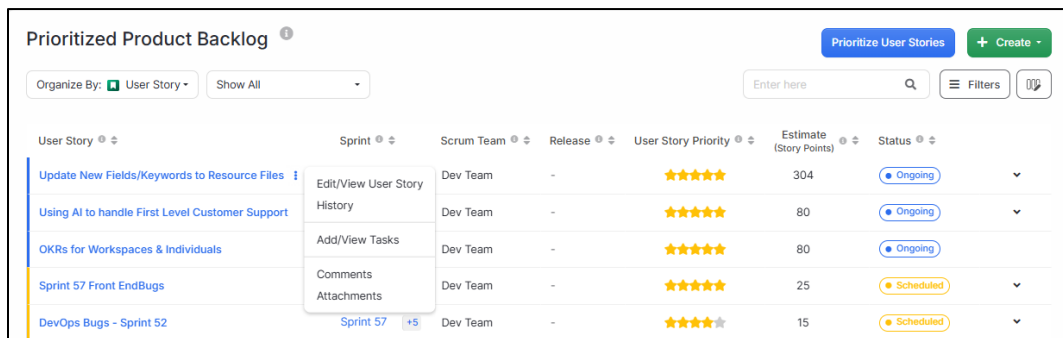


Figure 10-12: Refining Product Backlog in Scrum (Source: Vabro)

Figure 10-13 is a Jira interface for backlog refinement. Visual elements support backlog item editing, prioritization adjustments, and collaboration across scrum roles.

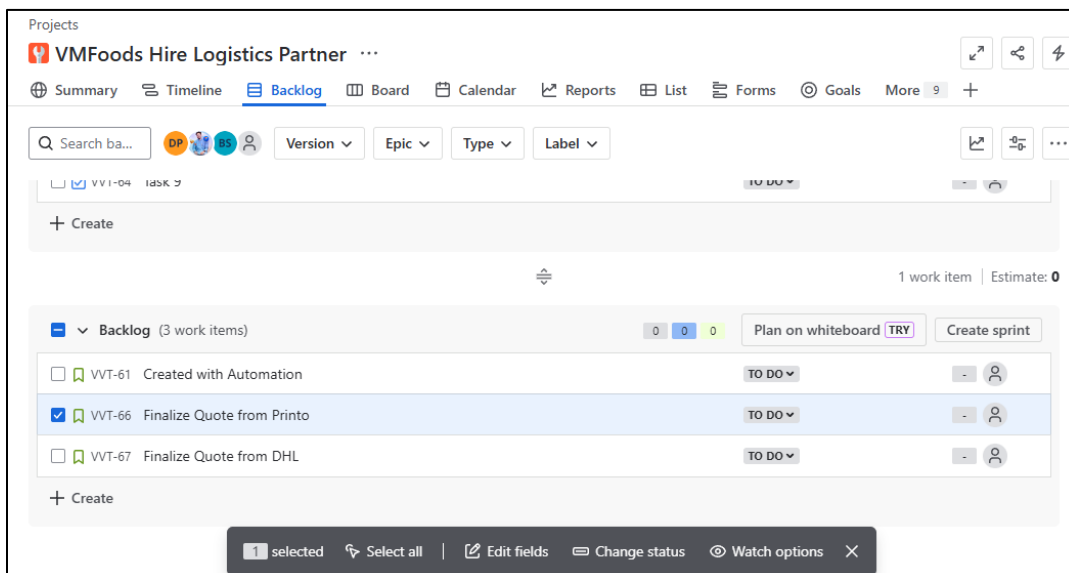


Figure 10-13: Product Backlog Refinement in Scrum (Source: Jira)

10.3.3 Outputs

10.3.3.1 Updated Prioritized Product Backlog*

The Prioritized Product Backlog may be updated with new or updated User Stories; work related to new Change Requests or identified risks; or to reflect the reprioritization of existing User Stories. For more information on the Prioritized Product Backlog, see section 8.5.3.1.

10.3.3.2 Updated Release Planning Schedule

The Release Planning Schedule may be updated to reflect the impact of new or changed User Stories in the Prioritized Product Backlog. For more information on the Release Planning Schedule, see section 8.6.3.1.

10.4 Implement Phase Data Flow Diagram

Figure 10-14 is a comprehensive data flow diagram representing the entire “implement” phase. Maps how each process and tool interact, from planning to sprint execution, to produce working deliverables and updated scrum artifacts.

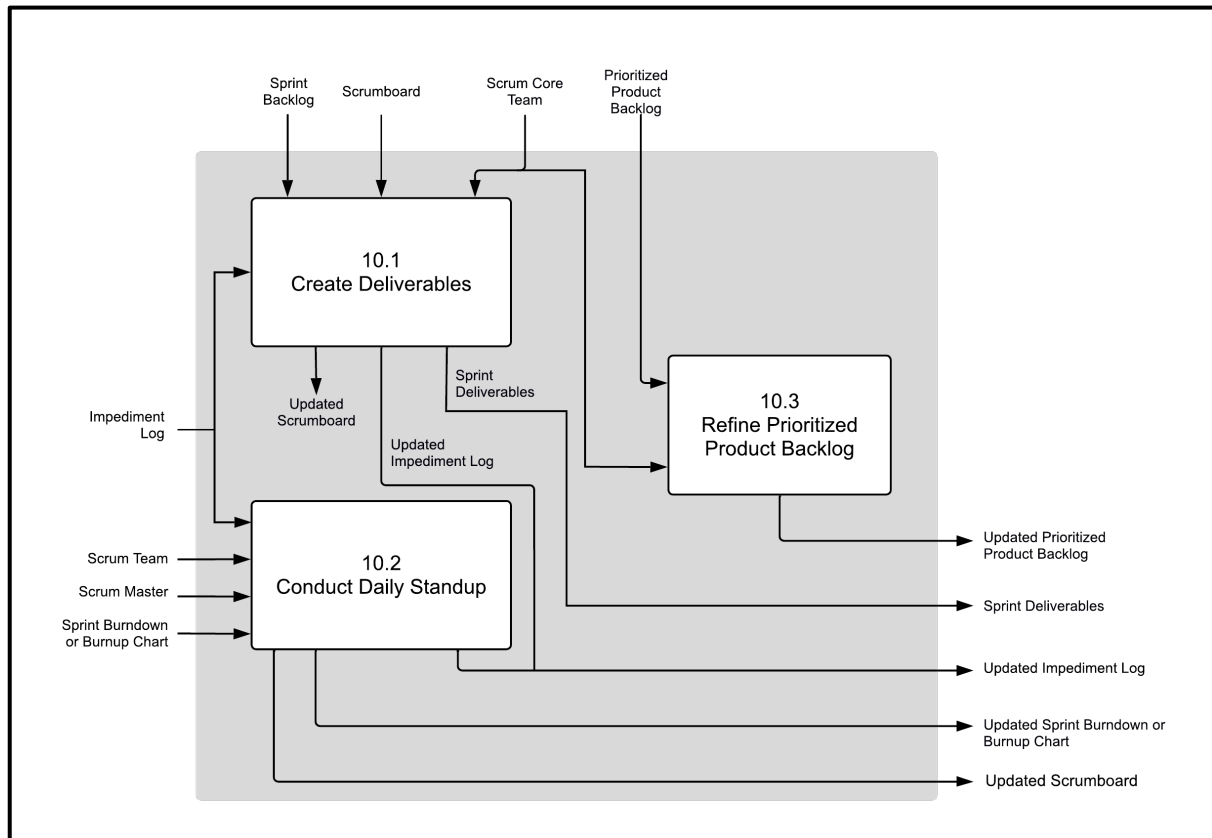


Figure 10-14: Implement Phase—Data Flow Diagram

11. REVIEW AND RETROSPECT

The Review and Retrospect phase is concerned with reviewing the deliverables and the work that has been done and determining ways to improve the practices and methods used to complete the project work. In large organizations the *Review* and *Retrospect* processes may also include convening Scrum of Scrums Meetings.

Review and Retrospect, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in *any* industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Product Owners, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter.

This chapter is written from the perspective of one Scrum Team working on one Sprint to produce potentially shippable deliverables, which could be part of a larger project, program, or portfolio. Additional information pertaining to Scaling Scrum for Large Projects is available in chapter 13. Additional information pertaining to Scaling Scrum for the Enterprise can be found in chapter 14.

Review and Retrospect is the third and final of the three phases that are repeated in every Sprint.

The Product Owner and relevant business stakeholders review the deliverables the team has created and provide feedback. The Product Owner evaluates each User Story and determines whether it meets its respective Acceptance Criteria and accordingly either accepts or rejects it.

As the final part of a Sprint, the Scrum Team determines ways to continually improve its work.

It is also important to realize that although all phases and processes are defined uniquely in the SBOK® Guide, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to overlap some phases and/or processes, depending on the specific requirements of each project.

Figure 11-1 provides a high-level overview of the “review and retrospect” phase. Summarizes key processes including demonstrating deliverables, validating user stories, conducting retrospectives, and incorporating feedback to improve future sprints.

11.1 Demonstrate and Validate Sprint—In this process, the Scrum Team demonstrates the Sprint Deliverables to the Product Owner in a Sprint Review Meeting. The purpose of this meeting is to secure approval of the Sprint User Stories by the Product Owner.

This process is not only an important quality element in a Scrum project, but it is also a key element to maintain stakeholder engagement. The business stakeholders are encouraged to participate in the Sprint Review Meeting to gain first-hand knowledge of the Product or Service and its progress, and to provide feedback. Stakeholder feedback is an important input to future Sprints.

11.2 Retrospect Sprint—In this process, the Scrum Master and Scrum Team meet to discuss the lessons learned throughout the Sprint. This information is documented as lessons learned which will be applied to future Sprints. As a result, there may be agreed actionable improvements or updated Scrum Guidance Body Recommendations. This process is an essential component of the continuous improvement in Scrum.

11.1 Demonstrate and Validate Sprint	11.2 Retrospect Sprint
INPUTS 1. Scrum Core Team* 2. Sprint Deliverables* 3. Sprint Backlog* 4. Done Criteria* 5. User Story Acceptance Criteria* 6. Business Stakeholder(s) 7. Release Planning Schedule 8. Identified Risks 9. Dependencies 10. Scrum Guidance Body Recommendations TOOLS 1. User Story Acceptance/Rejection* 2. Sprint Review Meetings* 3. Earned Value Analysis 4. Scrum Guidance Body Expertise 5. AI-powered Scrum Project Tool OUTPUTS 1. Accepted User Stories* 2. Rejected User Stories 3. Updated Risks 4. Earned Value Analysis Results 5. Updated Release Planning Schedule	INPUTS 1. Scrum Master* 2. Scrum Team* 3. Outputs from <i>Demonstrate and Validate Sprint</i>* 4. Product Owner 5. Scrum Guidance Body Recommendations TOOLS 1. Retrospect Sprint Meeting* 2. ESVP 3. Speed Boat 4. Metrics and Measuring Techniques 5. Scrum Guidance Body Expertise 6. AI-powered Scrum Project Tool OUTPUTS 1. Agreed Actionable Improvements* 2. Assigned Action Items and Due Dates 3. Proposed Non-Functional Items for Prioritized Product Backlog 4. Retrospect Sprint Log(s) 5. Scrum Team Lessons Learned 6. Updated Scrum Guidance Body Recommendations

Figure 11-1: Review and Retrospect Overview

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 11-2 below shows the mandatory inputs, tools, and outputs for processes in the Review and Retrospect phase.

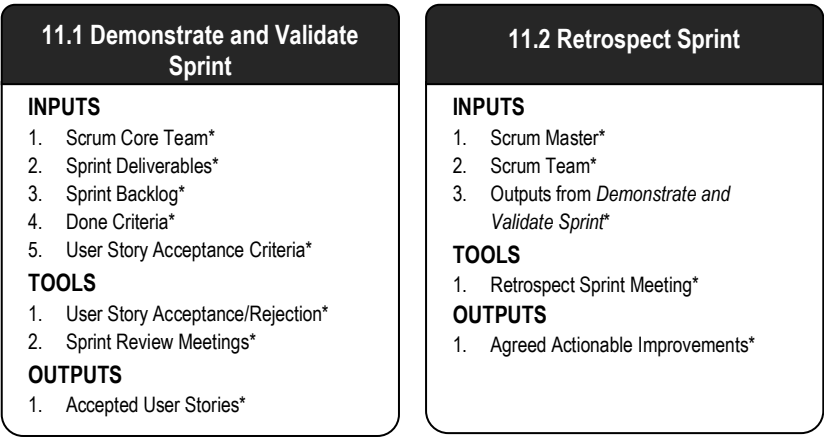


Figure 11-2: Review and Retrospect Overview (Essentials)

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

11.1 Demonstrate and Validate Sprint

In this process, the Scrum Team demonstrates the Sprint Deliverables to the Product Owner in a Sprint Review Meeting. The purpose of this meeting is to secure approval of the Sprint User Stories by the Product Owner.

This process is not only an important quality element in a Scrum project, but it is also a key element to maintain stakeholder engagement. The business stakeholders are encouraged to participate in the Sprint Review Meeting to gain first-hand knowledge of the Product or Service and its progress, and to provide feedback. Business stakeholder feedback is an important input to future Sprints.

Figure 11-3 shows all the inputs, tools, and outputs for the *Demonstrate and Validate Sprint* process.

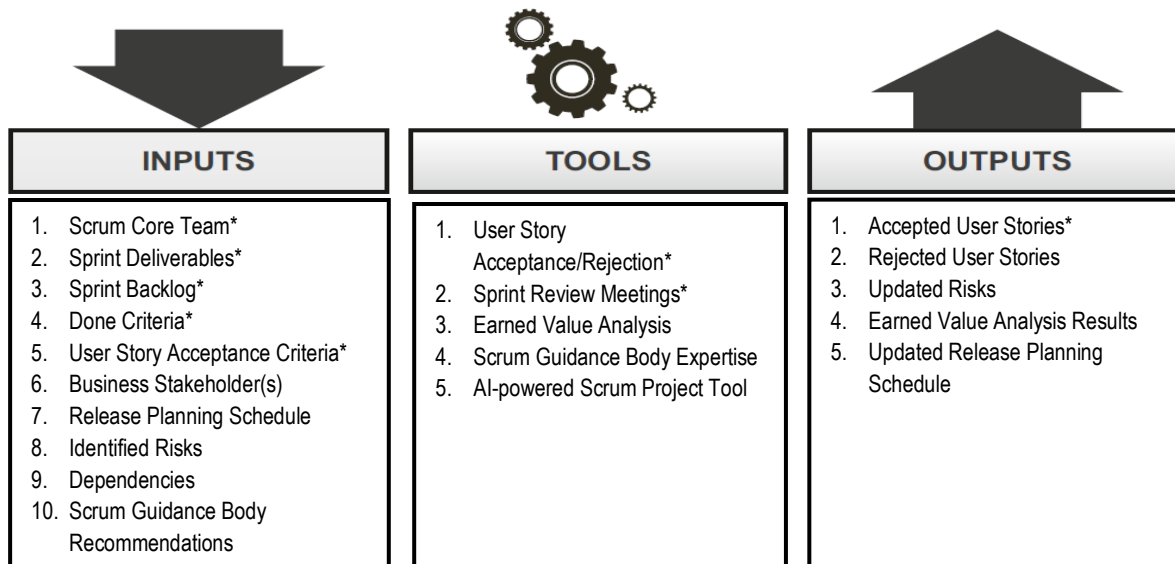


Figure 11-3: Demonstrate and Validate Sprint—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 11-4 shows a data flow diagram for the “demonstrate and validate sprint” process. Illustrates how feedback from sprint reviews informs outputs like accepted stories, risk updates, and improved planning.

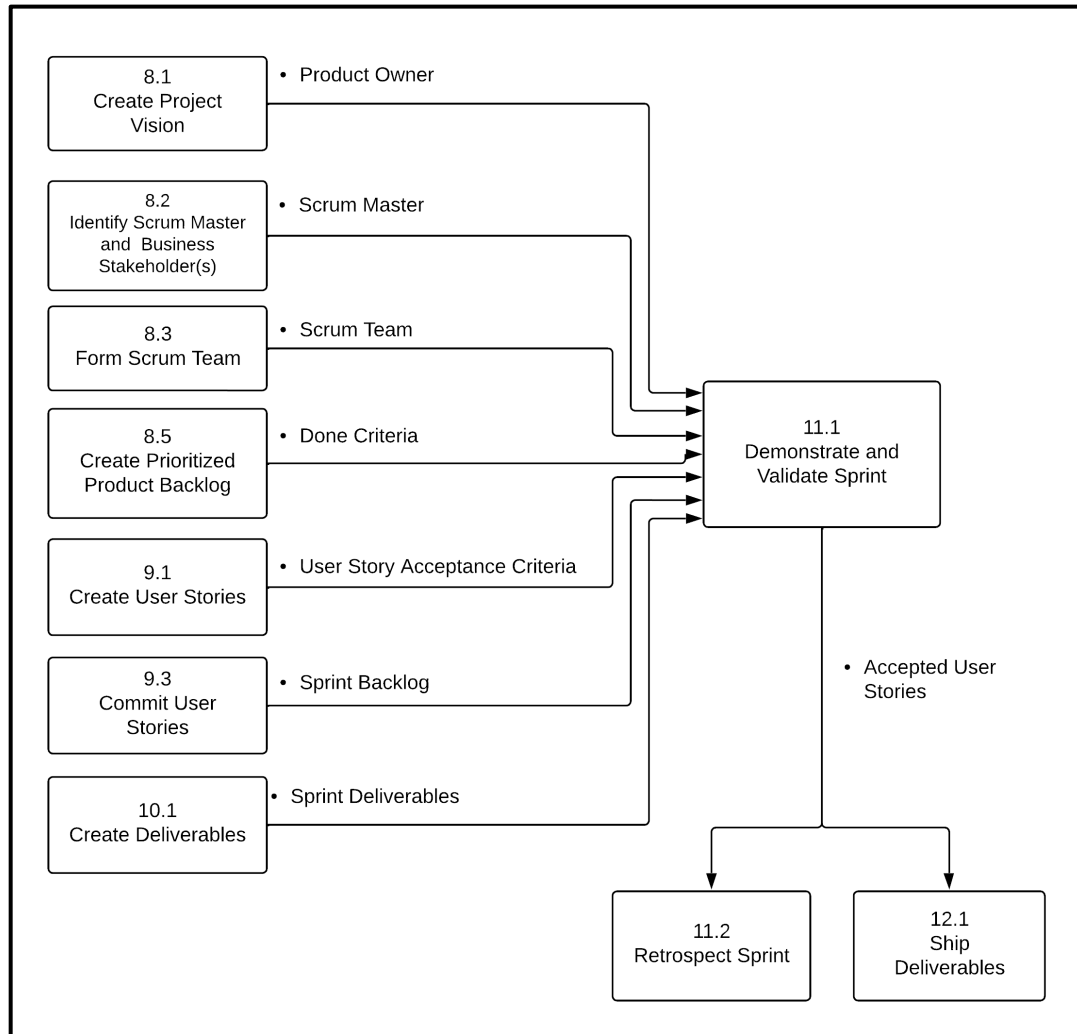


Figure 11-4: Demonstrate and Validate Sprint—Data Flow Diagram

11.1.1 Inputs

11.1.1.1 Scrum Core Team*

Described in section 3.2.1.

11.1.1.2 Sprint Deliverables*

Described in section 10.1.3.1.

11.1.1.3 Sprint Backlog*

Described in sections 9.3.3.2 and 9.6.3.1.

11.1.1.4 Done Criteria*

Described in section 8.5.3.2.

11.1.1.5 User Story Acceptance Criteria*

Described in sections 9.1.3.2 and 9.4.1.3.

11.1.1.6 Business Stakeholder(s)

Described in section to 8.2.3.2.

11.1.1.7 Release Planning Schedule

Described in section 8.6.3.1.

11.1.1.8 Identified Risks

Described in section to 8.4.3.4.

11.1.1.9 Dependencies

Described in section 9.4.3.4.

11.1.1.10 Scrum Guidance Body Recommendations

In the *Demonstrate and Validate Sprint* process, Scrum Guidance Body Recommendations may include best practices about how to conduct Sprint Review Meetings and how to evaluate results from Earned Value Analysis. Also, there may be guidance available on how to share experiences with other people in the Scrum Core Team and with other Scrum Teams in the project. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

11.1.2 Tools

11.1.2.1 User Story Acceptance/Rejection*

After User Stories are completed by the Scrum Team, they are made available to the Product Owner for review. The Product Owner can review the User Stories as soon as each is complete or can review them all in a Sprint Review Meeting convened at the end of the Sprint. The Product Owner accepts the User Stories that satisfy the User Story Acceptance Criteria and the Done Criteria and rejects those that do not meet the criteria, with feedback about why a User Story was rejected. If time is still available in the current Sprint, rejected User Stories are made available to the team to address the reasons for rejection and are provided again to the Product Owner in the same Sprint for an additional review. At the end of a Sprint, any remaining rejected User Stories would remain on the Prioritized Product Backlog to be considered for completion in a future Sprint.

11

11.1.2.2 Sprint Review Meeting*

Sprint Review Meetings are convened at the end of every Sprint. The Scrum Core Team members and relevant business stakeholder(s) participate in Sprint Review Meetings to present the deliverables. The Scrum Team demonstrates the User Stories and deliverables created as part of the Sprint, including the new functionalities or products created. User Story deliverables that have already been previously approved by the Product Owner prior to the Sprint Review Meeting are also demonstrated by the team during this meeting to ensure that the business stakeholder(s) also review the respective functionality. The Product Owner and business stakeholder(s) inspect all the deliverables and determine if changes need to be made in a subsequent Sprint. By the end of the Sprint Review Meeting, all User Stories in the Sprint are considered and some are approved, and others are rejected by the Product Owner based on whether they meet the Acceptance Criteria and the Done Criteria.

11.1.2.3 Earned Value Analysis

Described in section 4.6.1.

11.1.2.4 Scrum Guidance Body Expertise

In the *Demonstrate and Validate Sprint* process, Scrum Guidance Body expertise could relate to documented best practices about how to conduct Sprint Review Meetings. There may also be some experts available to help provide guidance on how to better facilitate a Sprint Review Meeting. For more information on Scrum Guidance Body Expertise, see section 8.4.2.7.

11.1.2.5 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 11-5 is a Vabro interface showing how ai helps manage user story validation and approvals. Includes status indicators, reviewer inputs, and acceptance criteria tracking for each user story.

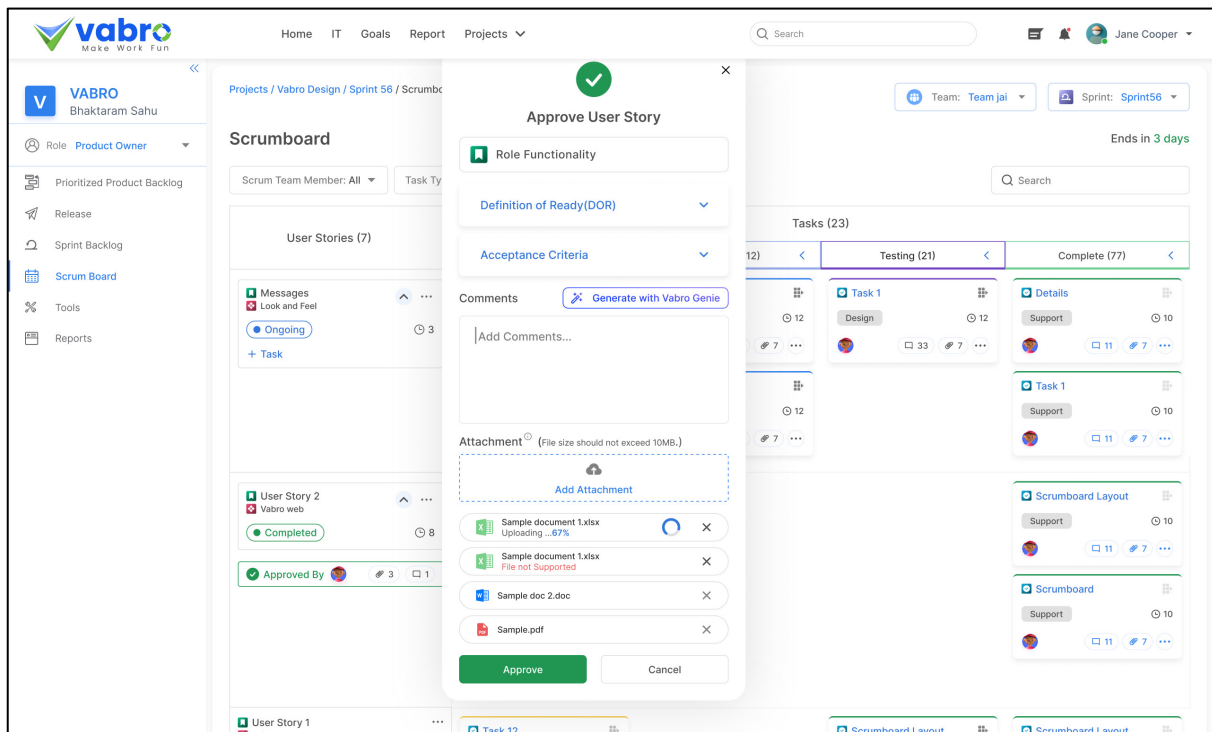


Figure 11-5: User Story Review and Approval (Source: Vabro)

11.1.3 Outputs

11.1.3.1 Accepted User Stories*

The objective of a Sprint is to create potentially shippable deliverables (or product increments) that meet the User Story Acceptance Criteria defined by the customer and Product Owner. User Stories that meet the Acceptance Criteria are formally accepted by the Product Owner. These accepted User Story deliverables may be released to the customer, if desired. A list of the accepted User Stories is maintained and updated after each Sprint Review Meeting.

11.1.3.2 Rejected User Stories

If User Stories do not meet the Acceptance Criteria, they are considered incomplete and are rejected by the Product Owner. Rejected User Stories get added back into the Prioritized Product Backlog so that they can be considered as part of a subsequent Sprint. The work on deliverables associated with rejected User Stories may be done by any Scrum Team to which those User Stories are assigned to in the future.

Since some work might already be done in creating the deliverables of these rejected User Stories, if the partially completed deliverables are assigned for completion in a future Sprint, the future estimate of those User Stories may be lesser than what was estimated for the original User Stories. However, in some cases, Scrum Teams can choose to completely ignore the deliverables associated with the rejected User Stories and consider the upcoming work as new User Stories.

11.1.3.3 Updated Risks

Described in section to 8.4.3.4.

11.1.3.4 Earned Value Analysis Results

Described in section 4.6.1.

11.1.3.5 Updated Release Planning Schedule

Described in section to 8.6.3.1.

11.2 Retrospect Sprint

In this process, the Scrum Master and Scrum Team meet to discuss the lessons learned throughout the Sprint. This information is documented as lessons learned which will be applied to future Sprints. As a result, there may be agreed actionable improvements or updated Scrum Guidance Body Recommendations. This process is an essential component of the continuous improvement in Scrum.

Figure 11-6 shows all the inputs, tools, and outputs for the *Retrospect Sprint* process.

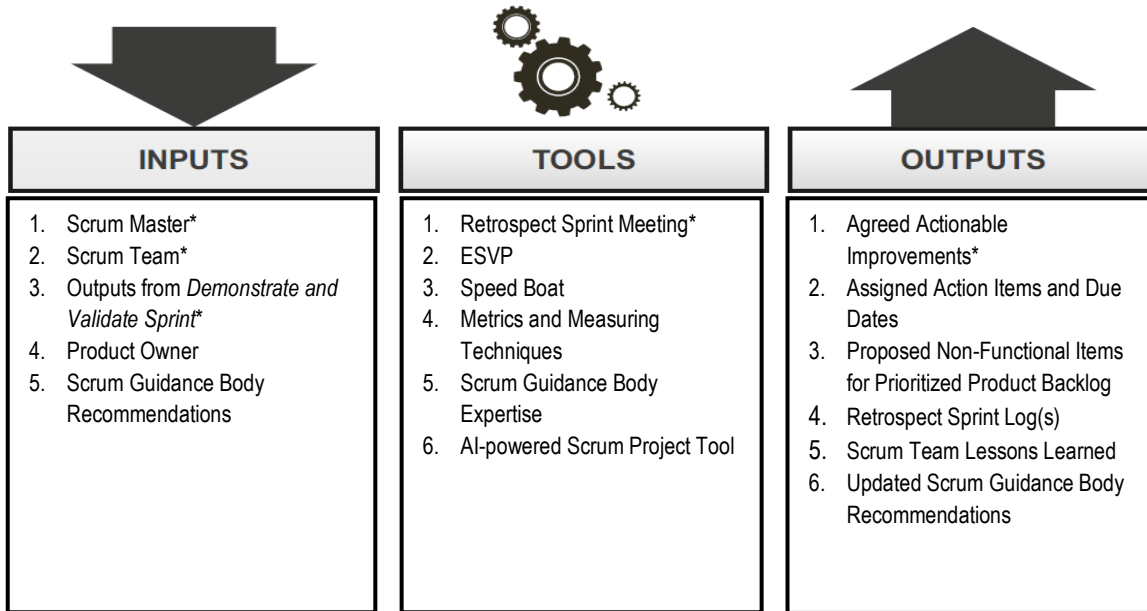


Figure 11-6: Retrospect Sprint—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 11-7 is a data flow diagram for the “retrospect sprint” process. Tracks how retrospective inputs result in continuous improvement suggestions, action items, and updates to scrum processes.

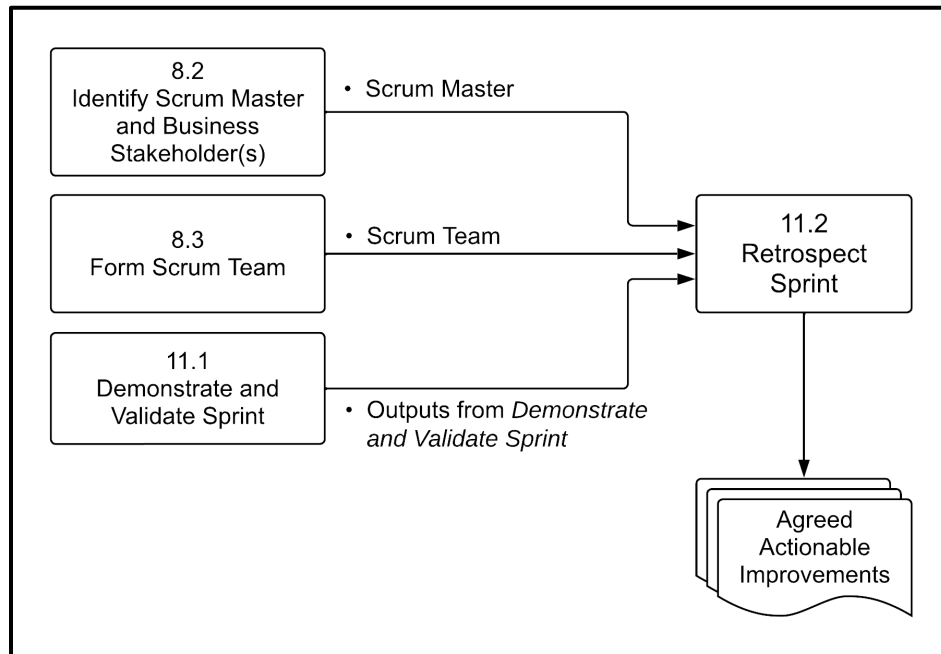


Figure 11-7: Retrospect Sprint—Data Flow Diagram

11.2.1 Inputs

11.2.1.1 Scrum Master*

Described in section 8.2.3.1.

11.2.1.2 Scrum Team*

Described in section 8.3.3.1.

11.2.1.3 Outputs from Demonstrate and Validate Sprint*

The outputs from the *Demonstrate and Validate Sprint* process provide valuable insight while performing the *Retrospect Sprint* process. For more information on the *Demonstrate and Validate Sprint* outputs, see section 11.1.3.

11.2.1.4 Product Owner

Described in section 8.1.3.1.

11.2.1.5 Scrum Guidance Body Recommendations

The Scrum Guidance Body may provide guidelines for conducting Retrospect Sprint Meetings, including suggestions for tools to be utilized and documentation or deliverables expected from the meetings. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

11.2.2 Tools

11.2.2.1 Retrospect Sprint Meeting*

The Retrospect Sprint Meeting is an essential element of the 'inspect-adapt' Scrum framework and it is the last step in a Sprint. All Scrum Team members attend the meeting, which is facilitated or moderated by the Scrum Master. It is recommended but not required for the Product Owner to attend. One team member acts as the scribe and documents discussions and items for future action. It is essential to hold this meeting in an open and relaxed environment to encourage full participation by all team members. Discussions in the Retrospect Sprint Meeting encompass both what went wrong and what went right.

Primary objectives of the meeting are to identify three specific items:

1. Things the team needs to keep doing: best practices
2. Things the team needs to begin doing: process improvements
3. Things the team needs to stop doing: process problems and bottlenecks

These areas are discussed, and a list of agreed actionable improvements is created.

11.2.2.2 Explorer, Shopper, Vacationer, Prisoner (ESVP)

ESVP is an exercise that can be conducted at the start of the Retrospect Sprint Meeting to understand the mindset of the participants and set the tone for the meeting. Attendees are asked to anonymously indicate which best represents how they feel regarding their participation in the meeting.

- Explorer—Wants to participate in and learn everything discussed in the retrospective
- Shopper—Wants to listen to everything and choose what he takes away from the retrospective
- Vacationer—Wants to relax and be a tourist in the retrospective
- Prisoner—Wants to be elsewhere and is attending the retrospective because it is required

The Scrum Master then collates the responses, prepares, and shares the information with the group.

11.2.2.3 Speed Boat

Speed Boat is a technique that can be used to conduct the Retrospect Sprint Meeting. Team members play the role of the crew on a speed boat. The boat must reach an island, which is symbolic of the project vision. The attendees use sticky notes to record engines and anchors. Engines help them reach the island, while anchors hinder them from reaching the island. This exercise is Time-boxed to a few minutes. Once all items are documented, the information is collated, discussed, and prioritized by way of a voting process. Engines are recognized and mitigation actions are planned for the anchors, based on priority.

11.2.2.4 Metrics and Measuring Techniques

Various metrics can be used to measure and contrast the team's performance in the current Sprint to their performance in previous Sprints. Some examples of these metrics include:

- Team velocity—Number of story points done in a given Sprint
- Done success rate—Percentage of story points that are done versus those that are committed
- Estimation effectiveness—Number or percentage of deviations between estimated and actual time spent on tasks and User Stories
- Review feedback ratings—Quantitative or qualitative feedback ratings solicited from team members that provide a measurement of team performance
- Team morale ratings—Results from self-assessments of team member morale
- Peer feedback—360-degree feedback mechanisms, used to solicit constructive criticism and insight into team performance
- Progress to release or launch—Business value provided in each release, as well as value represented by the current progress towards a release; this contributes to team motivation and to the level of work satisfaction

11.2.2.5 Scrum Guidance Body Expertise

In the *Retrospect Sprint* process, Scrum Guidance Body expertise could relate to best practices on how to conduct Retrospect Sprint Meetings. There may also be experts available to provide guidance on how to use the tools in this process to deliver agreed and actionable improvements for future Sprints. For more information on Scrum Guidance Body Expertise, see section 8.4.2.7.

11.2.2.6 AI-powered Scrum Project Tool

Described in section 2.4.4

11.2.3 Outputs

11.2.3.1 Agreed Actionable Improvements*

Agreed actionable improvements are the primary output of the *Retrospect Sprint* process. This is a list of actionable items that the team has come up with to address problems and improve processes in order to enhance team performance in future Sprints.

11.2.3.2 Assigned Action Items and Due Dates

Once the agreed actionable improvements have been elaborated and refined, action items to implement the improvements may be assigned by the Scrum Team and each action item will have a defined due date for completion.

11.2.3.3 Proposed Non-Functional Items for Prioritized Product Backlog

When the initial Prioritized Product Backlog is developed, it is based on User Stories and required functionalities. Often, non-functional requirements may not be fully defined in the initial stages of the project and can surface during the Sprint Review or Retrospect Sprint Meetings. These items should be added to the Prioritized Product Backlog as they are discovered. Some examples of non-functional requirements are response times, capacity limitations, and security-related issues.

11.2.3.4 Retrospect Sprint Log(s)

The Retrospect Sprint Log is a record of the opinions, discussions, and actionable items raised in a Retrospect Sprint Meeting. The Scrum Master usually facilitates creation of this log with input from Scrum Core Team members. The collection of all Retrospective Sprint Logs becomes the project diary, which includes details on project successes, issues, problems, and resolutions. These logs are public documents available to anyone in the organization.

11.2.3.5 Scrum Team Lessons Learned

The self-organizing and empowered Scrum Team is expected to learn from any mistakes made during a Sprint. These lessons learned can help the team to improve their performance in future Sprints. Lessons learned may also be documented as Scrum Guidance Body recommendations to be shared with other Scrum Teams.

There can also be positive lessons learned as part of a Sprint. These positive lessons learned are a key part of the retrospective and should be appropriately shared within the team and with the Scrum Guidance Body, so that all Scrum teams can work towards continuous self-improvement.

Sometimes improvements go beyond the power of the Scrum Team or the Scrum Guidance Body Recommendations. In such cases, help from Senior Management and/or other stakeholders may be required, and the respective items are escalated and followed up by the Scrum Master or Product Owner.

11.2.3.6 Updated Scrum Guidance Body Recommendations

As a result of the Retrospect Sprint Meeting, suggestions may be made to revise or enhance the Scrum Guidance Body recommendations. If suggestions are accepted, these will be incorporated as updates to the applicable Scrum Guidance Body documentation.

11.3 Review and Retrospect Phase Data Flow Diagram

Figure 11-8 shows a data flow diagram summarizing the entire review and retrospect phase. Illustrates how review meetings, validation steps, and retrospectives contribute to feedback loops and enhanced team performance.

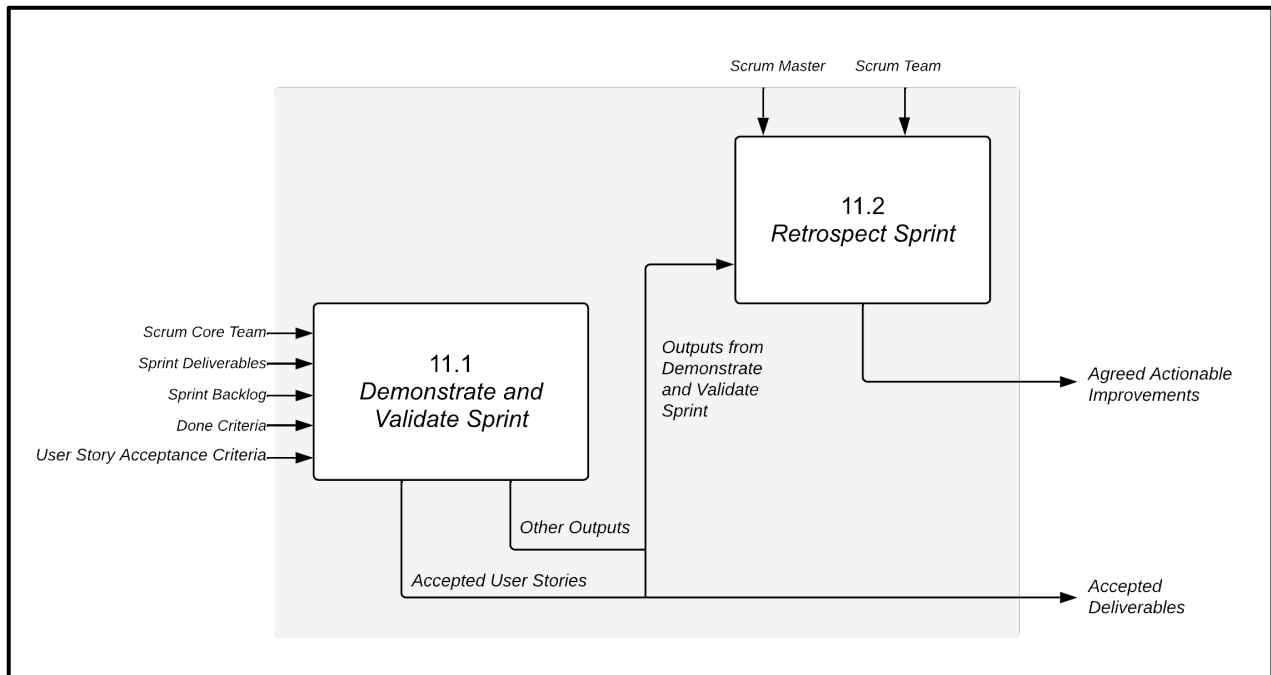


Figure 11-8: Review and Retrospect Phase—Data Flow Diagram

12.RELEASE

The Release phase emphasizes delivering the Accepted Deliverables to the customer and identifying, documenting, and internalizing the lessons learned during the project.

Release, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Portfolios, programs, and/or projects in any industry
- Products, services, or any other results to be delivered to business stakeholders
- Projects of any size or complexity

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can be applied effectively to any project in any industry—from small projects or teams with as few as six team members to large, complex projects with up to several hundred members in several teams.

To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Product Owners, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter.

This chapter is written from the perspective of one Scrum Team working on one Sprint to produce potentially shippable deliverables, which could be part of a larger project, program, or portfolio. Additional information pertaining to Scaling Scrum for Large Projects is available in chapter 13. Additional information pertaining to Scaling Scrum for the Enterprise can be found in chapter 14.

The Release phase is typically conducted multiple times during a Scrum project. Although the result of each Sprint is a potentially shippable product, there is not necessarily a release after every Sprint.

Deliverables from accepted User Stories of one or more previous Sprints are released to relevant business stakeholders for acceptance and use as defined in the Release Schedule.

In addition to the release of deliverables, the Scrum Core Team and organizational business stakeholders determine ways to improve the execution of future releases in the project.

It is also important to realize that although all phases and processes are defined uniquely in the SBOK® Guide, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to overlap some phases and/or processes, depending on the specific requirements of each project.

Figure 12-1 provides a high-level overview of the “release” phase in scrum. Includes key processes like shipping deliverables and conducting release retrospectives to ensure stakeholder value and continuous improvement at the product level.

12.1 Ship Deliverables—In this process, all deliverables from the accepted User Stories of previously completed Sprints are delivered or transitioned to the relevant business stakeholders. A formal Working Deliverables Agreement documents the successful completion of the release.

12.2 Retrospect Release—In this process, which completes a release, business stakeholders and Scrum Core Team members assemble to reflect on the release and identify, document, and internalize the lessons learned. Often these lessons lead to the documentation of agreed actionable improvements to be implemented in future project releases.

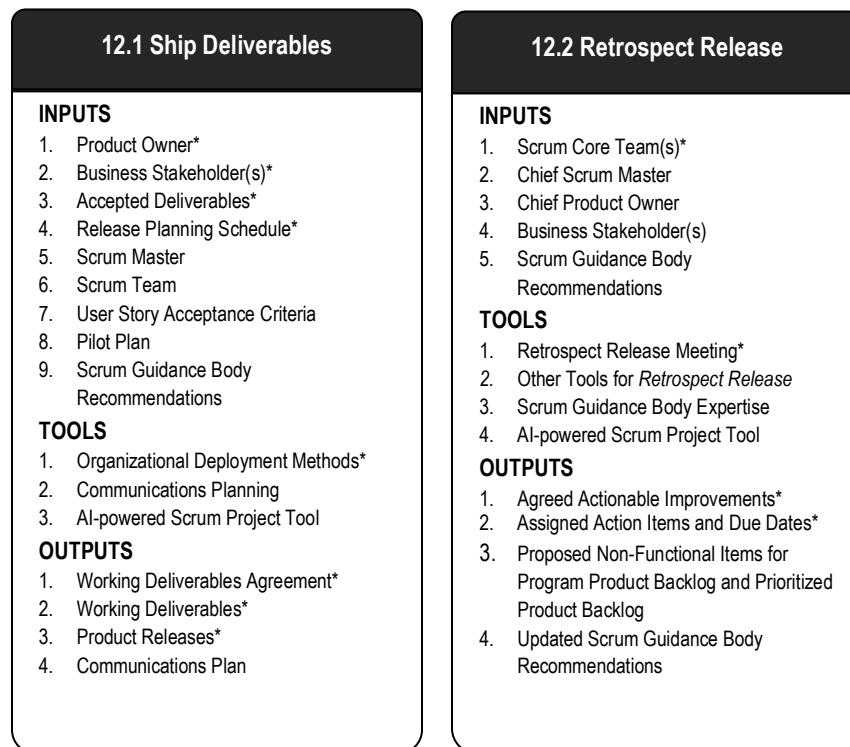


Figure 12-1: Release Overview

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

Figure 12-2 below shows the mandatory inputs, tools, and outputs for processes in the Release phase.

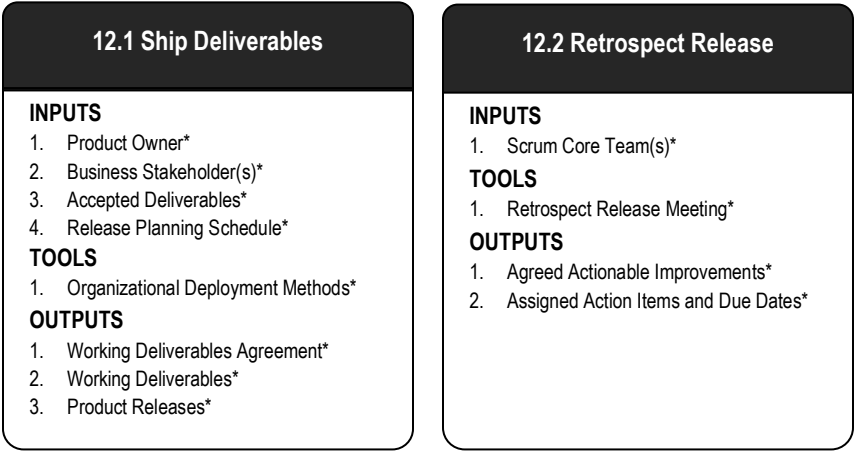


Figure 12-2: Release Overview (Essentials)

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

12.1 Ship Deliverables

In this process, all deliverables from the accepted User Stories of previously completed Sprints are delivered or transitioned to the relevant business stakeholders. A formal Working Deliverables Agreement documents the successful completion of the release.

Figure 12-3 shows all the inputs, tools, and outputs for the *Ship Deliverables* process.

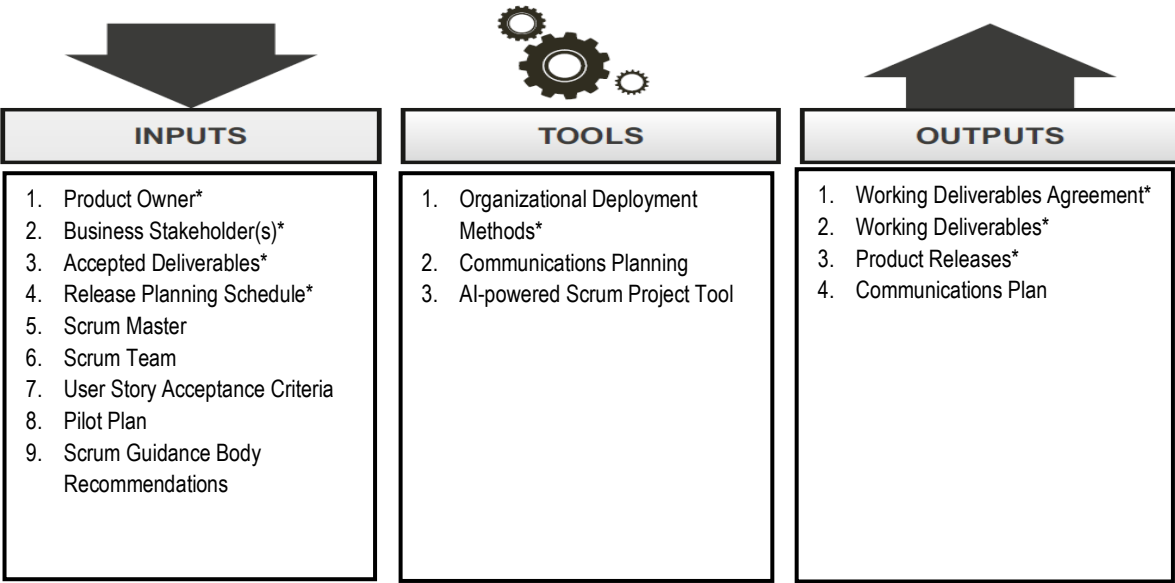


Figure 12-3: Ship Deliverables—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 12-4 is a data flow diagram for the “ship deliverables” process. Shows how deliverables and planning data flow into deployment tools to generate product releases and communication plans.

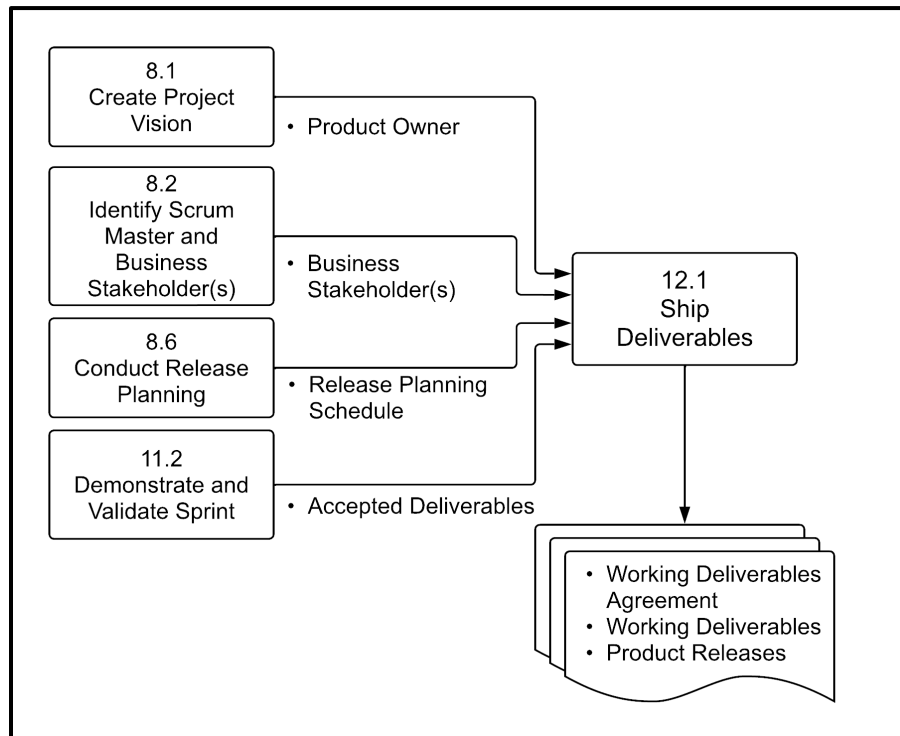


Figure 12-4: Ship Deliverables—Data Flow Diagram

12.1.1 Inputs

12.1.1.1 Product Owner*

Described in section 8.1.3.1.

12.1.1.2 Business Stakeholder(s)*

Described in section 8.2.3.2.

12.1.1.3 Accepted Deliverables*

These are the deliverables that are created by the Scrum Team and associated with the Accepted User Stories which are formally approved by the Product Owner.

Accepted User Stories is described in section to 11.1.3.1.

12.1.1.4 Release Planning Schedule*

Described in section 8.6.3.1.

12.1.1.5 Scrum Master

Described in section 8.2.3.1.

12.1.1.6 Scrum Team

Described in section 8.3.3.1.

12.1.1.7 User Story Acceptance Criteria

Described in section 9.1.3.2.

12.1.1.8 Pilot Plan

A Pilot Plan can be used to map out a pilot deployment in detail. The scope and objectives of the deployment, the target deployment user base, a deployment schedule, a transition plan, required user preparation, evaluation criteria for the deployment, and other key elements related to the deployment are specified in the Pilot Plan and shared with business stakeholders.

12.1.1.9 Scrum Guidance Body Recommendations

In the *Ship Deliverables* process, the Scrum Guidance Body can provide recommendations and guidelines regarding the deployment of products. Best practices should be considered when deploying a product to the customer in order to maximize the value delivered. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

12.1.2 Tools

12.1.2.1 Organizational Deployment Methods*

The deployment mechanisms of each organization tend to be different based on the industry, target users, and positioning of the product. Depending on the product being delivered, deployment can take place remotely or may involve the physical shipping or transition of the item(s). Because deployment often involves an important level of risk, organizations normally have well-defined and established deployment mechanisms, with detailed processes in place to ensure compliance with any applicable standards and quality assurance factors. These might include signoffs by specific management representatives, user approval mechanisms, and guidelines regarding the minimum functionality needed for a release.

12.1.2.2 Communications Planning

Communications planning is used to create the project's Communications Plan. This plan specifies the records that must be created and maintained throughout the project. A variety of methods are used to convey important project information to business stakeholders. As the User Story deliverables are tested, the status of the testing activities is communicated as per the Communications Plan as determined by the Product Owner and sponsor.

12.1.2.3 AI-powered Scrum Project Tool

Described in section 2.4.4

Figure 12-5 is a Vabro interface used for release management in scrum. Displays how ai-powered tools assist with packaging, approval tracking, and version release documentation for stakeholder delivery.

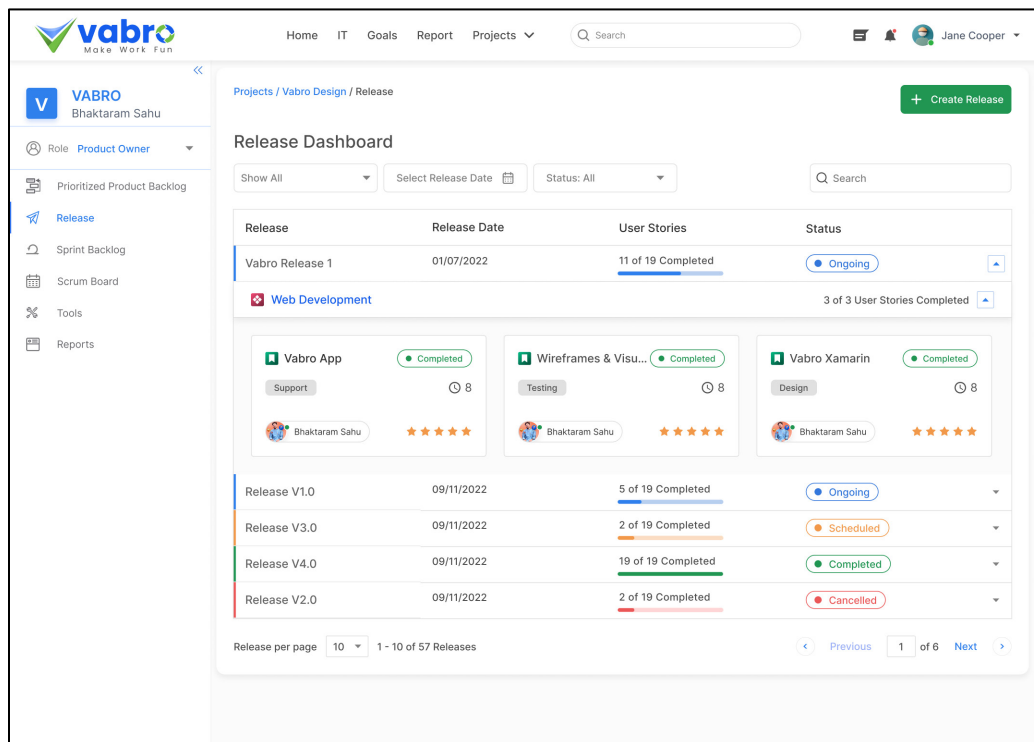


Figure 12-5: Release Management in Scrum (Source: Vabro)

Figure 12-6 is a Jira interface for managing releases. Includes visualization of release dates, deployment checklists, and linked backlog items to ensure completeness and traceability.

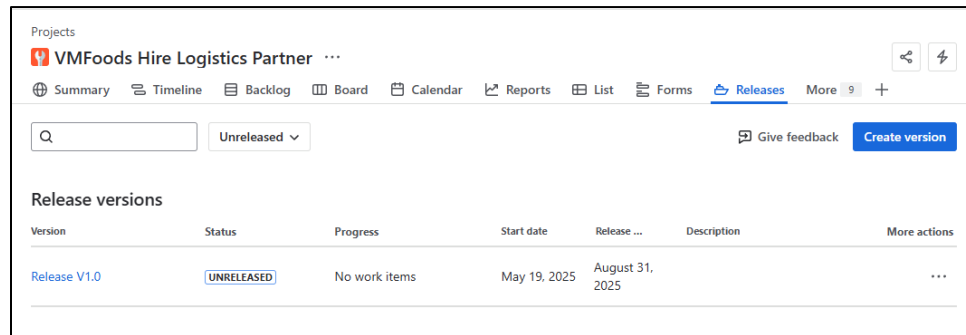


Figure 12-6: Release Management in Scrum (Source: Jira)

12.1.3 Outputs

12.1.3.1 Working Deliverables Agreement*

Deliverables that are accepted receive formal business sign-off and approval from the customer and/or sponsor. Obtaining formal acceptance from the client for each working deliverable is essential for revenue recognition, the acceptance of the overall project results, and the fulfillment of the project objectives.

12.1.3.2 Working Deliverables*

Working deliverables are the final completed, tested, and approved and shippable deliverables for which the project was sanctioned. As new product increments are created, they are continually integrated into prior increments, so there will always be a potentially shippable product available at all times throughout the project.

12.1.3.3 Product Releases*

Product releases should include the following:

- **Release Content**—This consists of essential information about the deliverables that can assist the customer support team in their use.
- **Release Notes**—Release notes include external or market-facing shipping criteria for the product to be delivered.

12.1.3.4 Communications Plan

The Communications Plan defines the methods to be used to convey important project information to business stakeholders, as well as who is responsible for various communication activities. A common communication mechanism is a visual display depicting essential information in an easy-to-interpret format, posted in an accessible location, and kept up to date with the most current information.

12.2 Retrospect Release

In this process which completes a release, business stakeholders and Scrum Core Team members assemble to reflect on the release and identify, document, and internalize the lessons learned. Often these lessons lead to the documentation of agreed actionable improvements to be implemented in future project releases.

Figure 12-7 shows all the inputs, tools, and outputs for the *Retrospect Release* process.

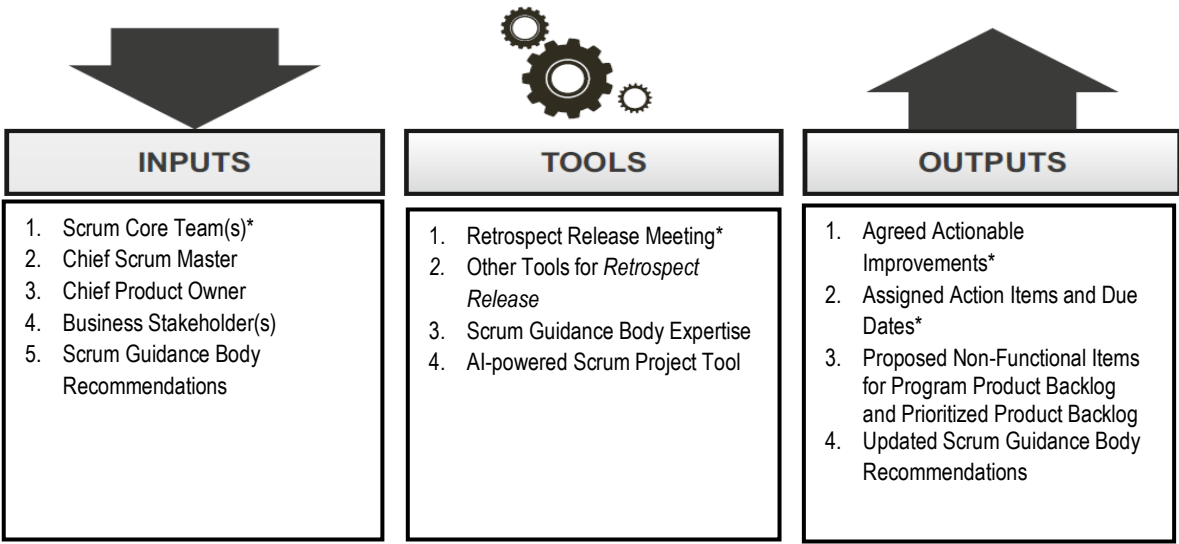


Figure 12-7: Retrospect Release—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

Figure 12-8 is a data flow diagram for the “Retrospect Release” process. Illustrates how feedback from program or portfolio levels feeds into improvements, action plans, and updated scrum practices.

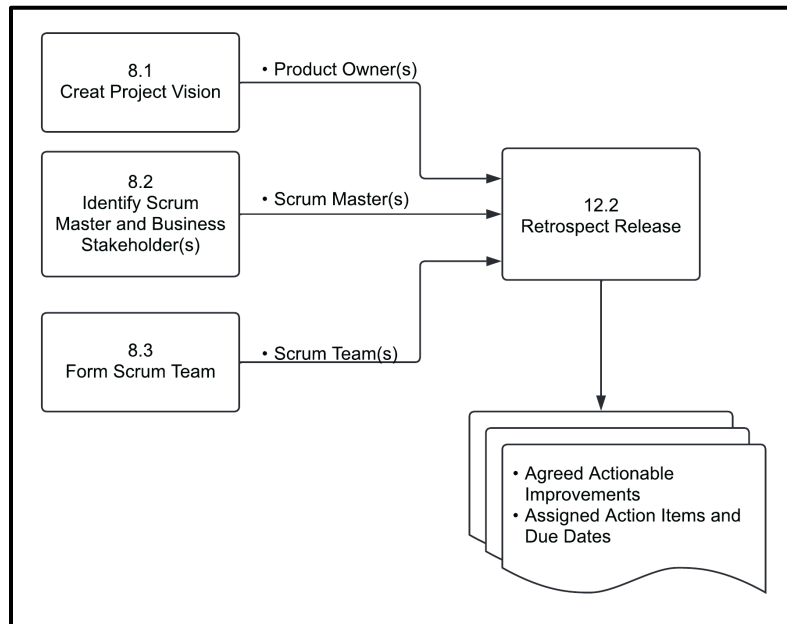


Figure 12-8: Retrospect Release—Data Flow Diagram

12.2.1 Inputs

12.2.1.1 Scrum Core Team(s)*

Described in section 8.4.1.1.

12.2.1.2 Chief Scrum Master

Described in section 3.7.2.2.

12.2.1.3 Chief Product Owner

Described in section 3.7.2.1.

12.2.1.4 Business Stakeholder(s)

Described in section 8.2.3.2.

12.2.1.5 Scrum Guidance Body Recommendations

In the *Retrospect Release* process, recommendations from the Scrum Guidance Body can include a repository of internal templates that support all projects and guidance for conducting the Retrospect Release Meeting. The guidance provided can relate to administrative procedures, audits, evaluations, and project transition criteria. Often, recommendations also include how the organization will maintain the knowledge base of lessons learned and information from all projects. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

12.2.2 Tools

12.2.2.1 Retrospect Release Meeting*

The Retrospect Release Meeting is a meeting used to determine ways in which team collaboration and effectiveness can be improved in future releases. Positives, negatives, and potential opportunities for improvements are also discussed. This meeting is not Time-boxed and may be conducted in person or in a virtual format. Attendees include the Scrum Core Team, Chief Scrum Master, Chief Product Owner, and business stakeholder(s). During the meeting, lessons learned are documented and participants look for opportunities to improve processes and address inefficiencies. If suggestions for improvements are outside the approval for the Scrum Core Team and/or the Scrum Guidance Body, these suggestions should be escalated to the appropriate organizational executives or others outside the project.

12.2.2.2 Other Tools for Retrospect Release

Some of the tools used in the *Retrospect Sprint* process can also be used in this process. Examples include:

- Explorer, Shopper, Vacationer, Prisoner (ESVP),
- Speed Boat, and
- Metrics and Measuring Techniques

For more information on the above tools, see sections 11.2.2.2, 11.2.2.3, and 11.2.2.4.

12.2.2.3 Scrum Guidance Body Expertise

In the *Retrospect Release* process, the primary responsibility of the Scrum Guidance Body is to ensure that the lessons learned from all projects are not lost but are embedded throughout the organization and its continuous improvement efforts. In addition to Scrum-related expertise, expertise may also be provided in various other areas (such as quality management and human resource management), that may be helpful in the *Retrospect Release* process. Also, there may be suggestions from the Scrum Guidance Body concerning how the Retrospect Release Meeting should be conducted. For more information on Scrum Guidance Body Expertise, see section 8.4.2.7.

12.2.2.4 AI-powered Scrum Project Tool

Described in section 2.4.4

12.2.3 Outputs

12.2.3.1 Agreed Actionable Improvements*

Described in section 11.2.3.1.

12.2.3.2 Assigned Action Items and Due Dates*

Described in section 11.2.3.2.

12.2.3.3 Proposed Non-Functional Items for Program Product Backlog and Prioritized Product Backlog

When the initial Program Product Backlog or Prioritized Product Backlog are developed, they are based on User Stories and required functionalities. Often, non-functional requirements may not be fully defined in the early stages of the project and can surface during the Sprint Review, Retrospect Sprint, or Retrospect Release Meetings. These items should be added to the Program Product Backlog (for the program) and the relevant Prioritized Product Backlog (for the project) as they are discovered. Some examples of non-functional requirements are response times, capacity limitations, and security-related issues.

12.2.3.4 Updated Scrum Guidance Body Recommendations

Described in section 11.2.3.6.

12.3 Release Phase Data Flow Diagram

Figure 12-9 depicts the final two processes in the release phase: "Ship Deliverables" and "Retrospect Release." It shows the sequential flow where deliverables are shipped to the customer, followed by a release retrospective to evaluate performance, gather lessons learned, and improve future delivery cycles.

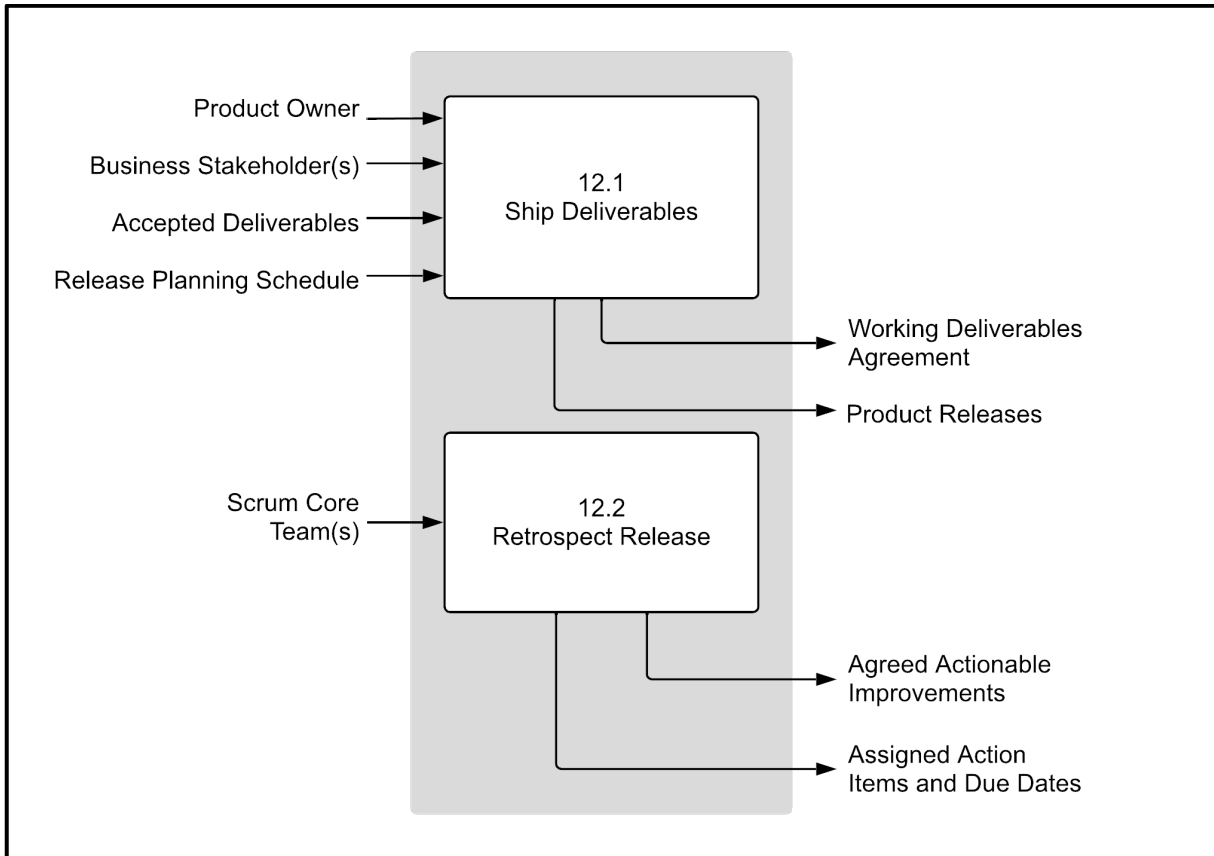


Figure 12-9: Release Phase—Data Flow Diagram

This Page is Intentionally Left Blank.

13. SCALING SCRUM FOR LARGE PROJECTS

This chapter emphasizes additional aspects of Scrum that are applicable to large projects. Scaling Scrum for large projects, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Large projects in any industry
- Products, services, or any other results to be delivered to business stakeholders

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable. Scrum can not only be applied effectively to small projects in any industry but also to large, complex projects which may have hundreds of people assigned to many teams. In addition to the impacts that a large project can have on the fundamental Scrum processes discussed in chapters 8 through 12, this chapter introduces additional inputs, tools, and outputs that could apply to large projects.

To facilitate the best application of the Scrum framework, this chapter identifies additional inputs, tools, and outputs as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional. It is recommended that the Scrum Team and those individuals being introduced to the Scrum framework and processes focus primarily on the mandatory inputs, tools, and outputs; while Chief Product Owners, Product Owners, Chief Scrum Masters, Scrum Masters, and other more experienced Scrum practitioners strive to attain more thorough knowledge of the information in this entire chapter. It is also important to realize that although all processes are defined uniquely in the *SBOK® Guide*, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to perform some processes in parallel, depending on the specific requirements of each project.

This chapter is written from the perspective of a large project effort that coordinates activities of multiple Scrum Teams to jointly produce potentially shippable product increments/deliverables. Additional information pertaining to the application of Scrum to any project (large or small) can be found in chapters 2 through 7, which cover Scrum principles and Scrum aspects.

Large Scrum Project vs. Typical Small Scrum Project

The fundamental Scrum processes defined in chapters 8 through 12 are valid for Scrum projects with one Product Owner, one Scrum Master, and one to three Scrum Teams. These are usually considered small Scrum projects.

When dealing with large projects requiring the efforts of four or more Scrum Teams with multiple Product Owners and multiple Scrum Masters, the fundamental processes defined in chapters 8 through 12 remain valid, but some additional considerations and updates to inputs, tools, and outputs may be required. This may include additional coordination and synchronization efforts. The impact to the fundamental Scrum processes when scaling Scrum to large projects are described in this chapter.

The definition of what constitutes a large project usually depends on the organization and/or the complexity of the projects being undertaken. A key criterion for whether a project is considered small versus large is whether the project requires multiple Scrum Masters and/or multiple Product Owners. If the project still requires only one Scrum Master and one Product Owner, then these individuals can normally handle any additional communication and synchronization efforts required by the project.

Some reasons additional inputs, tools, and outputs would be needed for large projects are as follows:

Scrum Teams

- Increased interaction and dependencies among Scrum Teams, as complexity increases for a large project
- Need to manage conflicts, resolve issues, manage dependencies, and set priorities among all the Scrum Teams
- Requirement for specialization as some Scrum Teams may require specialized resources for specific tasks (and these particular skill sets are not needed on all Scrum Teams)
- Necessity to define certain guidelines and standards that should be adhered to by all Scrum Teams (e.g., security standards within a company or legal and governmental guidelines for specific industries); these may be defined by the Scrum Guidance Body
- Requirement to set up an environment or working area for the large project, which would then be used by all Scrum Teams
- Need for coordinating the outputs from several Scrum Teams to facilitate the release of a large project

Scrum Masters

- Need for collaboration between Scrum Masters when addressing impediments and for synchronizing the work of the multiple Scrum Teams

Product Owners

- Need for collaboration between Product Owners when working with business stakeholders, refining the Prioritized Product Backlog, and working with Scrum Teams.

It is also important to note that as Scrum is scaled for large projects, additional supporting services may be needed such as architects, product managers, compliance, information security, governance bodies, and so on.

Figure 13-1 shows the Vabro dashboard used for setting up a large project, displaying information on the project, personal tasks, and navigation options, allowing users to manage work, settings, and access the admin console efficiently.

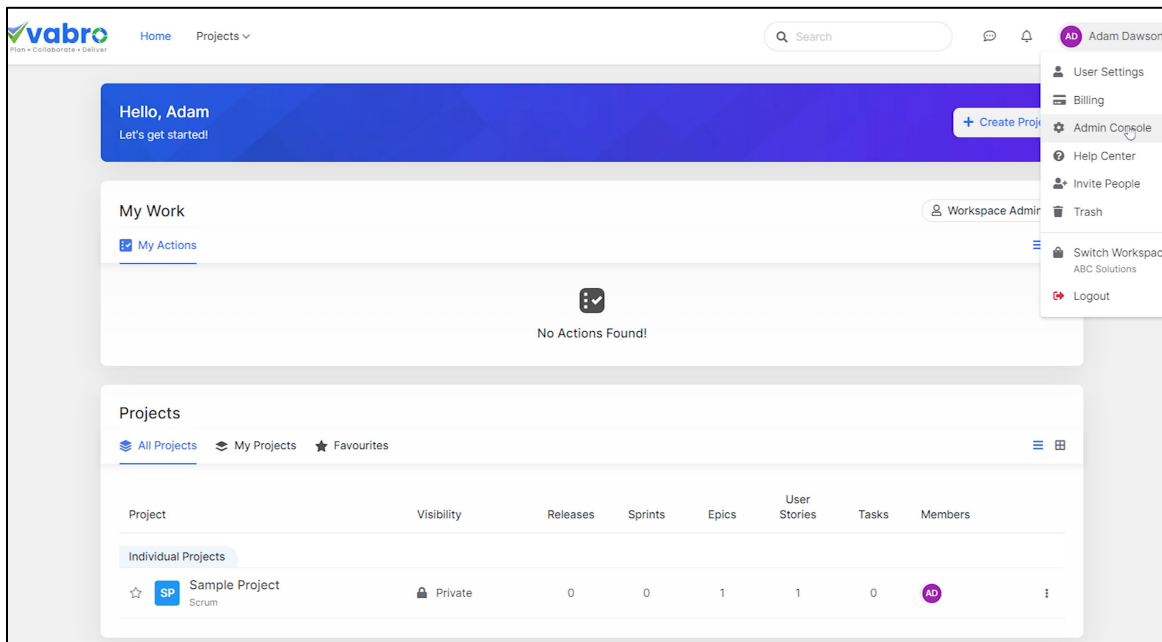


Figure 13-1: Setting up a Large Project (Source: Vabro)

13.1 Impact of Large Projects on Fundamental Scrum Processes

Although the fundamental Scrum processes described in chapters 8 through 12 remain valid for large projects, there are additional considerations that should be noted. Tables 13-1 through 13.5 outline a summary of the impacts of a large project to the fundamental Scrum processes for each project phase.

13.1.1 Initiate

The Initiate phase in a large project has the same objectives and follows the same flow as the Initiate phase in a typical/small Scrum project.

Compared to a typical Scrum project, additional roles need to be identified and additional activities need to be performed to achieve an agreement on how the multiple Product Owners, Scrum Masters, and Scrum Teams will collaborate among themselves and with the business stakeholders.

Figure 13-2 shows the project creation screen in Vabro, where users define the template, visibility, name, description, and estimation criteria for structured planning.

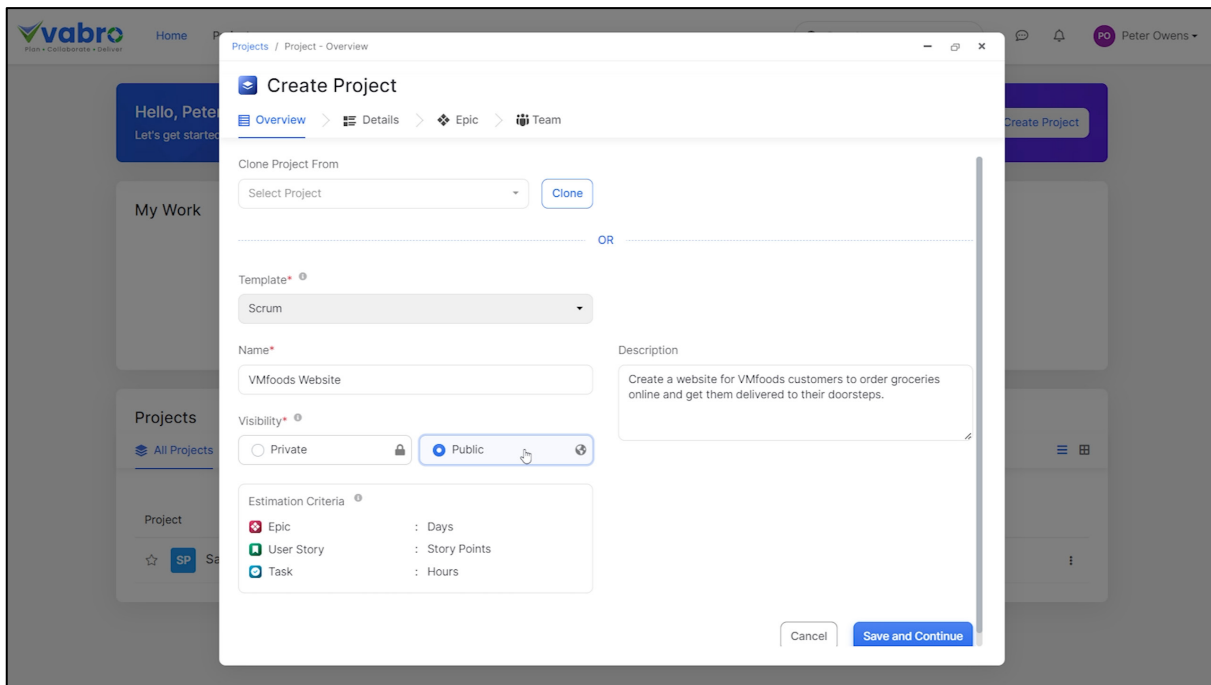


Figure 13-2: Initiating a Large Project (Source: Vabro)

Process	Summary of Impacts of a Large Project
8.1 Create Project Vision	A Chief Product Owner and additional Product Owners are identified in this process. In a large project, the Chief Product Owner has the overall business responsibility for the project and works with the sponsor and other business stakeholders to create the Project Vision. Additionally, the Product Owners Collaboration Plan is created. Additional Output: Identified Chief Product Owner* (see section 3.7.2.1) In a large project, the Chief Product Owner has the overall business responsibility for the project and works with the sponsor and other business stakeholders to create the project vision. As part of this process, the Chief Product Owner is identified. Updated Output: Identified Product Owner* (see section 8.1.3.1) Because in a large project, there are multiple Product Owners, they also need to be identified in this process. Additional Output: Product Owners Collaboration Plan* (see section 13.2.2) For large projects, it will be essential for the entire team of Product Owners to embrace Scrum and to collaborate to successfully deliver Scrum projects.

Process	Summary of Impacts of a Large Project
8.2 Identify Scrum Master and Business Stakeholder(s)	A Chief Scrum Master and additional Scrum Masters are identified in this process. The Chief Scrum Master focuses on multi-team interaction and synchronization. Several additional or updated outputs for this process are as follows. Additional Output: Identified Chief Scrum Master* (see section 3.7.2.2) Similar to the Chief Product Owner, the Chief Scrum Master should also be identified for a large project. He/she focuses on multi-team interaction and synchronization. Updated Output: Identified Scrum Masters* (see section 8.2.3.1) Because in a large project, there are multiple Scrum Masters that are identified in this process. Additional Output: Large Project Scrum Organization* (see section 13.2.1) It will be essential for the entire large project team to embrace a common understanding of Scrum and working agreements to successfully deliver the Scrum project. Additional Output: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) The Scrum Masters/Scrum Teams Collaboration Plan defines how the Scrum Masters and teams participate in the refining of the Prioritized Product Backlog. This plan would also define which team representatives would be involved in the refining process, and how they are selected. Additional Output: Shared Resources* (see section 13.2.4) Knowledge of any shared resources available to the Scrum Teams would be a necessary input in forming the individual Scrum Teams. Updated Output: Identified Supporting Services (see section 3.2.2) In addition to identifying supporting services, for a large project, some additional supporting services may be needed to coordinate activities between all Product Owners, Scrum Masters, and Scrum Teams.

Process	Summary of Impacts of a Large Project
8.3 Form Scrum Team	In a large project, multiple Scrum Teams are formed, involving multiple Product Owners and Scrum Masters, as well as a Chief Product Owner and a Chief Scrum Master. The Chief Product Owner and Chief Scrum Master are involved in determining the formation of the Scrum Teams and they also provide input during the selection of team members. The Chief Product Owner and Chief Scrum Master serve the interests of the larger project, while the Product Owners and Scrum Masters are more focused on their respective Scrum Teams. Additional Input: Chief Product Owner* (see section 3.7.2.1) For a large project, the Chief Product Owner would be involved in determining the formation of the Scrum Teams and have input regarding the members of the teams. The Chief Product Owner would serve the interests of the large project as a whole, while Product Owners would be focused on the individual team level. Additional Input: Chief Scrum Master* (see section 3.7.2.2) For a large project, the Chief Scrum Master would be involved in determining the formation of the Scrum Teams and have input to the members for the teams. The Chief Scrum Master would serve the interests of the large project as a whole, while Scrum Masters would be focused on the individual team level. Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) Additional Input: Team Specialization* (see section 13.2.5) Some Scrum Teams and Scrum Team members may require specialized skills to work on specific issues related to large projects. Additional Tool: Large Project Communications Plan* (see section 13.3.1) This plan highlights how to manage effective communication between all people involved with a large project. Additional Tool: Large Project Resource Planning* (see section 13.3.2) This helps manage the complexity of allocating several types of resources to the numerous Scrum Teams working in parallel. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8) Since large projects would have many people, there would be significant complexity and interaction between Scrum Teams, it is recommended to use an AI-powered Scrum Project Tool to automate processes, manage complexity, generate reports, manage communication between business stakeholders, etc.

Process	Summary of Impacts of a Large Project
8.3 Form Scrum Team (cont'd)	Additional Tool: Environment Identification* (see section 13.3.3) In large projects, it is important to identify the number and types of environments needed because numerous Scrum Teams will be working simultaneously to carry out the work of their respective Sprints. Additional Output: Environment and Environment Schedule* (see section 13.2.6) After environments are identified, an Environment Schedule is created which is used for the coordination of Sprint activities across teams. Additional Output: Updated Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) As Scrum Teams are formed, inputs from the teams and additional considerations will result in updates to Scrum Masters/Scrum Teams Collaboration Plan.
8.4 Develop Epic(s)	Any Product Owner interaction with the customer and other business stakeholders is handled by the Chief Product Owner and the multiple Product Owners, rather than a single Product Owner. How that interaction is divided is defined in the Product Owner Collaboration Plan. Any interaction with and participation of Scrum Masters and / or Scrum Team members occurs as defined in the Scrum Masters/Scrum Teams Collaboration Plan. Otherwise, the creation of Epics is done the same way as in a typical small Scrum project. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) It defines how the multiple Product Owners work together and with the Chief Product Owner. It addresses how they work with the business stakeholders for collecting requirements, make updates to the Prioritized Product Backlog, and work with the multiple Scrum Teams. There will be only one Product Owner directly interfacing each Scrum Team. However, decisions must be made regarding how Scrum Teams will be allocated among the Product Owners, and how many Scrum Teams each Product Owner will work with. Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)

Process	Summary of Impacts of a Large Project
8.5 Create Prioritized Product Backlog	Typically, the various Product Owners contribute to the creation of the Prioritized Product Backlog, as agreed to in the Product Owners Collaboration Plan. Final prioritization decisions and resolution of any conflicts between Product Owners are handled by the Chief Product Owner. Otherwise, the process is handled the same way as in a typical small Scrum project. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) Since the Product Owners Collaboration Plan defines how the Product Owners make updates to the Prioritized Product Backlog, it is an important input to this process. Additional Tool: Prioritized Product Backlog Assignments* (see section 13.3.4) Since the Chief Product Owner and several Product Owners are involved with a large project, Prioritized Product Backlog Assignments ensure effective allocation of Epics and User Stories to all the Product Owners. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)
8.6 Conduct Release Planning	For a large project, the Release Planning Schedule is created by the Chief Product Owner. Since releases can be more complicated with large projects, a Release Readiness Plan is created in the process (if needed) which is later used to confirm that the requirements for each release have been met. Otherwise, the process is handled the same way as in a typical Scrum project Additional Input: Chief Scrum Master* (see section 3.7.2.2) Additional Output: Release Readiness Plan* (see section 13.2.7) The Release Readiness Plan includes specific activities that need to be done shortly before release planning.

Table 13-1: Impact of Large Projects to Fundamental Scrum Processes—Initiate Phase

13.1.2 Plan and Estimate

As in a typical/small Scrum project, the Plan and Estimate phase in a large project has the objective to agree on and plan the work that will be completed in the upcoming Sprint. Each Scrum Team works with its respective Scrum Master and Product Owner to commit to the specific work for the Sprint and to plan how the team will perform the work (as in a typical Scrum project).

Compared to a small Scrum project, additional steps need to be taken to divide the work between the multiple Scrum Teams. Which teams and team members participate in the creation and estimation of which User Story, and how the commitment of User Stories and the creation of the respective deliverables are divided between the different teams is based on team specialization and the Collaboration Plan.

Figure 13-3 displays the User Story creation interface in Vabro, allowing users to define the name, description, work type, and team assignment for backlog items.

The screenshot shows the 'Create User Story' modal in the Vabro application. The modal is titled 'Create User Story' and has tabs for 'Overview', 'Details', and 'Task'. The 'Overview' tab is active. It features a 'Clone User Story From' section with a dropdown menu and a 'Clone' button. Below this is an 'OR' separator. The main form has four fields: 'Name*' (text input), 'Description' (text area), 'Work Type*' (radio buttons for 'Finite' and 'Recurring'), and 'Worked By*' (radio buttons for 'Single Team' and 'Multiple Teams'). There are red error messages below the 'Work Type' and 'Worked By' sections stating 'This cannot be changed once saved.' At the bottom right are 'Cancel' and 'Save and Continue' buttons. The background shows the Vabro dashboard with a sidebar menu and a top navigation bar.

Figure 13-3: Creating a User Story in Plan and Estimate Phase (Source: Vabro)

Process	Summary of Impacts of a Large Project
9.1 Create User Stories	In a large project, multiple Product Owners and multiple Scrum Teams are involved in the Creation of User Stories. Not every Product Owner and not every Scrum Team/Scrum Team member can be involved in the creation of every User Story. Hence, the creation of User Stories is divided between the multiple Product Owners based on the Product Owners Collaboration Plan, as well as between the multiple Scrum Teams based on team specialization and the Collaboration Plan. Otherwise, the process is done the same way as in a typical small Scrum project. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) Additional Tool: Environment Identification* (see section 13.3.3) In a large project, it is important to identify the number and types of environments needed for Scrum Teams to work effectively. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)
9.2 Estimate User Stories	In a large project, not every Scrum Team/Scrum Team member can be involved in the estimation of every User Story. Hence, only specific teams/team members usually participate in the estimation of User Stories. Which team or team members estimate which User Stories is decided based on team specialization and the Collaboration Plan. Which Product Owner works with the respective Scrum Teams/Team members usually depends on who created the respective User Stories and is best suited to answer any questions related to the User Stories being estimated. This is also based on the Product Owners Collaboration Plan. Otherwise, the estimation of User Stories is done the same way as in a typical Scrum project. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)

Process	Summary of Impacts of a Large Project
9.3 Commit User Stories	For each Sprint in a large project, each team is asked to commit to a specific subset of User Stories for the Sprint and then to create the respective deliverables. The decision, which team is asked to commit and implement which User Stories, depends on the specific skills of each team, and is based on team specialization. Otherwise, the commitment of User Stories is handled the same way as in a typical small Scrum project, based on priorities, estimates, and team-specific velocity. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)
9.4 Identify Tasks	Each team does the identification of tasks to decompose the User Stories that the team has committed to. Each Scrum Team identifies tasks in a similar way as they would in a typical small Scrum project. Updated Tool: Dependency Determination* (see section 8.5.2.6) Also described in section 9.4.2.3. In large projects, properly identifying dependencies helps the Scrum Teams determine which of their decisions and activities may impact other teams. It may also influence the relative order in which a single Scrum Team executes its respective tasks to create the Sprint Deliverables. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)
9.5 Estimate Tasks	Tasks identified in the previous process would optionally be estimated, exactly as it is done in a typical small Scrum project. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)
9.6 Update Sprint Backlog	Each team updates its specific Sprint Backlog according to its committed User Stories and associated tasks. Each team updates the Sprint Backlog exactly as it is done in a typical small Scrum project. Additional Input: Environment and Environment Schedule (see section 13.2.6) Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)

Table 13-2: Impact of Large Projects to Fundamental Scrum Processes—Plan and Estimate Phase

13.1.3 Implement

In the Implement phase of a large project, each Scrum Team, facilitated by its Scrum Master, creates the deliverables that are associated with the User Stories that were committed to by working on and completing the tasks that the team identified in the Plan and Estimate phase.

Compared to a small Scrum project, additional steps need to be taken to ensure there is effective communication between the multiple teams (as defined in the Communications Plan) and that the work is synchronized (as detailed) in the Collaboration Plan).

As in a typical/small Scrum project, while the Scrum Teams are creating the deliverables of the Sprint, the Chief Product Owner and the other Product Owners refine the Prioritized Product backlog to keep it up to date with any changes in requirements and/or changes to priorities. They also ensure that the set of User Stories they would like the Scrum Teams to commit to deliver in the next Sprint will be ready for estimation and commitment.

On a large project, the Product Owners divide their work, based on the Product Owners Collaboration Plan. Interactions with the Scrum Teams and/or Scrum Team members is based on the Scrum Team's specialization and the Collaboration Plan.

Figure 13-4 shows the “Prioritize User Stories” screen in Vabro, enabling users to rank backlog items by priority, DoR, status, and story point estimates.

User Story	Epic	Release	User Story Priority	Estimate (Story Points)	Status	Scrum Team	DoR
L3 Tech Browser	Customer Support Tickets	-	★★★★★	-	Not Scheduled	-	No
L2 Tech Mobile App	Customer Support Tickets	-	★★★★★	-	Not Scheduled	-	No
L1 Tech	Customer Support Tickets	-	★★★★★	-	Not Scheduled	-	No
L2 Tech Browser	Customer Support Tickets	-	★★★★★	-	Not Scheduled	-	No

Figure 13-4: Prioritization of User Stories in Implement Phase (Source: Vabro)

Process	Summary of Impacts of a Large Project
10.1 Create Deliverables	Each team creates the deliverables associated with the User Stories it has committed to. Different from a typical small Scrum Project, a Scrum Team in a large project faces limitations in its freedom to organize its work. These limitations are due to the need to share resources with other teams (as specified and agreed to in the Shared Resources) as well as the Environment and Environment Schedule, and also due to potential impacts from decisions that other teams may make. Furthermore, additional activities are required to ensure effective communication and synchronization between the Scrum Teams. Additional Input: Release Readiness Plan* (see section 13.2.7) Additional Tool: Scrum of Scrum Meetings* (see section 13.3.5) These are focused meetings where Scrum Team representatives meet to share the status of their respective teams. Additional Tool: Release Preparation Methods* (see section 13.3.6) Release preparation methods are used to execute the tasks identified in the Release Readiness Plan in order to ready the deliverables to be shipped/released.
10.2 Conduct Daily Stand Up	Each team conducts its Daily Standup Meeting exactly as it is done in a typical Scrum project. However, because each Scrum Master may be working with multiple Scrum Teams, some coordination effort is required to avoid scheduling conflicts between the Daily Standup Meetings of their respective Scrum Teams. Additional Optional Tool: AI-powered Scrum Project Tool (see section 13.3.8)

Process	Summary of Impacts of a Large Project
10.3 Refine Prioritized Product Backlog	The differences compared to a typical small Scrum project are the same differences as in the processes <i>Develop Epics</i> and <i>Create Prioritized Product Backlog</i>. Specifically, any Product Owner interaction with the customer and other business stakeholders is handled by the Chief Product Owner and/or the multiple Product Owners, rather than a single Product Owner. How that interaction is divided is defined in the Product Owners Collaboration Plan. Additionally, the Chief Product Owner makes final prioritization decisions. Any interaction with and participation of Scrum Masters and/or Scrum Team members occurs as defined in the Scrum Masters/Scrum Teams Collaboration Plan. Otherwise, the Refining of the Prioritized Product Backlog is handled as in a typical Scrum project. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) The Product Owners Collaboration Plan defines how the Product Owners make updates to the Prioritized Product Backlog. Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) Additional Input: Team Specialization* (see section 13.2.5) Additional Tool: AI-powered Scrum Project Tool (see section 13.3.8) Additional Optional Tool: Release Readiness Sprint* (see section 13.3.7) Additional Outputs: Updated Release Readiness Plan* (see section 13.2.7) Changes to the Prioritized Product Backlog made during the refining of the Prioritized Product Backlog may impact the Release Readiness Plan.

Table 13-3: Impact of Large Projects to Fundamental Scrum Processes—Implement Phase

13.1.4 Review and Retrospect

In the Review and Retrospect phase of a large project, each Scrum Team demonstrates the deliverables the team has created in a Sprint to its respective Product Owner for approval and feedback and then meets to determine ways to continually improve their work. This is done the same way as in a typical Scrum project.

In addition, the Product Owners meet to determine ways to improve their work, and as the final step of a Sprint, the Chief Product Owner and the Chief Scrum Master meet with the appropriate Product Owners, Scrum Masters, and Scrum Team members to share the results of their respective Retrospect Sprint Meetings. Because each Scrum Master and each Product Owner may be working with multiple Scrum Teams, some coordination effort is required to avoid scheduling conflicts between the Sprint Review Meetings and/or the Retrospect Sprint Meetings of different Scrum Teams.

Process	Summary of Impacts of a Large Project
11.1 Demonstrate and Validate Sprint	This process is carried out individually by each Scrum Team. For each team, the respective Product Owner approves the completed User Stories. However, this can be complex due to inter-dependencies. There may be times when the Chief Product Owner attends Sprint Review Meetings for some Scrum teams which have inter-dependent deliverables. Additional Input: Chief Product Owner* (see section 3.7.2.1)

Process	Summary of Impacts of a Large Project
11.2 Retrospect Sprint	Each Scrum Team in the project meets with its Scrum Master for a Retrospect Sprint Meeting, which is done in the same way as in a typical small Scrum project. Because a single Scrum Master and a single Product Owner may work with multiple Scrum Teams, some coordination effort is required to avoid scheduling conflicts between the Retrospect Sprint Meetings of different Scrum Teams. Also, the Chief Product Owner and the Product Owners meet for a Retrospect Sprint Meeting to discuss their collaboration and other aspects of the Sprint. Additionally, Scrum Masters and/or other representatives from each Scrum Team meet for a special Scrum of Scrums Meeting (SoS) to share best practices and other results of the Retrospect Sprint Meetings of the different teams, for example, issues with inter-team collaboration. Because in a large project, very often best practices, and problems stem from the collaboration between the multiple Scrum Teams and the team of Product Owners, it is common practice for the Chief Product Owner and other Product Owners to participate in this meeting. Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) Refining the Prioritized Product Backlog may be particularly difficult where large projects are concerned. If not done effectively, refining can cause problems and waste effort across teams. Therefore, it is recommended that refining be discussed as part of the retrospective, specifically focusing on how the multiple Product Owners interface with each other and with the Scrum Teams to effectively conduct backlog refinement. Also, it is preferable to have Epics and User Stories with a lot of dependencies to be grouped together. Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) For large projects the refining of the Prioritized Product Backlog may be particularly difficult. If not done effectively, refining can create problems and waste effort across the teams. Therefore, it is recommended that refining be discussed as part of the retrospective, with a specific focus on the interworking between the various Scrum Masters, Scrum Teams and how they interact with Product Owners for the refining activities.

Table 13-4: Impact of Large Projects to Fundamental Scrum Processes—Review and Retrospect Phase

13.1.5 Release

In the Release phase of a large project, the Product Owners of individual Scrum Team coordinate with the Chief Product Owner to ensure that the deliverables of all Scrum Teams are synchronized, integrated and released as required by the customer and other business stakeholders. Additionally, the Product Owners Collaboration Plan and Scrum Masters/Scrum Teams Collaboration Plan is refined to ensure better coordination across all teams in a large project.

Process	Summary of Impacts of a Large Project
12.1 Ship Deliverables	In a large project, all Accepted Deliverables from previously completed Sprints are generally delivered or transitioned to business stakeholders in the same way as in a typical small Scrum project. However, based on business considerations and due to the complexity of large projects, additional preparation steps may be necessary to prepare for the Release. Additional Input: Chief Product Owner (see section 3.7.2.1) Additional Input: Chief Scrum Master (see section 3.7.2.2) Additional Input: Release Readiness Plan (see section 13.2.7) Additional Optional Tool: Release Readiness Sprint (see section 13.3.7) Sometimes a Release Readiness Sprint may need to be planned for the whole project release. Requirements of such a Sprint are then added to the Prioritized Product Backlog.

Process	Summary of Impacts of a Large Project
13.2 Retrospect Release	Additional Input: Chief Product Owner* (see section 3.7.2.1) Additional Input: Chief Scrum Master* (see section 3.7.2.2) Additional Input: Product Owners Collaboration Plan* (see section 13.2.2) For large projects the refining of the Prioritized Product Backlog may be particularly difficult. If not done effectively, refining can cause problems and waste effort across teams. Therefore, it is recommended that refining be discussed as part of the retrospective, specifically focusing on how the multiple Product Owners interface with each other and with the Scrum Teams to effectively conduct backlog refinement. Also, it is preferable to have Epics and User Stories with a lot of dependencies to be grouped together. Additional Input: Scrum Masters/Scrum Teams Collaboration Plan* (see section 13.2.3) For large projects coordinating and managing activities and interdependencies across multiple Scrum Masters and Scrum Teams may be very difficult – so it is recommended that a Scrum Masters/Scrum Teams Collaboration Plan is available and continually refined to ensure better collaboration across all Scrum Masters and Scrum Teams.

Table 13-5: Impact of Large Projects to Fundamental Scrum Processes—Release Phase

13.2 Additional Inputs and Outputs for Large Projects

13.2.1 Large Project Scrum Organization*

The appropriate organizational structure needed to implement and support large projects is defined in consultation with the Chief Product Owner, Chief Scrum Master, Senior Management, Scrum Guidance Body, and other relevant experts. Organizations planning to use Scrum to implement large projects should fully embrace the Scrum framework. The organization should be able to support the effort by committing the required resources. If the organization cannot commit the required resources, plans must be made to procure necessary resources such as people, tools, and workspace. It is imperative that an organization planning to use Scrum be prepared to drastically change its work culture and habits in order to truly realize the benefits of using Scrum.

In a Scrum environment where large projects are being developed, there will be numerous Scrum Teams competing for resources. Thus, it is important to manage the organizational resources in an optimal way to achieve the overall project objectives. The Large Project Scrum Organization should consider the components that will be developed; the skills, costs, and other resources required to develop them; the Scrum Teams' current velocities (or estimated/assumed velocities) to provide a high-level estimate for the project's duration; the communication requirements; and other interfaces that the Scrum Teams need to maintain.

13.2.2 Product Owners Collaboration Plan*

The Chief Product Owner works with key Product Owners to create the Product Owners Collaboration Plan. The Product Owners Collaboration Plan should specifically define how the Product Owners should collaborate with the Chief Product Owner. Minimally, this plan should define how many Scrum Teams a Product Owner can be assigned to (usually based on factors such as experience, time, and domain knowledge), how the work of gathering requirements from the business stakeholders will be allocated among the Product Owners, how the Prioritized Product Backlog will be updated with new requirements or changes in requirements, and how the Product Owners will collaborate with multiple Scrum Teams. It should be noted that each Scrum Team will still collaborate with only one Product Owner; however, a single Product Owner can collaborate with more than one Scrum Team, if required.

13.2.3 Scrum Masters/Scrum Teams Collaboration Plan*

The Chief Scrum Master works with key Scrum Masters, the Scrum Guidance Body, and at times some already-identified members of the Scrum Teams to create the Scrum Masters/Scrum Teams Collaboration Plan. The Scrum Masters/Scrum Teams Collaboration Plan defines how the numerous Scrum Teams will collaborate with each other to provide the highest value in the shortest possible time. A large project will have multiple Scrum Masters assigned—each facilitating and ensuring a productive work environment for their respective Scrum Team. It is possible that one Scrum Master may be facilitating more than one Scrum Team at the same time. Scrum Masters need to collaborate with other Scrum Masters as well as the Chief Scrum Master, the Chief Product Owner, and the Product Owners to develop the list of components and resources needed in common for all teams throughout the project. They also help provide inputs for the creation of the Release Readiness Plan.

The plan should include information on specialized domains assigned to qualified teams, how the teams will support Prioritized Product Backlog refining and estimation (i.e., which of the team members will participate in the refining sessions and high-level estimation exercises), and how the teams will organize the Scrum of Scrums (SoS) meetings. It may also be necessary to use an AI-powered Scrum Project Tool to facilitate the use of Scrum on large projects.

The Scrum Masters/Scrum Teams Collaboration Plan may also include information on how each Scrum Team will be coached. For example, it may include information on whether there will be a separate coach in addition to the Scrum Master; whether there will be a Scrum Master at each location in the case of distributed teams; how the team members will collaborate with colocated Scrum Masters and with Scrum Masters who are not colocated, and so on.

Although Scrum Teams working on a large project need to interact with each other when creating the Prioritized Product Backlog, Product Owners can ensure that User Stories with a lot of dependencies are grouped together, and owned by a single Product Owner so that deliverables will be worked on by only one or a few Scrum Teams. This minimizes the dependencies of tasks between different Scrum Teams, the more effectively each Scrum Team can work efficiently to create its deliverables.

13.2.4 Shared Resources*

Shared resources can include people, environment, and equipment that are needed by some or all of the Scrum Teams working on the project. In a large project, the shared resources are limited and may be needed by multiple Scrum Teams at the same time. In this context, the Chief Product Owner, the Chief Scrum Master, other Product Owners, and other Scrum Masters need to agree on a method for how shared resources will be allocated. An example of a method for allocating shared resources could be to ensure that resources are assigned first to those teams working on the most important/highest value features. When competing requests have similar priorities, the Chief Product Owner should decide the allocation of resources based on current business requirements, priorities, and other defined criteria.

13.2.5 Team Specialization*

There are three considerations of team specialization that should be considered when applying Scrum to large projects. The first dimension is the need for accomplishing a set of specific tasks. For example, one specialized team might be an integration team that has expert knowledge of continuous integration. This knowledge could be especially important when conducting a Release Readiness Sprint (if there is a need for specific tasks to be performed prior to a release).

The second dimension is the need for special skills of individual team members. Theoretically, all Scrum Team members are generalists and specialists in that they have general knowledge of various fields and are experts in at least one. However, this might not be the case in a large project. Members of the team may need to possess specific skills—such as special domain knowledge like security—which may not be available across all the teams working on the large project for which the skill is needed. In this case, it would be extremely costly to train everyone in all the required specialized domains.

Experts with specialized skills and knowledge can be assigned to the Scrum project and used as needed in different teams. However, at times it may be necessary to hire experts from external sources when they are needed, keeping in mind that adding a new team member will impact team velocity.

The third dimension for acquiring experts is due to any limitations in team flexibility. Typically, on a Scrum project, each team will have one or more domains in which they have significant expertise, a few domains they can work on with some input and training, and other domains in which they do not have the skills or experience. During Sprint planning, for every team, there will be a subset of User Stories that are logically assigned to the team based on their expertise, some that they can work on, and some that they may not be able to work on because they don't have the required knowledge or skills. In this case, experts may need to be acquired when there is not enough flexibility in the skills of team members.

Limited access to resources with specialized skills on a large Scrum project results in some level of risk to the project. Some top-priority User Stories may not be able to be completed in a single Sprint. Teams may need to work on some lower-priority User Stories while waiting for the availability of team members with specialized skills.

13.2.6 Environment and Environment Schedule*

The required development environments may not always be available throughout the duration of a large project. For example, a crane needed for a construction activity, or a specialized testing environment may only be available on specific days. Once all the required environments are identified, an Environment Schedule is created and used for the coordination of Sprint activities in the *Update Sprint Backlog* process. The Environment Schedule is a calendar detailing how the Scrum Teams will access and/share any specific environments. The schedule allocates days and time periods for when each team can use each environment.

13.2.7 Release Readiness Plan*

The Chief Product Owner works with other Product Owners to create the Release Readiness Plan. The Release Readiness Plan details the steps to be taken by relevant Scrum Teams and any other individuals to confirm that the minimum requirements for release have been met, and the product or product increment is ready for release. The business decisions with their associated business justifications for performing release readiness tasks are also described in the Release Readiness Plan.

Because every Sprint creates a potentially shippable product or other deliverable, in a typical small Scrum project, a release can occur after any Sprint when it makes business sense to do so. However, in a large Scrum project, there may be certain activities related to release preparedness that should be performed for some Sprints in the project. For example, a project team may need to run a full set of expensive and time-consuming performance tests or conduct a special set of end-to-end integration tests just prior to a release. These activities are considered outside the defined Done Criteria for regular Sprints, and in such cases, a separate Release Readiness Sprint will be necessary (see section 13.3.7) to complete the tasks needed to prepare for a release.

13.3 Additional Tools for Large Projects

13.3.1 Large Project Communications Plan

The Communications Plan for a large project is created by the Chief Product Owner, Chief Scrum Master, and Scrum Guidance Body, with input from other Product Owners, Scrum Masters, Scrum Teams, and other relevant persons. A Large Project Communications Plan is essential in a large project as miscommunication or lack of communication on the project can be detrimental to the collaborative efforts and can result in project failure. A Communications Plan should include information pertaining to how communications will take place across all Scrum Teams and to business stakeholders, including the communication methods to be used, the communication channels or mechanisms to communicate key information, the responsibilities for communications, the classification of and means to deal with sensitive information, the timing of communication activities, and the processes to assess communication effectiveness. The Communications Plan should also include the timing and frequency of the Scrum of Scrums Meetings (SoS) and how these meetings will be conducted.

Each Scrum Team may also have its own Communications Plan (see section 12.1.3.4) that specifies the records that must be created and distributed, and how these records will be maintained throughout the project. The plan should also include the methods to be used to convey important project information to business stakeholders and who is responsible for all the various communication activities. Since it may be exceedingly difficult for everyone in a large project to be colocated, using an AI-powered Scrum Project Tool may help facilitate effective communication.

13

13.3.2 Large Project Resource Planning*

Resource planning for large projects is essential due to the complexity of allocating several types of resources to the numerous Scrum Teams working in parallel. There will be competing needs for scarce resources, and the Chief Product Owner and other Product Owners must plan for delivering the greatest value in the shortest amount of time. Resource planning in a large project should take into consideration the various costs associated with resources such as people, training, hardware and software, external services, and physical space. The Chief Product Owner and other Product Owners may have to coordinate with external sources to acquire resources and augment staff (e.g., external resources may need to be hired to work with the existing full-time team and may also need to interact with the existing vendor management team within the organization). When hiring external resources, the Chief Product Owner and team must comply with organizational policies for dealing with external resources and vendors.

In large projects, the Chief Product Owner may also need to consider additional resource planning to address the needs of specialized teams and the need for setting up environments for numerous Scrum Teams working in parallel. The Chief Product Owner and the Product Owners can collaborate with Scrum Masters and Scrum Teams to define the specialized skills required for the large project, the number of resources required, the Scrum Teams that need specialized skills, and allocation estimation.

13.3.3 Environment Identification*

The Chief Scrum Master works with other Scrum Masters, relevant Product Owners, Scrum Team members, Supporting Services, the Scrum Guidance Body, and other experts as required to identify the required and appropriate environments that will be needed to effectively complete the large project. This should preferably happen once during the Initiate phase or when required by the teams and/or the SGB.

In a large project, it is important to identify the number and types of environments needed because numerous Scrum Teams will be working simultaneously to carry out the work of their respective Sprints and environment needs could be complicated and may conflict. Some examples of environments include software development or test areas, physical work areas, or environments with specialized equipment. Process boundaries for each Scrum Team may also affect environments. Also, with distributed teams working in different time zones, it may be possible to conduct 24-hour testing and maximize the use of different environments. Therefore, it is imperative to create an Environment Schedule which shows the testing times for each team. For software projects, the Environment Schedule can also include information on how and by whom code will be promoted to each environment.

13.3.4 Prioritized Product Backlog Assignment*

In a large project, the Prioritized Product Backlog (with its corresponding Epics and User Stories) is created by the Chief Product Owner and other Product Owners in much the same way as for typical small Scrum projects.

The creation of Epics and User Stories by the Product Owners may depend on multiple factors such as:

- How the requirements were collected from the business stakeholders
- Knowledge and experience/skill set of the Product Owners

The Chief Product Owner and other Product Owners work together to review the Prioritized Product Backlog and determine which Epics and User Stories will be owned by each Product Owner. Although some Epics and User Stories may be created by a particular Product Owner, they may be assigned to another Product Owner to manage and implement as part of the large Scrum project. Epics and User Stories must be prioritized, and some pre-existing estimates may be used to facilitate their assignment to different Product Owners.

The assignment of Epics and User Stories to Product Owners is influenced by other factors such as:

- The Product Owner who created the Epic/User Story (as he/she will often own its implementation)
- The Product Owner who has the team with the appropriate skill sets to complete the Epic/User Story
- The specific customer, sponsor, or organization linked to the Epic/User Story (as there could be a past relationship with a particular Product Owner)
- The number of dependencies pertaining to each Epic/User Story (as grouping User Stories with a lot of dependencies under one Product Owner could allow the teams to work relatively independent of each other, without the need to spend too much time in coordination with other teams)

The assignment and prioritization of Epics or User Stories to Product Owners can be done over a period of time through meetings and if possible, through the use of an AI-powered Scrum Project Tool.

13.3.5 Scrum of Scrums (SoS) Meeting*

A Scrum of Scrums (SoS) Meeting is an essential element when scaling Scrum to large projects. The objective of most Scrum of Scrum meetings is the synchronization of teams during the creation of deliverables but could also be used for sharing best practices after Retrospect Sprint Meetings, and for planning future Sprints. The frequency of SoS meetings is project-specific and depends on project size and complexity, dependencies between different teams, etc. If Epics or User Stories in Sprints can be completed without too much interaction with other Epics or User Stories, these meetings may not be required too often; on the other hand, if the dependencies are high, then a higher frequency of SoS meetings may be required.

Typically, there is one representative in the SoS meeting from each Scrum Team—usually the Scrum Master—but it is also common for other members of a Scrum Team to attend the meeting if this makes sense. This meeting is usually facilitated by the Chief Scrum Master and is intended to focus on areas of coordination and integration between the different Scrum Teams.

These are preferably short meetings where at least one representative from each Scrum Team (e.g., Scrum Master and/or others) meet to share the status of the work being done by their respective teams, somewhat similar to the Daily Standup Meeting. They are usually not Time-boxed to allow all representatives to share their information, even for exceptionally large projects. The Scrum of Scrums (SoS) Meeting is held at predetermined intervals or when required by Scrum Teams to facilitate the necessary sharing of information among the various Scrum Teams. Issues, dependencies, and risks impacting multiple Scrum Teams can be closely monitored, which helps the multiple teams working on a large project to better coordinate and integrate their work. It is the responsibility of the Chief Scrum Master (or another Scrum Master who facilitates the SoS Meetings) to ensure that all representatives have an environment conducive to open and honest sharing of information, including feedback to other team representatives. For larger projects, involving a considerable number of teams, multiple levels of SoS meetings may be convened to share the status of each respective team.

Each Scrum Team representative provides updates from his or her team in turn. These updates are usually provided in the form of answers to the following four specific questions.

1. What has my team been working on since the last meeting?
2. What will my team do until the next meeting?
3. What were other teams counting on our team to finish that remains undone?
4. What is our team planning to do that might affect other teams?

The answers to these four questions provide information that allows each team to clearly understand the work status of all other teams and if there are or could be any issues with upcoming deliveries.

13.3.6 Release Preparation Methods*

Release preparation methods are the methods that will be used to execute the tasks identified in the Release Readiness Plan to prepare the deliverables to be shipped or released. These methods can be project specific, but more likely are valid on a program or portfolio level. The Scrum Guidance Body may define them.

13.3.7 Release Readiness Sprint

If there is a need for specific tasks to be performed to prepare for a release and to confirm that the minimum requirements for release have been met, these tasks are performed in a Release Readiness Sprint. In a Release Readiness Sprint, no User Stories from the Prioritized Product Backlog are developed. Instead, the tasks as identified in the Release Readiness Plan (see 13.2.7) are performed. A Release Readiness Sprint is only conducted once per release as the first step in the *Ship Deliverables* process. The length of a Release Readiness Sprint may be different from the length of other Sprints.

If there is a Release Readiness Sprint, its corresponding Done Criteria are typically unique and differ from the Done Criteria of the User Stories for other Sprints (which must still be met). Done Criteria are defined with the purpose of ensuring that the Sprint deliverables are “potentially shippable.” The Release Readiness Sprint addresses all the activities that are only done once per release based on deliberate business decisions as justified in the Release Readiness Plan.

A Release Readiness Sprint is not mandatory unless there is a justified business decision to incorporate it into the project. Also, typically only the relevant team members are involved in the Release Readiness Sprint. Other Scrum Team members not involved in the Release Readiness Sprint can begin to work on other regular Sprints.

13.3.8 AI-powered Scrum Project Tool

Since large projects involve many Scrum Teams with several hundred or more people working on the project, and since teams may also be distributed, there can be significant complexity and interactions between the teams. It would therefore be beneficial for the teams to have access to a structured AI-powered Scrum Project Tool, or set of tools that can be used to automate processes, manage complexity, share information (between teams and to business stakeholders), generate reports, and so on.

Some specific tasks that could be managed by the AI-powered Scrum Project Tool on a large project include:

- Ability to form teams with appropriate roles and the ability to scale roles to large projects
- Ability to create and maintain a Prioritized Product Backlog for each team, including the creation, estimation, and management of Epics, User Stories, and Tasks
- Ability to support other important Scrum project artifacts, such as Scrumboards, Sprint Backlogs, meeting schedules/minutes, and so on
- Ability to allow for seamless and effective communication between all project team members
- Ability to support distributed teams (see section 2.5.3)
- Ability to support the creation of reports and metrics as required by different Scrum roles
- Ability to capture and share recommendations or expertise from the Scrum Guidance Body (e.g., lessons learned from Retrospective Sprints, best practices, Scrum-related organizational policies, and so on)

14. SCALING SCRUM FOR THE ENTERPRISE

This chapter emphasizes additional aspects of Scrum that are applicable to programs and portfolios. Scaling Scrum for the enterprise, as defined in *A Guide to the Scrum Body of Knowledge (SBOK® Guide)*, is applicable to the following:

- Programs, portfolios, and/or projects in any industry
- Products, services, or any other results to be delivered to business stakeholders

The term “program” in the *SBOK® Guide* refers to a collection of related projects and/or subprograms that must be managed in a coordinated fashion to produce and deliver program components. The term “portfolio” refers to a collection of programs and/or projects within the same organization that may or may not be related to each other and may or may not need to be managed in a coordinated fashion to meet portfolio objectives. From the Scrum framework perspective, programs and portfolios can be treated in a similar fashion but at various levels in the enterprise and thus potentially require different amounts of coordination for the underlying programs and/or projects.

This chapter addresses the impacts (to inputs, tools, and outputs) that a program or portfolio has on the fundamental Scrum processes described in chapters 8 through 12. This chapter also introduces additional processes that apply solely to programs and portfolios that are not relevant at a project level. To facilitate the best application of the Scrum framework, this chapter identifies inputs, tools, and outputs for each process as either “mandatory” or “optional.” Inputs, tools, and outputs denoted by asterisks (*) are mandatory, or considered critical for project success, whereas those with no asterisks are optional.

It is recommended that those individuals being introduced to the application of the Scrum framework for the enterprise focus primarily on the mandatory inputs, tools, and outputs; while Program/Portfolio Product Owners, Chief Product Owners, Product Owners, Program/Portfolio Scrum Masters, Chief Scrum Masters, Scrum Masters, and other more experienced Scrum practitioners strive to attain a more thorough knowledge of the information in this entire chapter. It is also important to realize that although all processes are defined uniquely in the *SBOK® Guide*, they are not necessarily performed sequentially or separately. At times, it may be more appropriate to complete some processes in parallel or iteratively, depending on the specific requirements of each program or portfolio. This chapter is written from the perspective of a single program or portfolio team that coordinates and prioritizes activities of multiple underlying Scrum projects and/or programs. Additional information pertaining to the application of Scrum at the project level can be found in chapters 2 through 7, which cover Scrum principles and Scrum aspects.

Enterprise Scrum vs. Single Scrum Project

When dealing with Scrum at an enterprise level, there may be several hundred Scrum Teams, with several thousand people working in multiple projects within programs and/or portfolios in the company. Applying Scrum processes at a program or portfolio level will have certain impacts on the underlying projects. In general, the Scrum projects are still executed using the fundamental Scrum processes discussed in chapters 8 through 12 for typical small projects; with the additional inputs, tools, and outputs outlined in chapter 13 for large projects (having multiple Product Owners and/or Scrum Masters).

The impacts of programs and portfolios to the Scrum project-level processes described in chapters 8 through 12 are outlined in section 14.1 of this chapter as additional inputs, tools, and outputs. The additional processes and considerations that are relevant only at the program or portfolio level are addressed in sections 14.2 through 14.8.

Some of the questions that arise at the program or portfolio level are similar to those that come up on a large Scrum project. The synchronization between teams and overall collaboration are the biggest challenges faced in a large Scrum project, and these challenges also exist at the program or portfolio level. The biggest challenges for a program or portfolio, however, may occur on the business side, because the business priorities of different projects may conflict with each other and may sometimes also conflict with the overall goals of the program or portfolio and need to be aligned. As in a large Scrum project, additional inputs, tools, and outputs are required to address additional prioritization, alignment, and coordination efforts. Some reasons for additional inputs, tools, and outputs needed for programs and/or portfolios are as follows:

Product Owners

- Program and Portfolio—Need for alignment of conflicting business goals
- Program—Need for collaboration between the Program Product Owner and the Product Owners from the projects in the program, such as:
 - refining of the Prioritized Program Backlog,
 - interfacing with business stakeholders to synchronize messages, and
 - avoiding duplication of work within the program (i.e., synergy)
- Portfolio—Need for collaboration between the Portfolio Product Owner, the Program Product Owners, and the Product Owners from the programs and projects in the portfolio, such as:
 - refining of the Prioritized Portfolio Backlog,
 - interfacing with business stakeholders to synchronize messages, and
 - avoiding duplication of work in the portfolio (i.e., synergy)

Scrum Masters

- Program and Portfolio—Need for collaboration between Scrum Masters when addressing impediments
- Program—Synchronizing the work of the Scrum Teams from multiple projects, if required
- Portfolio—Synchronizing the work of the Scrum Teams from multiple programs and projects, if required

Scrum Teams

- Program and Portfolio—Need to manage dependencies among Scrum Teams
- Program and Portfolio—Need to manage shared resources and any resource conflicts between Scrum Teams
- Program and Portfolio—Need to define certain guidelines and standards that should be adhered to by Scrum Teams for all projects of the program or portfolio (e.g., security standards within the organization or legal and/or government regulations for specific industries)—these may need to be documented by the Scrum Guidance Body.
- Program and Portfolio—Requirement to set up and maintain an environment to be used by multiple Scrum Teams

14.1 Impact of Programs or Portfolios to Fundamental Scrum Processes

Tables 14-1 through 14-4 outline a summary of the impacts of programs and portfolios to the fundamental Scrum processes for each project phase.

14.1.1 Initiate

The additional inputs from the program/portfolio level that need to be taken into consideration for the Initiate phase are as follows:

Process	Summary of Impacts of a Program or Portfolio
8.1 Create Project Vision	The project vision is created with additional input from the program/portfolio the project belongs to. Beyond that, there is no other change to this process. Additional Input: Program/Portfolio Product Owner The roles of the Program/Portfolio Product Owner are described in sections 3.7.4.1 and 3.7.4.2. Additional Input: Program/Portfolio Scrum Master The roles of the Program/Portfolio Scrum Master are described in sections 3.7.4.3 and 3.7.4.4. Additional Input: Program/Portfolio Business Stakeholders Program/portfolio business stakeholders are described in section 14.3.3.5. They influence all projects in the program/portfolio. Additional Input: Prioritized Program/Portfolio Backlog The Prioritized Program/Portfolio Backlog is described in section 14.6.1.2. The Prioritized Program/Portfolio Backlog contains requirements for the program or portfolio that may impact the Project Vision.

Process	Summary of Impacts of a Program or Portfolio
8.2 Identify Scrum Master and Business Stakeholder(s)	The Scrum Master and the business stakeholder(s) of the project are identified with additional input from the program/portfolio the project belongs to. Beyond that, there is no other change to this process. Additional Input: Program/Portfolio Product Owner The roles of the Program/Portfolio Product Owner are described in sections 3.7.4.1 and 3.7.4.2. Additional Input: Program/Portfolio Scrum Master The roles of the Program/Portfolio Scrum Master are described in sections 3.7.4.3 and 3.7.4.4. Additional Input: Program/Portfolio Business Stakeholders Program/portfolio business stakeholders are described in section 14.3.3.5. They influence all projects in the program or portfolio.
8.3 Form Scrum Team	There is no additional impact to this process for a project in a program or portfolio.
8.4 Develop Epic(s)	The Epics are developed with additional input from the program/portfolio the project belongs to. Beyond that, there is no other change to this process. Additional Input: Prioritized Program/Portfolio Backlog The Prioritized Program/Portfolio Backlog is described in section 14.6.1.2. The Prioritized Program/Portfolio Backlog contains requirements for the program or portfolio that may impact the development of Epics. Additional Input: Program/Portfolio Risks Program and portfolio risks are described in sections 7.6.1 and 7.6.2. Risks related to a program or portfolio also impact the projects that are part of the respective program or portfolio. During risk assessment of the program or portfolio, if it is determined that a risk may affect an individual project, relevant information about the risk must be communicated to the Product Owner and Scrum Team. Program and portfolio risks become inputs to the <i>Develop Epic(s)</i> process and these risks can have an impact on how this process is conducted.

Process	Summary of Impacts of a Program or Portfolio
8.5 Create Prioritized Product Backlog	The Prioritized Product Backlog is created with additional input from the Prioritized Program/Portfolio Backlog. Beyond that, there is no other change to this process. Additional Input: Prioritized Program/Portfolio Backlog The Prioritized Program/Portfolio Backlog is described in section 14.6.1.2. The Prioritized Program/Portfolio Backlog contains requirements for the Program/Portfolio that may impact the creation of the Prioritized Product Backlog.
8.6 Conduct Release Planning	Release Planning is conducted with additional input from the program/portfolio the project belongs to. Beyond that, there is no other change to this process. Additional Input: Program/Portfolio Product Owner The roles of the Program/Portfolio Product Owner are described in sections 3.7.4.1 and 3.7.4.2. Additional Input: Program/Portfolio Scrum Master The roles of the Program/Portfolio Scrum Master are described in sections 3.7.4.3 and 3.7.4.4. Additional Input: Prioritized Program/Portfolio Backlog The Prioritized Program/Portfolio Backlog is described in section 14.6.1.2. The Prioritized Program/Portfolio Backlog may contain key coordination dates and/or deadlines for some requirements that the project need to adhere to.

Table 14-1: Impact of a Program or Portfolio to Fundamental Scrum Processes—Initiate Phase

Figure 14-1 displays the "Create Release" screen in Vabro, where users define a release name, description, and target release date before saving and continuing.

Figure 14-1: Creating a Release during Initiation (Source: Vabro)

14.1.2 Plan and Estimate

Using Scrum on a program/portfolio level has no impact on the Plan and Estimate phase of the respective projects in the program/portfolio.

14.1.3 Implement

The additional inputs from the program/portfolio level that need to be taken into consideration for the Implement phase are as follows:

Process	Summary of Impacts of a Program or Portfolio
10.1 Create Deliverables	Creation of the deliverables is not impacted by using Scrum at the program or portfolio level.

10.2 Conduct Daily Stand Up	Conducting the Daily Stand Up is not impacted by using Scrum at the program or portfolio level.
10.3 Refine Prioritized Product Backlog	When refining the Prioritized Product Backlog, new or changed requirements from the program or portfolio level need to be prioritized and incorporated into the Prioritized Product Backlog as appropriate. Additional Input: Program/Portfolio Product Owner The roles of the Program/Portfolio Product Owner are described in chapters 3.7.4.1 and 3.7.4.2. In this process, the Program/Portfolio Product Owner communicates any updated requirements from the program or portfolio level to the project. Additional Input: Prioritized Program/Portfolio Backlog The Prioritized Program/Portfolio Backlog is described in section 14.6.1.2. Any changes in the Prioritized Program/Portfolio Backlog need to be incorporated into the project's Prioritized Product Backlog.

Table 14-2: Impact of a Program or Portfolio to Fundamental Scrum Processes—Implement Phase

14.1.4 Review and Retrospect

Representatives from the program or portfolio may provide feedback during the Sprint reviews or retrospectives. Beyond that, there is no other change to the Review and Retrospect phase of a project.

Process	Summary of Impacts of a Program or Portfolio
11.1 Demonstrate and Validate Sprint	Representatives from the program/portfolio may provide feedback. Beyond that, there is no other change to this process. Additional Input: Program/Portfolio Product Owner The roles of the Program/Portfolio Product Owner are described in sections 3.7.4.1 and 3.7.4.2. Additional Input: Program/Portfolio Business Stakeholders Program/portfolio business stakeholders are described in section 14.3.3.5.
11.2 Retrospect Sprint	The <i>Retrospect Sprint</i> process is not impacted by using Scrum at the program or portfolio level.

Table 14-3: Impact of a Program or Portfolio to Fundamental Scrum Processes—Review & Retrospect Phase

14.1.5 Release

Applying Scrum at the program/portfolio level may have an impact on individual project releases, since there may be dependencies between the different project releases. For example, if the deliverables of two projects, A and B, should ideally be released together, but the deliverables of project A are delayed, then this may impact the release of the deliverables of project B, even if the deliverables of project B are completed on time.

Figure 14-2 shows a "Program Release Dashboard" from the Vabro platform, tracking multiple software releases, their progress, user stories, and current status.

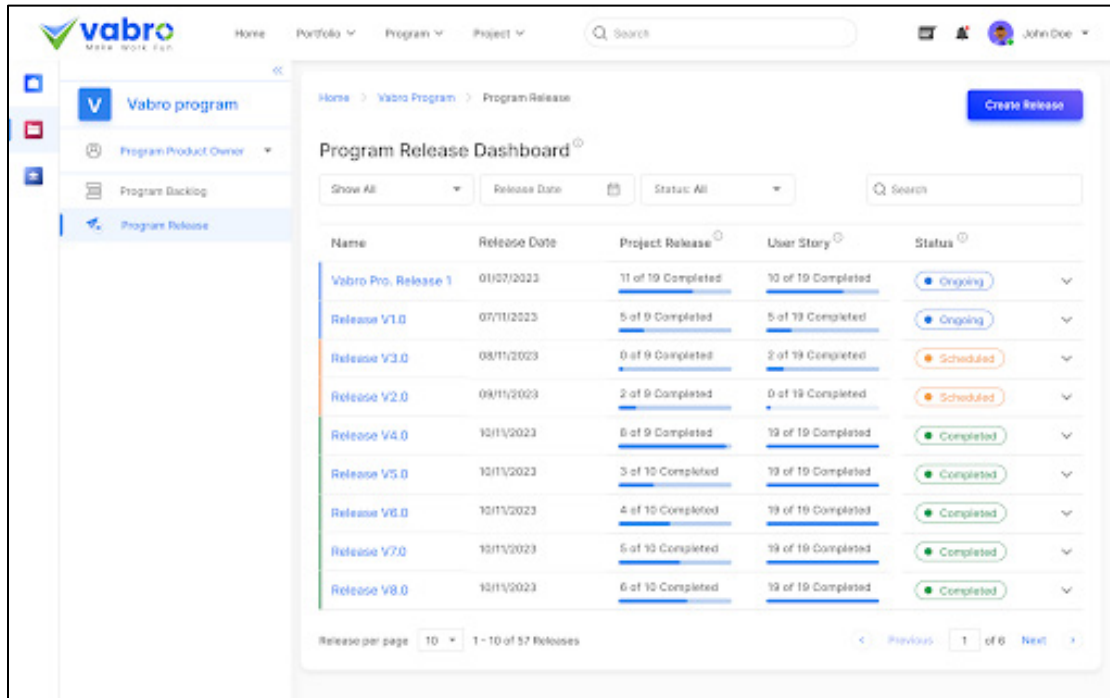


Figure 14-2: Program Release Dashboard (Source: Vabro)

Process	Summary of Impacts of a Program or Portfolio
12.1 Ship Deliverables	Deliverables for a program or portfolio are created in the same way as they are for individual projects. However, they may be some dependencies on deliverables from other projects that need to be coordinated by the Program/Portfolio Product Owner or the Program/Portfolio Scrum Master. Updated Input: Program/Portfolio Product Owner Described in sections 3.7.4.1 and 3.7.4.2. Updated Input: Program/Portfolio Scrum Master Described in sections 3.7.4.3 and 3.7.4.4.
12.2 Retrospect Release	The <i>Retrospect Release</i> process is not impacted by using Scrum at the program or portfolio level.

Table 14-4: Impact of a Program or Portfolio to Fundamental Scrum Processes—Release Phase

14.2 Additional Processes to Scale Scrum for the Enterprise (Program/Portfolio)

All Scrum projects that are part of a larger program or portfolio can apply Scrum processes, as described in chapters 8 through 13, with the additional impacts as described in Section 14.1.

However, when scaling Scrum for the enterprise, certain additional processes may be required to manage the added complexity of several hundreds or thousands of people working on the associated projects, and the additional coordination requirements at a program or portfolio level. All these processes are not necessarily sequential and can be applied in parallel and iteratively as required by the enterprise.

Figure 14-3 presents an overview of scaling scrum at the enterprise level. Describes how multiple scrum teams and organizational layers interact to deliver complex products through synchronized planning and collaboration.

14.3 Create/Update Program or Portfolio Teams—In this process, additional roles are created or identified to manage programs and portfolios. These roles include Program Product Owner, Portfolio Product Owner, Program Scrum Master, Portfolio Scrum Master, business stakeholders, and supporting services.

14.4 Create/Update Program or Portfolio Components—In this process, the program or portfolio Product Owners, Scrum Masters, and business stakeholders identify and create the common components and resources required for the program or portfolio. The Minimum Done Criteria are defined, and all other relevant business stakeholders are identified. Dependencies between projects are addressed, common impediments are discussed, and best practices are shared. Sometimes, recommendations for improvements to the Scrum Guidance Body are made.

14.5 Review and Update Scrum Guidance Body—In this process, the Scrum Guidance Body recommendations are regularly reviewed by the members of the Scrum Guidance Body and are updated when necessary. Changes in the membership of the Scrum Guidance Body are also addressed. The main objective of this process is to constantly monitor and work towards improving the productivity of the Scrum projects, programs, and portfolios within the organization.

14.6 Create/Refine Prioritized Program or Portfolio Backlog—In this process, the Program or Portfolio Backlog is first created based on the program or portfolio requirements. On an ongoing basis, the Prioritized Program or Portfolio Backlog is reviewed to add or update requirements, risks, and priorities.

14.7 Create/Update Program or Portfolio Releases—In this process, the program or portfolio releases are planned, considering any dependencies between the releases. Program or portfolio release planning will impact release planning at the project level. The Program or Portfolio Release Schedule is created and should be revisited regularly based on the progress of project deliverables, new or changed requirements or their priorities, and other factors.

14.8 Retrospect Program or Portfolio Releases—In this process, the Program or Portfolio Product Owner and business stakeholders get together to retrospect a program or portfolio release and to also discuss and internalize the lessons learned. Often, these lessons learned lead to Agreed Actionable Improvements to be implemented in future releases. Sometimes, improvements to the Scrum Guidance Body may be recommended.

14.3 Create/Update Program or Portfolio Teams	14.4 Create/Update Program or Portfolio Components	14.5 Review and Update Scrum Guidance Body
INPUTS 1. Company Vision and Mission* 2. Senior Management* 3. Organizational Resource Matrix 4. Consultants TOOLS 1. Company Human Resource Plan* 2. Stakeholders Analysis* OUTPUTS 1. Portfolio Product Owner* 2. Program Product Owner* 3. Portfolio Scrum Master* 4. Program Scrum Master* 5. Business Stakeholders* 6. Supporting Services*	INPUTS 1. Company Vision and Mission* 2. Portfolio Product Owner* 3. Portfolio Scrum Master* 4. Program Product Owner* 5. Program Scrum Master* 6. Organizational Resource Matrix 7. Scrum Guidance Body Recommendations 8. Business Stakeholders TOOLS 1. Communication Plan(s)* 2. Company Human Resource Plan* 3. Stakeholder Analysis* 4. Scrum of Scrum (SoS) Meeting* 5. Scrum of Scrum of Scrums (SoSoS) Meeting 6. Communication Techniques 7. AI-powered Scrum Project Tool OUTPUTS 1. Minimum Done Criteria* 2. Shared Resources* 3. Identified Business Stakeholders* 4. Updated Impediment Logs* 5. Updated Dependencies* 6. Product Owners Collaboration Plan* 7. Scrum Masters/Scrum Teams Collaboration Plan* 8. Recommended Scrum Guidance Body Improvements	INPUTS 1. Regulations* 2. Recommended Scrum Guidance Body Improvements* 3. Scrum Guidance Body Members* TOOLS 1. Members Selection Criteria* 2. Scrum Guidance Body Meetings* 3. Performance Reports 4. Benchmarking OUTPUTS 1. Updated Scrum Guidance Body Recommendations* 2. Actionable Escalations 3. Updated Scrum Guidance Body Members 4. Rejected Updates to the Scrum Guidance Body Recommendations

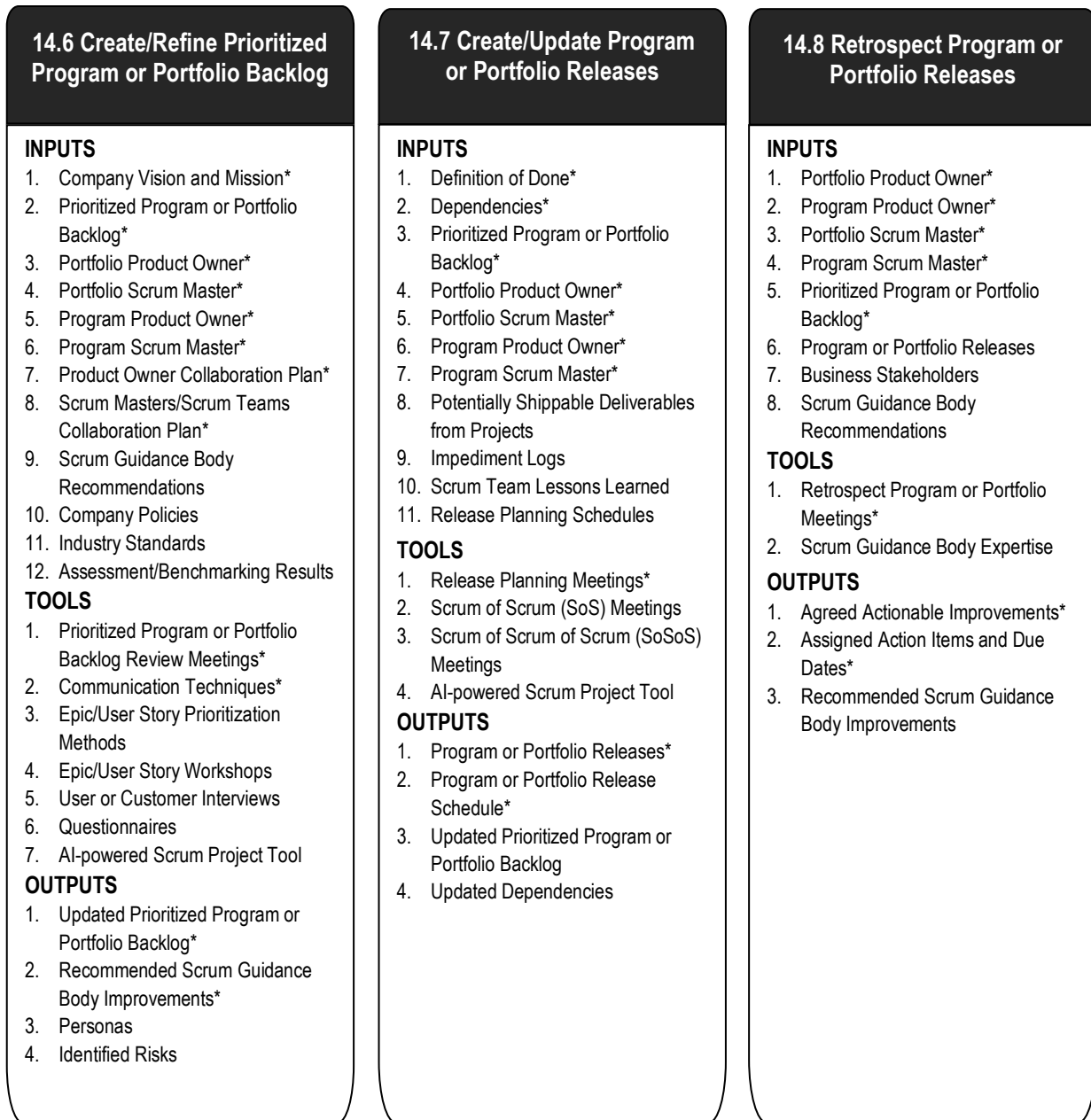


Figure 14-3: Scaling Scrum for the Enterprise

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

14.3 Create/Update Program or Portfolio Teams

Program or portfolio teams need to be created and/or identified before Scrum processes can be applied in an enterprise environment. Some important roles include Program Product Owner, Portfolio Product Owner, Program Scrum Master, Portfolio Scrum Master, business stakeholders, and supporting services.

It is important to note that all these people need not be identified and assigned at the beginning of the program or portfolio. Some key persons may be identified early on, while others may be assigned over time depending on the specific requirements of the program or portfolio.

Outputs from the *Create/Update Program or Portfolio Teams* process become inputs to the process after the initial team is created. For example, once the Program Product Owner or Program Scrum Master are identified, these individuals would be involved with any future changes or updates to the program team.

Figure 14-4 shows all the inputs, tools, and outputs for the *Create/Update Program or Portfolio Teams* process.



Figure 14-4: Create/Update Program or Portfolio Teams—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

14.3.1 Inputs

14.3.1.1 Company Vision and Mission*

The company vision and the company mission are important inputs for any project, but they are even more crucial in planning programs and especially portfolios. Programs and portfolios should be driven by the overall mission and vision of the enterprise as this ensures unity of effort throughout the organization. Understanding the company vision helps the portfolio or program teams keep their focus on the organization's objectives and the future potential of the company. The Program or Portfolio Product Owners can take guidance and direction from the company vision to create each Project Vision Statement. The company mission provides a framework for formulating the strategies of the organization and guides overall decision making. The Project Vision Statement must be framed such that its fulfillment helps the organization fulfill its mission.

14.3.1.2 Senior Management*

Senior management includes senior representatives from within the internal organization of the company that are benefiting from, or responsible for, the program or portfolio deliverables. Senior management of the company could include the Chief Executive Officer, Chief Technology Officer, Chief Finance Officer, Vice Presidents, Directors and other senior employees from different divisions within the organization.

14.3.1.3 Organizational Resource Matrix

The Organizational Resource Matrix at a program or portfolio level should include the employees in the organization who have the skills and availability to perform senior roles related to Scrum projects. For more information on the Organization Resource Matrix, see section 8.2.1.5.

14.3.1.4 Consultants

If all the skills required to set up and manage Scrum programs or portfolios within the organization are not available internally within the company, then external consultants may be leveraged to guide the senior management in establishing the necessary program or portfolio teams.

14.3.2 Tools

14.3.2.1 Company Human Resource Plan*

The company's Human Resource Plan provides general information on when particular personnel will be available for various projects, programs, and portfolios. The plan also provides information about skills and capabilities available inside the company, and on plans for hiring personnel required for future efforts.

14.3.2.2 Stakeholder Analysis

Various stakeholder analysis techniques can be used to identify and analyze business stakeholders and any other relevant stakeholders impacted at the program and portfolio levels. Stakeholder analysis techniques can also be used to assess the interests, involvement, and potential impact of each identified stakeholder to the program or portfolio. Stakeholder analysis is also helpful in understanding the communication and involvement requirements needed to benefit the program or portfolio.

14.3.3 Outputs

14.3.3.1 Portfolio Product Owner*

Described in sections 3.7.4.2.

14.3.3.2 Program Product Owner*

Described in section 3.7.4.1.

14.3.3.3 Portfolio Scrum Master*

Described in sections 3.7.4.4.

14.3.3.4 Program Scrum Master*

Described in section 3.7.4.3.

14.3.3.5 Business Stakeholders*

In this process, business stakeholders that will play key roles in the program or portfolio are identified, including customers, users, and sponsors of the program or portfolio. Business stakeholders influence the program or portfolio itself and also influence the associated projects (within the program or portfolio) throughout each project's development. Program or portfolio business stakeholders can also help define the project/program/portfolio vision and provide guidance regarding business value. Program business stakeholders interface with the portfolio business stakeholders to support the Program Product Owner and Portfolio Product Owner to ensure alignment of the program with the goals and objectives of the portfolio. Program and/or portfolio business stakeholders are also involved with identifying business stakeholders for the individual projects and ensuring that the vision, objectives, outcomes, and releases of those associated projects within the program/portfolio align with those of the program/portfolio. At the portfolio level, business stakeholders could include members of the executive board of a company or government organization. At the program level, business stakeholders may include senior executives and the sponsor(s) of the program and associated projects.

14.3.3.6 Supporting Services*

At a program or portfolio level, support services should include people or groups responsible for managing training, logistics, marketing, finance, infrastructure, architecture, and other supporting services required for the successful functioning of the program or portfolio. Some of these people may also be working full-time for the program or portfolio. For more information on supporting services, see section 3.2.2.

14.4 Create/Update Program or Portfolio Components

In this process, the Program or Portfolio Product Owners, Scrum Masters, and business stakeholders identify and create the common components and resources required for the program or portfolio. The Minimum Done Criteria are defined, and all other relevant business stakeholders are identified. Dependencies between projects are addressed, common impediments are discussed, and best practices are shared. Sometimes, recommendations for improvements to the Scrum Guidance Body are made.

It is important to note that all program or portfolio components need not be created at the beginning. This is typically an iterative process as some key program or portfolio components are created early on, and others could be created or updated later as more information becomes available.

Outputs from the *Create Program or Portfolio Components* process become inputs to the process after their initial creation. For example, once the Minimum Done Criteria are identified, they will become inputs when the program or portfolio components are being updated.

Figure 14-5 shows all the inputs, tools, and outputs for the *Create/Update Program or Portfolio Components* process.

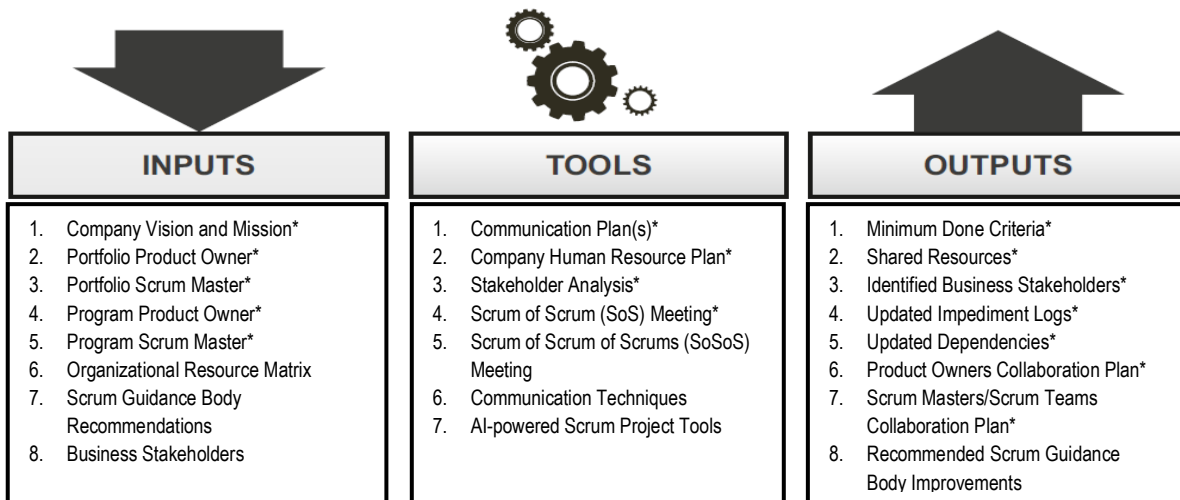


Figure 14-5: Create/Update Program or Portfolio Components—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

14.4.1 Inputs

14.4.1.1 Company Vision and Mission*

Both the company vision and the company mission are important inputs for any project, but even more so for programs and portfolios. Programs and portfolios should be driven by the overall mission and vision of the enterprise as this ensures unity of effort throughout the organization. For more information on the company vision and mission, see section 14.3.1.1.

Figure 14-6 shows a Vision Statement Worksheet with five prompts to define an organization's culture, goals, achievements, keywords, and a visual representation.

VISION STATEMENT WORKSHEET

ORGANIZATION NAME	ANSWERS
1. What do we want our organization to look like? Culture? Ethos? Mood?	
2. Where are we going?	
3. What can we realistically achieve?	
4. What words or phrases depict the type of organization and end-goals we want?	

5. Draw or insert a picture or clip art that represents a vision for the organization.

Figure 14-6: Company Vision Statement Template (Source: Smartsheet)

Figure 14-7 shows a Mission Statement Worksheet outlining six steps to create a mission statement, from asking questions to publishing, ending with space for the final statement.

MISSION STATEMENT WORKSHEET		
Answer these questions to find details and structure for your mission statement.		
PROCEDURES	ACTIVITY	ANSWERS
1. Ask questions	Identify what your organization does, what your organization creates, and who the products and services are geared towards.	
2. Discuss answers	Write down the ideas, words, and phrases that the answers in step one inspires.	
3. Edit ideas and keywords	Start to winnow the thoughts and keywords from your discussion.	
4. Draft mission statements	Begin drafting mission statements. Try creating several.	
5. Review	Share drafts with the team for their input.	
6. Publish your final effort	Incorporate feedback, copy edit the statement, and publish to the world.	
THE FINAL MISSION STATEMENT		

Figure 14-7: Company Mission Statement Template (Source: Smartsheet)

14.4.1.2 Portfolio Product Owner*

Described in section 3.7.4.2.

14.4.1.3 Portfolio Scrum Master*

Described in section 3.7.4.4.

14.4.1.4 Program Product Owner*

Described in section 3.7.4.1.

14.4.1.5 Program Scrum Master*

Described in section 3.7.4.3.

14.4.1.6 Organizational Resource Matrix

Described in section 8.2.1.5.

14.4.1.7 Scrum Guidance Body Recommendations

Scrum Guidance Body recommendations are especially important at the program and portfolio levels as appropriate guidance is needed for a potentially considerable number of related projects. For more information on Scrum Guidance Body Recommendations, see section 8.1.1.7.

14.4.1.8 Business Stakeholders

Described in sections 3.2.2 and 14.3.3.5.

14.4.2 Tools

14.4.2.1 Communication Plan(s)*

The Communication Plan(s) define how information is to be disseminated to business stakeholders and throughout the programs, portfolio, and organization as a whole. It should also define how and when to communicate and what mode of communication should be used. The portfolio roles provide guidance and input to the Communications Plan for the associated programs within the portfolio. Similarly, the program roles provide guidance and input to the Communications Plan for the projects within the program.

For more information on the Communications Plan, see section 12.1.3.4.

14.4.2.2 Company Human Resource Plan*

Described in section 14.3.2.1.

14.4.2.3 Stakeholder Analysis

Described in section 14.3.2.2.

14.4.2.4 Scrum of Scrums (SoS) Meeting*

The purpose of the Scrum of Scrums (SoS) meeting is similar to its use in large projects. At the program level, representatives from each underlying project in the program meet at regular intervals for Scrum of Scrums (SoS) meetings. For more information on SoS meetings, see section 13.3.5.

14.4.2.5 Scrum of Scrum of Scrums (SoSoS) Meeting

At the program and especially at the portfolio level, it makes sense to have another layer of meetings. Representatives from relevant or interrelated programs and projects in the program or portfolio meet at regular intervals, or as required. In attendance there would be representatives from each of the Scrum of Scrums meetings. This additional level of meetings is called the Scrum of Scrum of Scrums (SoSoS).

Figure 14-8 depicts the structure of a scrum of scrums of scrums (SoSoS) meeting. Illustrates cross-team coordination, issue escalation, and alignment across large-scale product or portfolio initiatives.

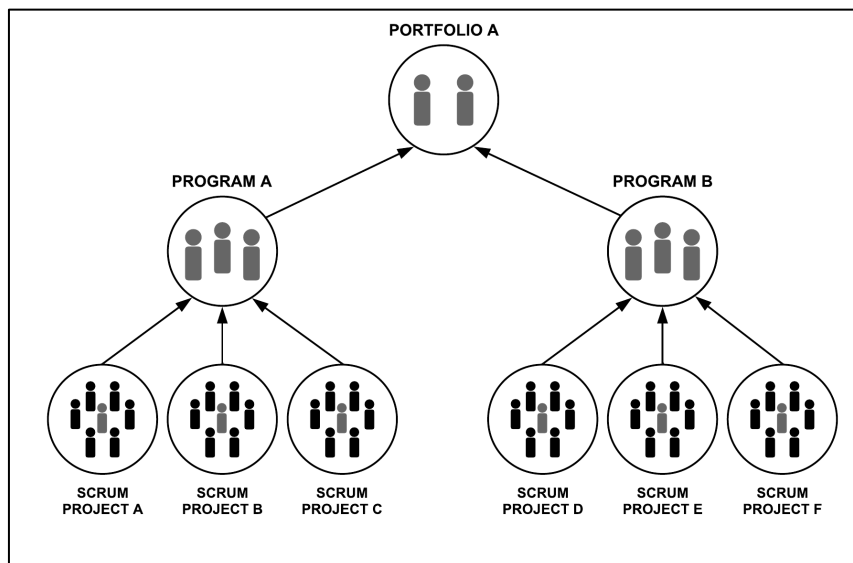


Figure 14-8: Scrum of Scrums of Scrums (SoSoS) Meeting

In this example, there are six Scrum projects going on simultaneously. Scrum projects A, B, and C are part of one program while Scrum projects D, E, and F are part of another program. A Scrum of Scrums meeting is held to coordinate the interdependencies within each of the two programs. A Scrum of Scrums of Scrums meeting may then be conducted to coordinate and manage dependencies across the two programs of the portfolio.

14.4.2.6 Communication Techniques

At the program or portfolio level, communication methods and techniques must scale to a larger number of participants and also due to the fact that not everyone will be in the same workplace. Additional considerations should be made for push versus pull communication types, for example, dashboards or reports published online for business stakeholders to view on demand (pull) or sent directly (push) at regular intervals. Communication across larger groups may utilize more tools, such as web meetings, email, instant messaging, and message boards when face-to-face communication is not possible. Use of an AI-powered Scrum Project Tool may also facilitate communication at a program or portfolio level. For more information on communication techniques, see section 10.3.2.2.

14.4.2.7 AI-powered Scrum Project Tool

At a program or portfolio level, the AI-powered Scrum Project Tool provides the ability to scale in order to manage the additional roles, coordination, reporting, communications, and other relevant requirements. For more information on the AI-powered Scrum Project Tool, see sections 2.4.4 and 13.3.8.

14.4.3 Outputs

14.4.3.1 Minimum Done Criteria*

The Minimum Done Criteria defined at a portfolio level applies to all underlying programs and projects. Similarly, the Minimum Done Criteria at the program level applies to all underlying projects. The cascading set of Done Criteria ensures that all underlying Done Criteria meet the minimum requirements established at the higher levels. The Scrum Guidance Body may be involved with defining the Minimum Done Criteria at a portfolio level. For more information on the Minimum Done Criteria, see section 5.4.4.

14.4.3.2 Shared Resources*

Described in section 13.2.4.

14.4.3.3 Identified Business Stakeholders*

Business stakeholders at the portfolio or program level are an input to this process. Additional business stakeholders are identified in this process. For more information on business holders relevant at the project level, see section 8.2.3.2.

14.4.3.4 Updated Impediment Logs*

Impediments faced by individual projects may be relevant to other projects within the program or portfolio. Therefore, the project-level Impediment Logs may need to be shared with the other projects and/or programs. As a result of the Scrum of Scrums (SoS) or the Scrum of Scrum of Scrums (SoSoS) meetings, there may be a need to update the project-level Impediment Logs. There could also be Impediment Logs at program or portfolio levels. For more information on the Impediment Log, see section 10.1.1.4.

14.4.3.5 Updated Dependencies*

There may be dependencies between inter-related projects and even between programs within the enterprise that need to be identified. Consequently, there should be coordination among the associated projects to manage those dependencies.

Examples of dependencies could include:

- Shared release dates for inter-related projects
- Dependencies between releases
- Dependencies pertaining to inter-related features

As a result of coordinating program or portfolio components, there may be a need to update the known dependencies with new dependencies or changes to existing dependencies. For example, there may be dependencies between projects within a program or portfolio. Looking at two projects A and B in a program, these two projects may need to have the same release date, or project A can be released only after the release of project B. In either case, if project B is delayed, project A will also be delayed, even if its deliverables are ready on time. For more information on dependencies at the project level, see sections 8.5.2.6 and 9.4.2.3.

14.4.3.6 Product Owners Collaboration Plan*

Described in section 13.2.2.

14.4.3.7 Scrum Masters/Scrum Teams Collaboration Plan*

Described in section 13.2.3.

14.4.3.8 Recommended Scrum Guidance Body Improvements

As a result of the *Create/Update Program or Portfolio Components* process, suggestions or feedback may be provided for potential improvements to the Scrum Guidance Body documentation. These recommended improvements will be discussed and agreed to or rejected by the Scrum Guidance Body (see section 14.5, *Review and Update Scrum Guidance Body*). If the suggestions are agreed to, they will be incorporated as updates to the Scrum Guidance Body documentation.

14.5 Review and Update Scrum Guidance Body

In this process, the recommended Scrum Guidance Body improvements are regularly reviewed by the members of the Scrum Guidance Body and are updated when necessary. Changes in the membership of the Scrum Guidance Body are also addressed in this process. The main objective is to constantly monitor and work towards improving the productivity of the Scrum projects, programs, and portfolios within the organization.

It is important to note that the process *Review and Update Scrum Guidance Body* is typically an iterative process, since Scrum projects continuously engage in retrospectives where improvement opportunities are regularly identified and are forwarded up to the program and portfolio levels. At the program or portfolio level, the Scrum Guidance Body reviews inputs from program and portfolio level retrospectives, identifies improvement opportunities, and helps in disseminating good practices across the enterprise.

Figure 14-9 shows all the inputs, tools, and outputs for the *Review and Update Scrum Guidance Body* process.

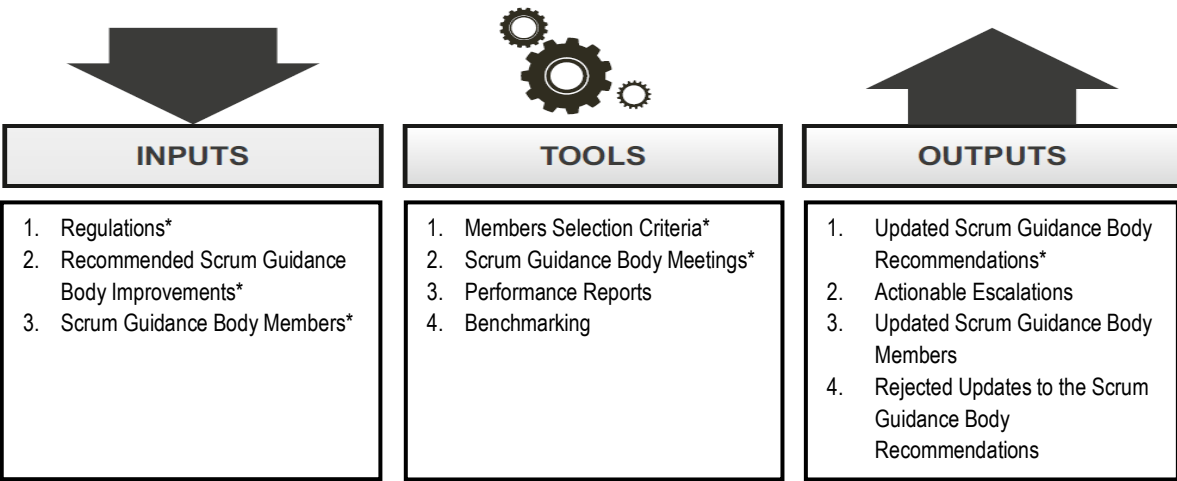


Figure 14-9: Review and Update Scrum Guidance Body—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a "mandatory" input, tool, or output for the corresponding process.

14.5.1 Inputs

14.5.1.1 Regulations*

Regulations include any federal, state, local, or industry regulations that the program or portfolio must adhere to. User Stories created to meet government regulations within a stipulated time period are included in the Portfolio or Program Product Backlog. At times, Scrum Guidance Body recommendations may need to be updated to reflect new regulations.

14.5.1.2 Recommended Scrum Guidance Body Improvements*

As a result of Scrum retrospectives and other processes, suggestions, and feedback to revise or enhance the Scrum Guidance Body guidelines, templates, and other documentation may be made. If the Scrum Guidance Body agrees with any suggestions or feedback, relevant changes will be incorporated as updates to the Scrum Guidance Body material and provided as recommendations to the project, program, and portfolio teams.

14.5.1.3 Scrum Guidance Body Members

The Scrum Guidance Body (SGB) members can include Scrum experts, Scrum coaches, external consultants, selected Scrum Masters, Product Owners, and team members (on all levels). However, there should be a limit on the number of members that the SGB can have to ensure that it remains relevant and does not become prescriptive in nature.

14.5.2 Tools

14.5.2.1 Members Selection Criteria*

Members selection criteria are created to define the Scrum Guidance Body members, their roles and responsibilities, the number of members, and their required skills and expertise. Each organization can have its own selection criteria for Scrum Guidance Body members; however, it is recommended that every member has Scrum expertise and can contribute effectively to the Scrum Guidance Body.

14.5.2.2 Scrum Guidance Body Meetings*

The Scrum Guidance Body meets regularly to discuss the potential need for an update of the Scrum Guidance Body recommendations (e.g., suggested improvements from retrospectives and other processes, updated regulations that need to be incorporated into the documentation, etc.). The Scrum Guidance Body decides the frequency of these meetings based on the specific needs of the enterprise.

14.5.2.3 Performance Reports

There may be reports available about the performance of Scrum projects, programs, and portfolios. Such performance reports might include information related to team velocity, delivered functionality, completion status, and so on. The Scrum Guidance Body can consider this information when determining improvement opportunities.

14.5.2.4 Benchmarking

Benchmarking is the process of comparing an organization's business processes and performance metrics to those of leading companies in the same or other industries. An enterprise should regularly benchmark its own practices against those of successful organizations (in order to keep up with the competition) and against current and upcoming industry standards and practices.

14.5.3 Outputs

14.5.3.1 Updated Scrum Guidance Body Recommendations*

After reviewing and considering the Scrum Guidance Body improvement suggestions, performance reports, and benchmarking data, changes may be necessary to the existing documentation. Any approved changes will lead to an update of the Scrum Guidance Body material and will be provided as recommendations for the current or future Scrum projects, programs, and portfolios.

14.5.3.2 Actionable Escalations

The Scrum Guidance Body may determine that some company policies do not allow teams to obtain the maximum benefits from the application of Scrum. In such cases, an escalation should be triggered in order to gain approval for a policy or other change.

14.5.3.3 Updated Scrum Guidance Body Membership

As a result of assessing the Scrum Guidance Body membership, new members may be added, and existing members may leave the Scrum Guidance Body.

14.5.3.4 Rejected Updates to the Scrum Guidance Body Recommendations

Recommended Scrum Guidance Body improvements may not always be accepted. If a recommended improvement is rejected by the Scrum Guidance Body members, an explanation of the reason(s) for the rejection is provided as feedback to the relevant parties.

14.6 Create/Refine Prioritized Program or Portfolio Backlog

In this process, the Program or Portfolio Backlog is first created based on the program or portfolio requirements. On an ongoing basis, the Prioritized Program or Portfolio Backlog is updated and maintained with new or updated requirements, risks, and priorities.

Any outputs from the *Create/Refine Prioritized Program or Portfolio Backlog* process become inputs to the process after their initial creation. For example, the Prioritized Program or Portfolio Backlog is created for the first time during this process, but it becomes a mandatory input to subsequent refining/updates.

Figure 14-10 shows all the inputs, tools, and outputs for the *Create/Refine Prioritized Program or Portfolio Backlog* process.

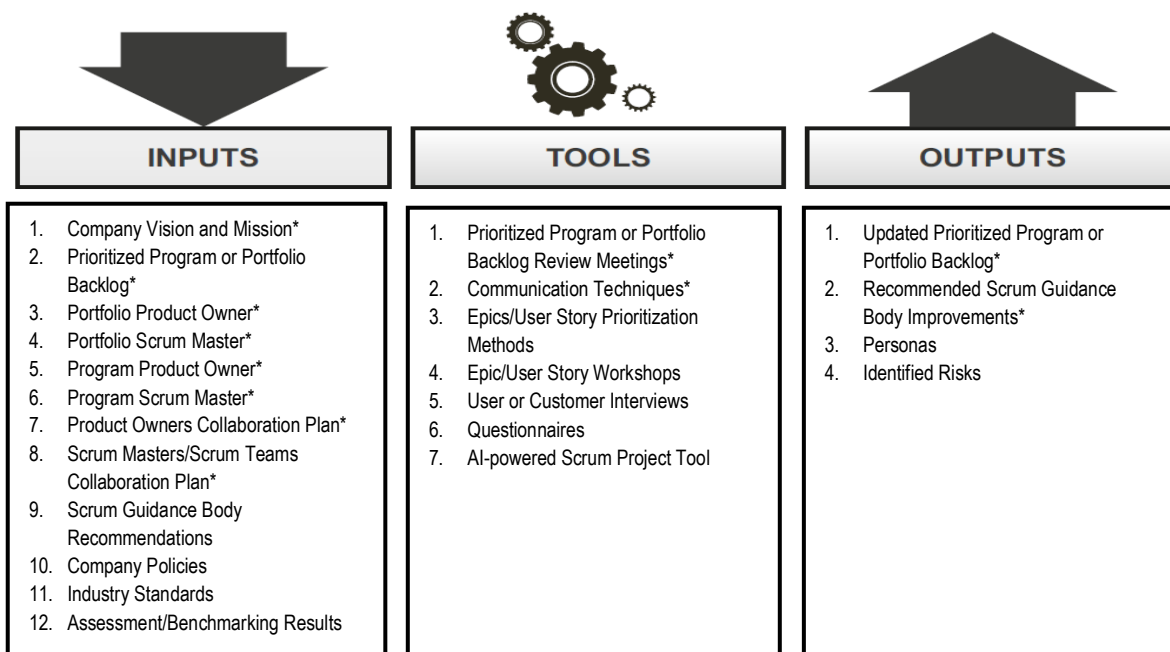


Figure 14-10: Create/Refine Prioritized Program or Portfolio Backlog—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

14.6.1 Inputs

14.6.1.1 Company Vision and Mission*

Described in section 14.3.1.1.

14.6.1.2 Prioritized Program or Portfolio Backlog*

The Program or Portfolio Product Owner develops the Prioritized Program or Portfolio Backlog, which contains a prioritized list of high-level business and project requirements, preferably written in the form of large Epics at a program or portfolio level. The Product Owners of individual projects later refine these as they create and prioritize the Product Backlogs for their own projects. These Prioritized Product Backlogs have much smaller but more detailed User Stories that can be estimated and committed by individual Scrum Teams. These details are shared between different projects to avoid unnecessary duplication of work effort.

The Prioritized Program Backlog plays a similar role at the program level as the Prioritized Product Backlog does at the project level. It captures the requirements for the program and their priorities. There are a few differences, however. The creation of the respective deliverables and their acceptance are handled within the projects of the program. Done or Acceptance Criteria for each Product Backlog item/User Story may be defined at the program level. The teams must adhere to these criteria, but new criteria can be added, as required.

The length of a Sprint is project specific and can therefore vary from project to project within a program. In addition, velocity varies from team to team. Therefore, it is not necessary to have very granular User Stories at the program level. Typically, programs have high-level requirements in the form of Epics, and the refinement of Epics at the program level goes only far enough to ensure that each respective Epic is clearly understood and that tangible Acceptance Criteria for the program can be defined.

The Prioritized Portfolio Backlog plays the same role at the portfolio level as the Prioritized Program Backlog does at the program level. The items in the Prioritized Portfolio Backlog provide inputs to the various Prioritized Program Backlogs and to the Prioritized Product Backlogs of each of the corresponding projects. As with the Prioritized Program Backlog, only minimal, if any, refinement of Epics is done at this level, because the refinement is handled within the associated projects at the level of each of the Prioritized Product Backlogs.

The Program or Portfolio Product Owner continuously refine the Prioritized Program or Portfolio Backlog to ensure that new business requirements are added, and existing requirements are properly documented and prioritized. This ensures that the most valuable requirements in meeting the portfolio or program's objectives are prioritized as high and the remaining are given a lower priority.

The Prioritized Program or Portfolio Backlog presents a larger picture of all projects that are part of the program or portfolio. Therefore, it can provide significant guidance regarding project goals, scope, objectives, and the expected business benefits.

14.6.1.3 Portfolio Product Owner*

The Portfolio Product Owner is responsible for the creation and the refining of the Prioritized Portfolio Backlog. For more information on the role of the Portfolio Product Owner, see section 3.7.4.2.

14.6.1.4 Portfolio Scrum Master*

At the portfolio level, the Portfolio Scrum Master plays a similar role as the Program Scrum Master plays for a program. He or she is a facilitator, solves problems, and removes impediments at the portfolio level. For more information on the role of the Portfolio Scrum Master, see section 3.7.4.4.

14.6.1.5 Program Product Owner*

At the program level, the Program Product Owner is responsible for and is the driver of the creation and the refining of the Prioritized Program Product Backlog. For more information on the role of the Program Product Owner, see section 3.7.4.1.

14.6.1.6 Program Scrum Master*

At the program level, the Program Scrum Master plays a role similar to that of the Scrum Master in a project. He or she is a facilitator, solves problems, and removes impediments at the program level. For more information on the role of the Program Scrum Master, see section 3.7.4.3.

14.6.1.7 Product Owners Collaboration Plan

Described in section 13.2.2.

14.6.1.8 Scrum Masters/Scrum Teams Collaboration Plan

Described in section 13.2.3.

14.6.1.9 Scrum Guidance Body Recommendations

When creating and refining the Prioritized Program or Portfolio Backlog, Scrum Guidance Body recommendations provide best practices that should be taken into consideration at the program or portfolio level. For more information on Scrum Guidance Body Recommendations, see sections 8.1.1.7 and 10.3.1.11.

14.6.1.10 Company Policies

Company policies are a set of principles, rules, and guidelines formulated or adopted by an organization. Changing company policies could affect existing Epics or User Stories that were created based on existing policies.

14.6.1.11 Industry Standards

New industry standards or changes to existing standards need to be implemented in order to maintain a viable product or service. Therefore, User Stories related to meeting these standards need to be included in the Prioritized Program or Portfolio Backlog and prioritized accordingly. Sometimes, the Scrum Guidance Body recommendations may also need to be changed to reflect new or changed industry standards.

14.6.1.12 Assessment/Benchmarking Results

Primarily, assessment or benchmarking results will necessitate an update to the Scrum Guidance Body recommendations for best practices. The results can also help set a minimum standard when creating a product or service and may lead to changes to the Done Criteria. Sometimes new assessments or benchmarking results may also provide impetus for a Program or Portfolio Product Owner to develop new Epics to implement any additional or updated best practices.

14.6.2 Tools

14.6.2.1 Prioritized Program or Portfolio Backlog Review Meetings*

Participation in the Program or Portfolio Backlog review meetings is quite different from participation in the Product Backlog review meetings at the project level. Scrum Teams participate in the refining sessions at the project level. At the program or portfolio level, there is representation from each project within the program or from each program and/or standalone projects within the portfolio. However, to streamline the meetings, it is recommended to have only one or a few representatives from each project or program attend at the program or portfolio level. Refer to related sections 6.5.1.2 and 10.3.2.1 for more information.

14.6.2.2 Communication Techniques*

Described in section 10.3.2.2.

14.6.2.3 Epics/User Story Prioritization Methods

At the program or portfolio level, there is normally a smaller number of requirements/Epics/User Stories than at the project level. Also, these requirements will be at a very high level and prioritization will be driven primarily by the business requirements (as determined by the business stakeholders), the Portfolio Product Owner, and the Program Product Owner. For more information on prioritization methods, see section 8.5.2.1.

14.6.2.4 Epic/User Story Workshops

Compared to projects, User Story Workshops for programs and portfolios aim to produce only higher-level Epics/User Stories as their outputs, so there will be significantly fewer Epics/User Stories at this point. However, the meetings still provide value as they are attended by representatives from the projects within the program or from the programs within the portfolio and these individuals can carry relevant information back to their respective teams. The Program or Portfolio Scrum Master typically coordinate these workshops. This ensures that requirements are well defined and understood throughout the program or portfolio. For more information on User Story Workshops, see section 8.4.2.2.

14.6.2.5 User or Customer Interviews

Described in section 8.4.2.4.

14.6.2.6 Questionnaires

Described in section 8.4.2.5.

14.6.2.7 AI-powered Scrum Project Tool

An appropriately designed AI-powered Scrum Project Tool provides an easy-to-understand view of the Prioritized Program or Portfolio Backlog and also helps the Program or Portfolio Product Owner to view and prioritize the requirements/Epics/User Stories. For more information on the AI-powered Scrum Project Tool, see sections 2.4.4 and 13.3.8.

14.6.3 Outputs

14.6.3.1 Updated Prioritized Program or Portfolio Backlog*

The Prioritized Program or Portfolio Backlog may be updated with new or updated Epics/User Stories; work related to new Change Requests or identified risks; and/or to reflect the reprioritization of existing Epics/User Stories. Refining the Program or Portfolio Backlog might result in a decision to initiate a new project (for example, to create a framework or a common web-interface to be used by all projects).

14.6.3.2 Recommended Scrum Guidance Body Improvements*

As a result of the *Create/Refine Program or Portfolio Backlog* process, suggestions or feedback might be provided for potential improvements to the Scrum Guidance Body documentation. These recommended improvements will be discussed and agreed to or rejected by the Scrum Guidance Body. If any of the suggestions are accepted, they will be incorporated as updates to the Scrum Guidance Body documentation. For more information, see the *Review and Update Scrum Guidance Body* process.

14.6.3.3 Personas

Described in section 8.4.3.2.

14.6.3.4 Identified Risks

Risks related to a program or portfolio will also impact the projects that are a part of the respective program or portfolio. During risk assessment of the program or portfolio, if it is determined that a risk may affect a lower-level program or project, relevant information about that risk must be communicated to the respective Product Owner and Scrum Team. Program and portfolio risks become inputs to the *Develop Epics* process for the relevant project(s) and can have an overall impact on how this process is conducted. For more information on identifying project risks, see sections 7.4.1 and 8.4.3.4. Program and portfolio risks are discussed in section 7.6.

14.7 Create/Update Program or Portfolio Releases

In this process, the program or portfolio releases are planned, considering any dependencies between the releases. Program or portfolio release planning will impact release planning at the project level. The Program or Portfolio Release Schedule is created and should be revisited regularly based on the progress of project deliverables, new or changed requirements or their priorities, and other factors.

The Release Planning Meeting is used to review the existing releases and plan for new releases.

Outputs from the *Create/Update Program or Portfolio Releases* process become inputs to the process after their initial creation. For example, initial program or portfolio releases may be created for the first time during this process, but these become mandatory inputs for future program or portfolio releases.

Figure 14-11 shows all the inputs, tools, and outputs for the *Create/Update Program or Portfolio Releases* process.

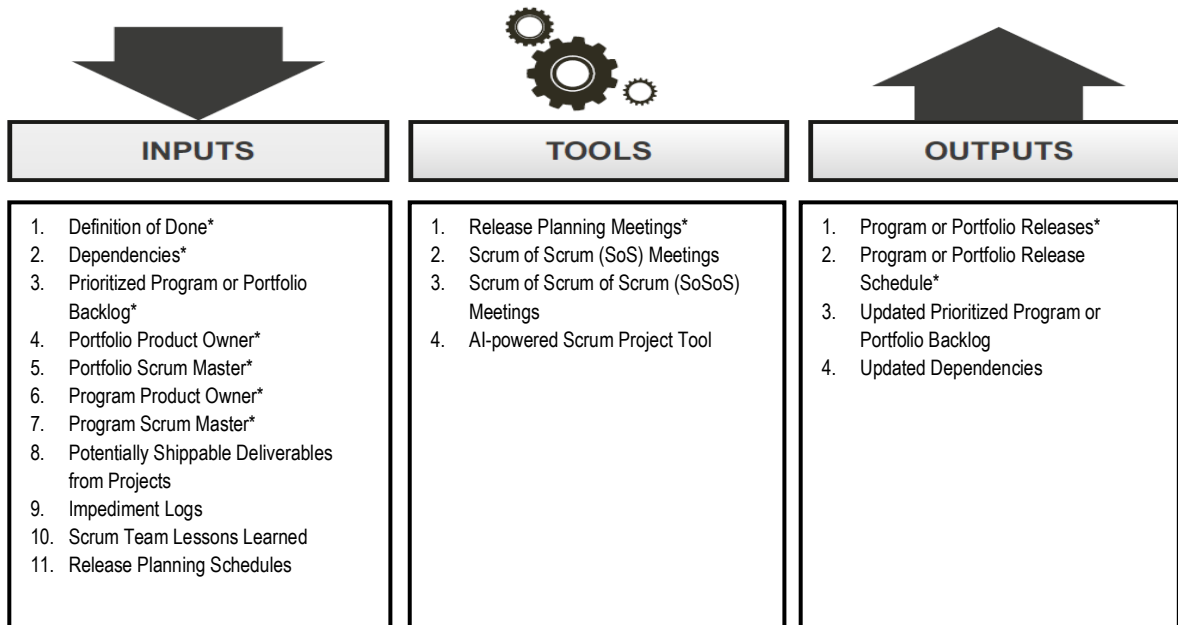


Figure 14-11: Create/Update Program or Portfolio Releases—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

14.7.1 Inputs

14.7.1.1 Definition of Done*

The Definition of Done (or Done Criteria) defined at the program or portfolio level can be used as the minimum Done Criteria for projects across the enterprise. For more information on the Done Criteria, see sections 5.4.3 and 8.5.3.2.

14.7.1.2 Dependencies*

Described in sections 8.5.2.6 and 8.5.3.5.

14.7.1.3 Prioritized Program or Portfolio Backlog*

Described in section 14.6.1.2.

14.7.1.4 Portfolio Product Owner*

Described in section 3.7.4.2.

14.7.1.5 Portfolio Scrum Master*

Described in section 3.7.4.4.

14.7.1.6 Program Product Owner*

Described in section 3.7.4.1.

14.7.1.7 Program Scrum Master*

Described in section 3.7.4.3.

14.7.1.8 Potentially Shippable Deliverables from Projects

Potentially shippable deliverables from projects are valuable inputs for coordination at the program or portfolio level. At the end of each Sprint in a project, product increments or deliverables are completed. The User Stories included in these increments meet the Done Criteria as well as their respective Acceptance Criteria.

14.7.1.9 Impediment Logs

Described in sections 10.1.1.4 and 14.4.3.4.

14.7.1.10 Scrum Teams Lessons Learned

Described in section 11.2.3.5.

14.7.1.11 Release Planning Schedules

These schedules, while tentative and subject to change, are vital to gauge whether or not the respective projects are likely to meet their required deadlines and are especially crucial with respect to dependencies. For more information on the Release Planning Schedule at the project level, see section 8.6.3.1.

14.7.2 Tools

14.7.2.1 Release Planning Meetings*

Release Planning Meetings take place between the Portfolio Product Owner, Program Product Owner, Portfolio Scrum Master, Program Scrum Master, and other relevant business stakeholders from the business and project teams to ensure that all program and portfolio releases are planned properly. The program and portfolio releases in turn provide valuable inputs to plan releases at the project level.

14.7.2.2 Scrum of Scrum (SoS) Meetings*

Described in section 13.3.5.

14.7.2.3 Scrum of Scrum of Scrum (SoSoS) Meetings*

Described in section 14.4.2.5.

14.7.2.4 AI-powered Scrum Project Tool

The AI-powered Scrum Project Tool helps the teams to easily view planned program or portfolio releases, make appropriate changes if required, and also plan additional releases. For more information on the AI-powered Scrum Project Tool, see sections 2.4.4 and 13.3.8.

14.7.3 Outputs

14.7.3.1 Program or Portfolio Releases*

A program or portfolio release includes the releases for all the underlying projects in the program or portfolio. There are typically two ways in which program or portfolio releases happen:

1. All deliverables for the underlying projects are completed but kept ready to be released as determined by the program or portfolio. There may be a specific release date on which the whole program or portfolio is released to the end customer.
2. Deliverables are released at a project level whenever they are ready, for example, in DevOps, which allows for continuous development, implementation, and deployment.

14.7.3.2 Program or Portfolio Release Schedule*

A Program or Portfolio Release schedule includes target dates for different releases planned at the program or portfolio level.

14.7.3.3 Updated Prioritized Program or Portfolio Backlog

The Prioritized Program or Portfolio Backlog is updated as existing releases are reviewed and proposed changes or new releases are added. Updating of Program or Portfolio releases may also impact the prioritization of Epics/User Stories in the Prioritized Program or Portfolio Backlog. For more information on the Prioritized Program or Portfolio Backlog, see section 14.6.1.2.

14.7.3.4 Updated Dependencies

Dependencies may be updated based on discussions in the Release Planning Meetings or sessions.

14.8 Retrospect Program or Portfolio Releases

In this process, the Program or Portfolio Product Owner and business stakeholders get together to retrospect a program or portfolio release and to also discuss and internalize the lessons learned. Often, these lessons learned lead to Agreed Actionable Improvements to be implemented in future releases. Sometimes, improvements to the Scrum Guidance Body may be recommended. These meetings may be scheduled after each program or portfolio release.

Figure 14-12 outlines “retrospect program or portfolio releases.” includes inputs from scrum teams and stakeholders, tools like retrospective meetings, and outputs such as improvement actions and guidance body updates.

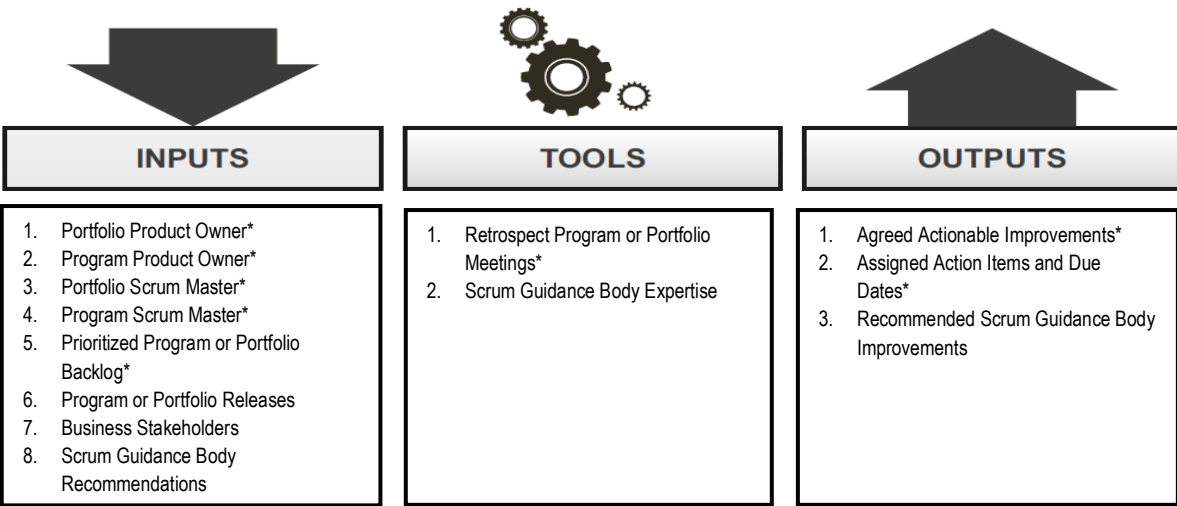


Figure 14-12: Retrospect Program or Portfolio Releases—Inputs, Tools, and Outputs

Note: Asterisks (*) denote a “mandatory” input, tool, or output for the corresponding process.

14.8.1 Inputs

14.8.1.1 Portfolio Product Owner*

Described in section 3.7.4.2.

14.8.1.2 Portfolio Scrum Master*

Described in section 3.7.4.4.

14.8.1.3 Program Product Owner*

Described in section 3.7.4.1.

14.8.1.4 Program Scrum Master*

Described in section 3.7.4.3.

14.8.1.5 Prioritized Program or Portfolio Backlog*

Described in section 14.6.1.2.

14.8.1.6 Program or Portfolio Releases*

Described in section 14.7.3.1.

14.8.1.7 Business Stakeholders

Described in sections 3.2.2 and 14.3.3.5.

14.8.1.8 Scrum Guidance Body Recommendations

During a retrospective of Program or Portfolio releases, the Scrum Guidance Body recommendations provide pertinent best practices including information on administrative procedures, audits, evaluations, and project transition criteria. This is similar to the role the Scrum Guidance Body Recommendations plays at the project level retrospectives (described in section 12.2.1.5).

14.8.2 Tools

14.8.2.1 Retrospect Program or Portfolio Meetings*

The Retrospect Program or Portfolio Meeting is similar to the Retrospect Release Meeting held at the project level (see section 12.2.2.1). The major difference is that the Retrospect Program and Portfolio Meetings are held much less frequently than the Retrospect Release Meetings and include the Program/Portfolio Product Owner, the Program/Portfolio Scrum Master, and the business stakeholders for the program or portfolio.

14.8.2.2 Scrum Guidance Body Expertise

Described in section 8.4.2.7.

14.8.3 Outputs

14.8.3.1 Agreed Actionable Improvements*

Described in section 11.2.3.1.

14.8.3.2 Assigned Action Items and Due Dates*

Described in section 11.2.3.2

14.8.3.3 Recommended Scrum Guidance Body Improvements

As a result of the *Retrospect Program or Portfolio Releases* process, suggestions or feedback may be provided for potential improvements to the Scrum Guidance Body documentation. These recommended improvements will be discussed and agreed to or rejected by the Scrum Guidance Body. If any of the suggestions are accepted, they will be incorporated as updates to the Scrum Guidance Body documentation. For more information, see the *Review and Update Scrum Guidance Body* process.

APPENDIX A. OVERVIEW OF AGILE

A.1 Introduction

This appendix intends to familiarize readers with the concept of Agile development and the various Agile methodologies.

The following sections are included:

A.2 Overview—This section discusses the definition of and the factors behind the rise of Agile.

A.3 Agile Manifesto—This section presents *The Agile Manifesto*, its principles, and *The Declaration of Interdependence* to provide the historical context of Agile.

A.4 Agile Methods—This section provides a brief overview of specific Agile methodologies including:

- Lean Kanban
- Extreme Programming
- Crystal Methods
- Dynamic Systems Development Methods
- Feature Driven Development
- Test Driven Development
- Adaptive Software Development
- Agile Unified Process
- Domain-Driven Design

A.2 Overview

The term “Agile” refers to being able to move or respond quickly and easily; being nimble. In any kind of management discipline, Agile as a quality would therefore be a valid aim. Agile project management specifically involves being adaptive during the creation of a product, service, or other deliverable.

It is important to understand that while Agile development methods are highly adaptive, it is also necessary to incorporate stability in their adaptive processes.

A.2.1 The Rise of Agile

Rapid changes in technology, market demands, and expectations have resulted in increased challenges to developing products and services using traditional project management models. This paved the way for the conceptualization and implementation of Agile methods and values in many organizations. Agile development models addressed the shortcomings associated with traditional project management models in meeting the ever-growing environmental demands and expectations that organizations were facing. Since traditional project management models emphasize extensive upfront planning and conforming to the plan once it is baselined, such models were not successful in meeting the reality of frequent environmental changes.

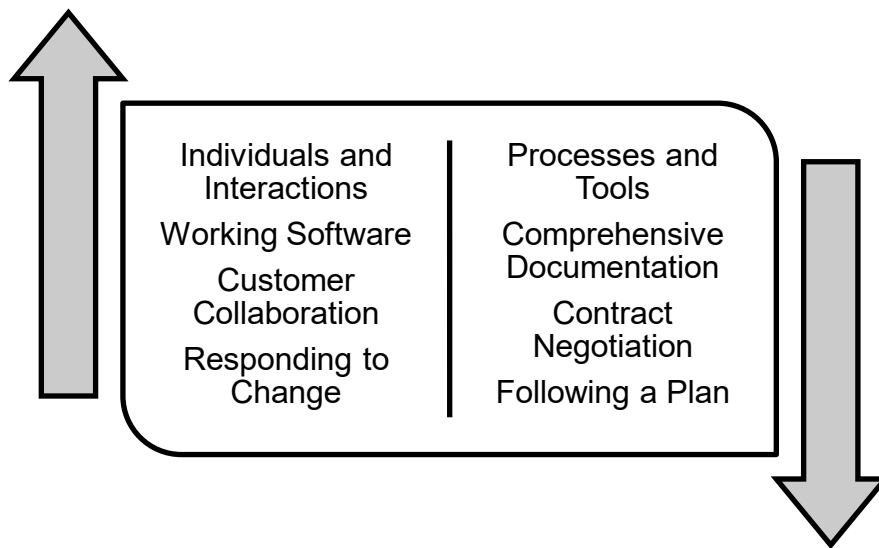
Agile relies on adaptive planning and iterative development and delivery. It focuses primarily on the value of people in getting the job done effectively. Though adaptive and incremental methodologies have existed since the 1950s, only methodologies that conform to *The Agile Manifesto* are regarded as truly “Agile”.

A.3 The Agile Manifesto

In February 2001, a group of seventeen computer gurus, software developers, and managers held a retreat to discuss lightweight software development methods. They formed the *Agile Alliance* and the discussions at those meetings later resulted in a *Manifesto for Agile Software Development*. The Manifesto was authored by Fowler and Highsmith (2001) and then signed by all participants to establish the basic guidelines for any Agile method.

The purpose of *The Agile Manifesto* was laid out as follows:

*We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:*



*That is, while there is value in the items on
the right, we value the items on the left more.*

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Permission to copy provided by the above authors by notice on <http://agilemanifesto.org/>.

The four trade-offs emphasized by The Agile Manifesto are elaborated as follows:

1. Individuals and interactions over processes and tools

Although processes and tools help in successfully completing a project, it is the individuals who undertake, participate in, implement a project, and determine which processes and tools to use. The key actors in any project are therefore individuals, so the emphasis should be on them and their interactions, rather than complicated processes and tools.

2. Working software over comprehensive documentation

While documentation is necessary and useful for any project, many teams focus on collecting and recording qualitative and quantitative descriptions of deliverables, when the real value delivered to the customer is primarily in the form of working software. Therefore, the Agile focus is on delivering working software in increments throughout the product lifecycle rather than detailed documentation.

3. Customer collaboration over contract negotiation

Traditionally, customers have been seen as outside players who are involved at the start and end of the product lifecycle and whose relationships were based on contracts and their fulfillment. Agile believes in a shared value approach in which customers are seen as collaborators. The development team and customers work together to evolve and develop the product.

4. Responding to change over following a plan

In the current market in which customer requirements, available technologies, and business patterns are constantly changing, it is essential to approach product development in an adaptive manner that enables change incorporation and fast product development lifecycles rather than emphasizing following plans formed with potentially outdated data.

A.3.1 Principles of the Agile Manifesto

The twelve principles of the Agile Manifesto by Fowler and Highsmith (2001) are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals, give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architecture, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

A.3.2 Declaration of Interdependence

The Agile project management *Declaration of Interdependence* was written in early 2005 by a group of fifteen project leaders as a supplement to *The Agile Manifesto*. It enumerates six management values needed to reinforce an Agile development mentality, particularly when managing complex, uncertain projects.

The declaration highlights that project teams, customers, and other business stakeholders are interdependent and connected and must recognize this to be successful. The values themselves are also interdependent.

We ...

increase return on investment by making continuous flow of value our focus.

deliver reliable results by engaging customers in frequent interactions and shared ownership.

expect uncertainty and manage for it through iterations, anticipation, and adaptation.

unleash creativity and innovation by recognizing that individuals are the ultimate source of value and creating an environment where they can make a difference.

boost performance through group accountability for results and shared responsibility for team effectiveness.

improve effectiveness and reliability through situationally specific strategies, processes, and practices.

Anderson, D., Augustine, S., Avery, C., Cockburn, A., Cohn, M., et al. 2005

A.4 Agile Methods

A number of Agile methodologies originated and gained traction in the 1990's and the early 2000's. While they differ in a variety of aspects, their commonality stems from their adherence to *The Agile Manifesto*.

The following Agile methods are briefly discussed below:

1. Lean Kanban
2. Extreme Programming (XP)
3. Crystal Methods
4. Dynamic Systems Development Methods (DSDM)
5. Feature Driven Development (FDD)
6. Test Driven Development (TDD)
7. Adaptive Software Development (ASD)
8. Agile Unified Process (AUP)
9. Domain-Driven Design (DDD)

A.4.1 Lean Kanban

The Lean concept optimizes an organization's system to produce valuable results based on its resources, needs, and alternatives while reducing waste. Waste could be from building the wrong thing, failure to learn, or practices that impede the process. Because these factors are dynamic in nature, a lean organization evaluates its entire system and continuously fine tunes its processes. The foundation of Lean is that the reduction of the length of each cycle (i.e., an iteration) leads to an increase in productivity by reducing delays, aids in error detection at an early stage, and consequently reduces the total amount of effort required to finish a task. Lean software principles have been successfully applied to software development.

Kanban means a “signboard” or “billboard” and it espouses the use of visual aids to assist and track production. The concept was introduced by Taiichi Ohno considered to be the father of the Toyota Production Systems (TPS). The use of visual aids is effective and has become common practice. Examples include task cards, Scrumboards, and Burndown Charts. These methods gained attention due to their practice at Toyota, a leader in process management. Lean Kanban integrates the use of the visualization methods as prescribed by Kanban along with the principles of Lean creating a visual incremental evolutionary process management system.

A.4.2 Extreme Programming

Extreme Programming (XP), which originated in Chrysler Corporation, gained traction in the 1990's. XP makes it possible to keep the cost of changing software from rising radically over time. The key attributes of XP include incremental development, flexible scheduling, automated test codes, verbal communication, ever-evolving design, close collaboration, and tying in the long- and short-term drives of all those involved.

XP values communication, feedback, simplicity, and courage. The different roles in the XP approach include customer, developer, tracker, and coach. It prescribes various coding, developer, and business practices as well as events and artifacts to achieve effective and efficient development. XP has been extensively adopted due to its well-defined engineering practices.

A.4.3 Crystal Methods

Alistair Cockburn introduced the Crystal methodologies of software development in the early 1990s. Crystal methods are intended to be people-centric, lightweight, and easy to adapt. Because people are primary, the developmental processes and tools are not fixed but are adjusted to the specific requirements and characteristics of the project. The color spectrum is used to decide on the variant for a project. Factors such as comfort, discretionary money, essential money, and life play a vital role in determining the “weight” of the methodology, which is represented in various colors of the spectrum. The Crystal family is divided into Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Diamond, and Crystal Sapphire.

All Crystal methods have four roles—executive sponsor, lead designer, developers, and experienced users. Crystal Methods recommend various strategies and techniques to achieve agility. A Crystal project cycle consists of chartering, delivery cycle, and wrap-up.

A.4.4 Dynamic Systems Development Methods (DSDM)

The Dynamic Systems Development Methods (DSDM) framework was initially published in 1995 and is administered by the DSDM Consortium. DSDM sets quality and effort in terms of cost and time at the outset and adjusts the project deliverables to meet set criteria by prioritizing the deliverables into “Must have,” “Should have,” “Could have,” and “Won’t have” categories (using the MoSCoW prioritization technique). DSDM is a system-oriented method with six distinct phases—Pre-project; Feasibility; Foundations; Exploration and Engineering; Deployment; and Benefit Assessment.

A.4.5 Feature Driven Development (FDD)

Feature Driven Development (FDD) was introduced by Jeff De Luca in 1997 and operates on the principle of completing a project by breaking it down into small, client-valued functions that can be delivered in less than two weeks’ time. FDD has two core principles—software development is a human activity and software development is a client-valued functionality.

FDD defines six major roles—Project Manager, Chief Architect, Development Manager, Chief Programmers, Class Owners, and Domain Experts with a number of supporting roles. The FDD process is iterative and consists of developing an overall model, building a feature list, and then planning, designing, and building by feature.

A.4.6 Test Driven Development (TDD)

Sometimes known as Test-First Development, Test Driven Development was introduced by Kent Beck, one of the creators of Extreme Programming (XP). Test Driven Development is a software development method that involves writing automated test code first and developing the least amount of code necessary to pass that test later. The entire project is broken down into small, client-valued features that need to be developed in the shortest possible development cycle. Based on clients' requirements and specifications, tests are written. The tests designed in the above stage are used to design and write the production code.

TDD can be categorized into two levels: Acceptance TDD (ATDD) requiring a distinct acceptance test and Developer TDD (DTDD) involving writing a single developer test. TDD has become popular because of the numerous advantages it offers like rapid and reliable results, constant feedback, and reduced debugging time.

A.4.7 Adaptive Software Development (ASD)

Adaptive Software Development (ASD) grew out of the rapid application development work by Jim Highsmith and Sam Bayer. The highlights of ASD are constant adaptation of processes to the work at hand, provision of solutions to problems surfacing in large projects, and iterative, incremental development with continuous prototyping.

Being a risk-driven and a change-tolerant development approach, ASD believes a plan cannot admit uncertainties and risks as this indicates a flawed and failed plan. ASD is feature-based and target-driven. The first phase of development in ASD is Speculate (as opposed to Planning) followed by the Collaborate and Learn phases.

A.4.8 Agile Unified Process (AUP)

Agile Unified Process (AUP) evolved from IBM's Rational Unified Process. Developed by Scott Ambler, AUP combines industry-tried-and-tested Agile techniques such as Test-Driven Development (TDD), Agile Modeling, Agile change management, and database refactoring, to deliver a working product of the best quality.

AUP models its processes and techniques on the values of Simplicity, Agility, Customizability, Self-organization, Independence of tools, and focus on high-value activities. The AUP principles and values are put into action in the phases of Inception, Elaboration, Construction, and Transition.

A.4.9 Domain-Driven Design (DDD)

Domain-driven design is an Agile development approach meant for handling complex designs with implementation linked to an evolving model. It was conceptualized by Eric Evans in 2004 and revolves around the design of a core domain. “Domain” is defined as an area of activity to which the user applies a program or functionality.

Many such areas are batched, and a model is designed. The model consists of a system of abstractions that can be used to design the overall project and solve the problems related to the batched domains. The core values of DDD include domain-oriented, model-driven design, ubiquitous language, and a bounded context.

In DDD, ubiquitous language is established, and the domain is modeled. Then design, development, and testing follow. Refining and refactoring of the domain model is done until it is satisfactory.

APPENDIX B. FIFTH EDITION UPDATES

B.1 Summary of Changes

The changes in SBOK 5th edition compared to 4th edition are relatively minor. The scope of updates made to the SBOK® Guide—Fifth Edition includes the following:

- Addition of a new tool, 'AI-powered Scrum Tool,' to most process ITOs, enabling students to learn how AI and Scrum tools can assist Scrum practitioners.
- Screenshots have been included to show how Scrum is implemented in several popular AI-enabled online Scrum tools such as Vabro.ai, Jira, ClickUp, Monday, Wrike, Asana, etc., helping participants understand Scrum in real-life scenarios.

General improvements were also made throughout the text to ensure information was accurate, clear, and complete. This included updates to tables and figures as appropriate.

This Page is Intentionally Left Blank.

REFERENCES

- Anderson, D., Augustine, S., Avery, C., Cockburn, A., Cohn, M., DeCarlo, D., Fitzgerald, D., Highsmith, J., Jepsen, O., Lindstrom, L., Little, T., McDonald, K., Pixton, P., Smith, P., and Wysocki, R. (2005) "Declaration of Interdependence," accessed September 2013, <http://www.pmdoi.org/>.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001) "Manifesto for Agile Software Development," accessed September 2013, <http://agilemanifesto.org/>.
- Fellers, G. (1994) *Why Things Go Wrong: Deming Philosophy In A Dozen Ten-Minute Sessions*. Gretna, LA: Pelican Publishing.
- Kano, N., Seraku, N., Takahashi, F., and Tsuji, S. (1984) "Attractive Quality and Must Be Quality." *Quality*, 14 (2): 39–48.
- Leffingwell, D. and Widrig, D. (2003) *Managing Software Requirements: A Use Case Approach, 2nd ed.* Boston: Addison-Wesley.
- Maslow, A. H. (1943) "A Theory of Human Motivation." *Psychological Review*, 50 (4): 370–396.
- McGregor, D. (1960) *The Human Side of Enterprise*. New York: McGraw-Hill.
- Ouchi, G.W (1982) Theory Z: How American Business Can Meet the Japanese Challenge.
- Patton, J. (2005) "It's All in How You Slice." *Better Software*, January: 16–40.
- Spears, L. C. (2010) "Character and Servant Leadership: Ten Characteristics of Effective, Caring Leaders." *The Journal of Virtues & Leadership*, 1 (1): 25–30.
- Takeuchi, H. and Nonaka, I. (1986) "The New New Product Development Game." *Harvard Business Review*, January–February: 137–146.

This Page is Intentionally Left Blank.

GLOSSARY

100-Point Method

The 100-Point Method was developed by Dean Leffingwell and Don Widrig (2003). It involves giving the customer 100 points they can use to vote for the features that they feel are most important.

Accepted Deliverables

Deliverables which meet the User Story Acceptance Criteria are accepted by the Product Owner. These are considered Accepted Deliverables that may be released to the customer if they so desire.

Actionable Escalations

The Scrum Guidance Body may determine that some company policies do not allow teams to get the maximum benefits from the application of Scrum. In such a case, an escalation needs to be triggered in order to get approval for a policy change.

Adaptation

Adaptation happens as the Scrum Core Team and business stakeholders learn through transparency and inspection and then adapt by making improvements in the work they do.

Affinity Estimation

Affinity Estimation is a technique used to quickly estimate a large number of User Stories using categories. The categories can be small, medium, or large, or they may be numbered using story point values to indicate relative size. Some key benefits of this approach are that the process is very transparent, visible to everyone, and is easy to conduct.

AI-powered Scrum Project Tools

Software platforms enhanced with artificial intelligence that help automate Scrum practices like backlog refinement, progress tracking, and sprint planning, providing real-time insights and predictive analytics.

Agreed Actionable Improvements

Agreed actionable improvements are the primary output of the *Retrospect Sprint* process. They are the list of actionable items that the team has come up with to address problems and improve processes in order to enhance their performance in future Sprints.

Approved Change Requests

Approved Change Requests are changes that have been approved to be included in the Prioritized Product Backlog. At times, Approved Change Requests may originate from the program or portfolio managers and would be inputs to be added to the list of approved project changes for implementation in future Sprints.

Assertive Leader

Assertive leaders confront issues and display confidence to establish authority with respect.

Assessments/Benchmarking Results

Assessment/benchmarking helps set a minimum standard when creating a product or service and leads to changed Done Criteria. Sometimes they may also provide impetus for a Program or Portfolio Product Owner to develop new User Stories to implement the best practices.

Assigned Action Items and Due Dates

Once the agreed actionable improvements have been elaborated and refined, action items to implement the improvements may be considered by the Scrum Team. Each action item will have a defined due date for completion.

Autocratic Leader

Autocratic leaders make decisions on their own, allowing team members little, if any, involvement, or discussion before a decision is made. This leadership style should only be used on rare occasions.

Automated Software Tools

Automated Software Tools are software tools used for scheduling, information collection, and distribution.

Better Team Coordination

The Scrum of Scrums Meeting facilitates coordination of work across multiple Scrum Teams. This is especially important when there are tasks involving inter-team dependencies. Incompatibility and discrepancies between the work and deliverables of different teams are quickly exposed. This forum also gives teams the opportunity to showcase their achievements and give feedback to other teams.

Benchmarking

An enterprise should benchmark its own practices against other companies regularly in order to keep up with the competition. Benchmarking is the process of comparing an organization's business processes and performance metrics to those of leading companies in the same or other industries.

Brainstorming

Sessions where relevant business stakeholders and members of the Scrum Core Team openly share ideas through discussions and knowledge sharing sessions, which are normally conducted by a facilitator.

Business Justification

Business Justification demonstrates the reasons for undertaking a project. It answers the question "Why is this project needed?" Business justification drives all decision making related to a project.

Business Needs

Business needs are those business outcomes that the project is expected to fulfill, as documented in the Project Vision Statement.

Business Requirements

Business Requirements define what must be delivered to fulfill business needs and provide value to business stakeholders. The sum of all the insights gained through various tools such as user or customer interviews, questionnaires, JAD sessions, Gap Analysis, SWOT Analysis, and other meetings, helps get a better perspective about the business requirements and helps in creating the Prioritized Product Backlog.

Business Stakeholders

Non-core individuals such as customers, users, and sponsors who influence the project. They provide input, validate deliverables, and help align product outcomes with business goals.

Change Request(s)

Requests for changes are usually submitted as Change Requests. Change Requests remain in an unapproved status until they are formally approved.

Chief Product Owner

In the case of large projects, the Chief Product Owner prepares and maintains the overall Prioritized Product Backlog for the project. He or she coordinates work among the Product Owners of the Scrum Teams. The Product Owners, in turn, manage their respective parts of the Prioritized Product Backlog.

Chief Scrum Master

In case of large projects, the Chief Scrum Master is responsible for moderating the Scrum of Scrums (SoS) Meeting and removing impediments that affect multiple teams.

Coaching/Supportive Leader

Coaching and supportive leaders issue instructions and then support and monitor team members through listening, assisting, encouraging, and presenting a positive outlook during times of uncertainty.

Collaboration

Collaboration in Scrum refers to the Scrum Core Team working together and interfacing with the business stakeholders to create and validate the deliverables of the project to meet the goals outlined in Project Vision. Collaboration occurs when a team works together to play off each other's contributions to producing something greater.

Collaboration Plan

Collaboration is an extremely important element in Scrum and the Collaboration Plan outlines how the various decision makers, business stakeholders, and team members engage and collaborate with each other.

Colocation

Colocation is having all Scrum Core Team members located in the same work place leveraging the advantages of better coordination, problem-solving, knowledge sharing, and learning.

Commit User Stories

In this process, the Scrum Team commits to delivering User Stories approved by the Product Owner for a Sprint. The result of this process would be Committed User Stories.

Communication Plan

This plan specifies the records that must be created and maintained throughout the project. A variety of methods are used to convey important project information to business stakeholders. The Communication Plan defines these methods as well as who is responsible for the various communication activities.

Company Human Resource Plan

The company Human Resource Plan provides general information on when particular personnel will be available for various projects, programs, and portfolios. The plan also provides information about the skills and capabilities available inside the company and on plans for hiring personnel required for future efforts.

Company Mission

The Company Mission provides a framework for formulating the strategies of a company or organization that guides their overall decision making.

Company Policies

Company policies are a set of principles, rules, and guidelines formulated or adopted by an organization. Changing company policies would affect created User Stories as they have been created following existing policies.

Company Vision

Understanding the Company Vision helps the project keep its focus on the organization's objectives and the future potential of the company. The Product Owner can take guidance and direction from the Company Vision to create the Project Vision Statement.

Conduct Daily Standup

Conduct Daily Standup is a process in which a highly focused, Time-boxed meeting is conducted every day. This meeting is referred to as a Daily Standup Meeting, which is a forum for the Scrum Team to update each other on their progress and any impediments they may be facing.

Conduct Release Planning

In this process, the Scrum Core Team reviews the high-level User Stories in the Prioritized Product Backlog to develop a Release Planning Schedule, which is essentially a phased deployment schedule that can be shared with the business stakeholder(s). The Length of Sprints is also determined in this process.

Conflict Management

Team members use Conflict Management techniques to manage any conflicts that arise during a Scrum project. Sources of conflict often include schedules, priorities, resources, reporting hierarchy, technical issues, procedures, personality, and costs.

Continuous Improvement

Continuous Improvement is a Scrum approach in which the team learns from experience and business stakeholder engagement to constantly keep the Prioritized Product Backlog updated with any changes in requirements.

Continuous Value Justification

Continuous Value Justification refers to assessing business value regularly to determine whether the justification or viability of executing the project continues to exist.

Core Role(s)

Core Roles are those roles which are mandatorily required for producing the product of the project, are committed to the project, and ultimately are responsible for the success of each Sprint within the project and of the project as a whole.

Create Deliverables

Create Deliverables is the process in which the Scrum Team works on the tasks in the Sprint Backlog to create Sprint Deliverables.

Create Prioritized Product Backlog

In this process, Epic(s) are refined and elaborated, then prioritized to create a Prioritized Product Backlog for the project. The Done Criteria are also established at this point.

Create Project Vision

In this process, the project business case is reviewed to create a Project Vision Statement that will serve as the inspiration and provide focus for the entire project. The Product Owner is identified in this process.

Create User Stories

In this process, User Stories and their related User Story Acceptance Criteria are created. User Stories are usually written by the Product Owner and are designed to ensure that the customer's requirements are clearly depicted and can be fully understood by all business stakeholders.

Cumulative Flow Diagram (CFD)

A Cumulative Flow Diagram (CFD) is a useful tool for reporting and tracking project performance. It provides a simple, visual representation of project progress at a particular point in time. It is usually used to provide a higher-level status of the overall project and not daily updates for individual Sprints.

Customer

The Customer is an individual or the organization that acquires the project's product, service, or other result. For any organization, depending on the project, there can be both internal customers (i.e., within the same organization) and external customers (i.e., outside of the organization).

Customer Value-based Prioritization

Customer Value-based Prioritization places primary importance on the customer and strives to implement User Stories with the highest value first. Such high value User Stories are identified and moved to the top of the Prioritized Product Backlog.

Daily Standup Meeting

The Daily Standup Meeting is a short daily meeting, Time-boxed to 15 minutes. The team members gather to report their progress by answering the following three questions:

1. What have I done since the last meeting?
2. What do I plan to do before the next meeting?
3. What impediments or obstacles (if any) am I currently facing?

Decomposition

Decomposition is a tool whereby high-level tasks are broken down into lower level, more detailed tasks. The User Stories are decomposed into tasks by members of the Scrum Team. Prioritized Product Backlog User Stories should be sufficiently decomposed to a level that provides the Scrum Team adequate information to create deliverables from the Tasks mentioned in the Task List.

Definition of Done (Done Criteria)

Done Criteria are generic conditions that requirements should meet for them to be acceptable to the stakeholders. All conditions of the Done Criteria must be satisfied for each User Story to be considered Done. The Scrum Team should use a checklist of the general Done Criteria to ensure a task is finished and the result meets the Definition of Done (DoD). A clear Definition of Done is critical because it helps remove ambiguity and allows the team to adhere to required quality norms.

Definition of Ready

The Definition of Ready is a set of rules or criteria applicable to each User Story in the Prioritized Product Backlog. A User Story must satisfy the Definition of Ready before being considered for estimation and inclusion into a Sprint. The Definition of Ready puts the onus on the Product Owner to properly define requirements for each User Story. Without properly defined requirements, it will be impossible to get reliable estimates, and the Scrum Team may not be able to effectively complete the required project work.

Delegating Leader

Delegating Leaders are involved in the majority of decision making; however, they delegate some planning and decision-making responsibilities to team members, particularly if they are competent to handle tasks. This leadership style is appropriate in situations where the leader is in tune with specific project details and when time is limited.

Demonstrate and Validate Sprint

In this process, the Scrum Team demonstrates the Sprint Deliverables to the Product Owner and relevant business stakeholders in a Sprint Review Meeting.

Dependency Determination

Once the Scrum Team has selected User Stories for a given Sprint, they should then consider any dependencies, including those related to the availability of people as well as any technical dependencies. Documenting dependencies properly helps the Scrum Teams determine the relative order in which tasks should be executed to create the Sprint Deliverables. Dependencies highlight the relationship and interaction between tasks both within the Scrum Team working on a given Sprint and with other Scrum Teams in the project.

Design Patterns

Design Patterns provide a formal way of recording a resolution to a design problem in a specific field of expertise. These patterns record both the process used and the actual resolution, which can later be reused to improve decision making and productivity.

Develop Epic(s)

In this process, the Project Vision Statement serves as the basis for developing large, high-level, unrefined User Stories referred to as Epics. User Group Meetings may be held to *Develop Epic(s)*.

Development in Phases Contract

This contract makes funding available each month or each quarter after a release is successfully completed. It gives incentive to both customer and supplier and ensures that the monetary risk of the customer is limited to that particular time period since unsuccessful releases are not funded.

DevOps

DevOps is a framework that helps manage the whole IT infrastructure and implementation by combining development and operations teams.

Directing Leader

Directing Leaders instruct team members regarding what tasks are required and when and how they should be performed.

Discretionary Dependencies

Discretionary Dependencies are dependencies that are placed into the workflow by choice. Typically, discretionary dependencies are determined by the Scrum Team based on past experiences or best practices in a particular field or domain.

Done Criteria

Done Criteria are a set of rules that are applicable to all User Stories. A clear definition of Done is critical because it removes ambiguity from requirements and helps the team adhere to mandatory quality norms. This clear definition is used to create the Done criteria that are an output of the *Create Prioritized Product Backlog* process. User Story is considered done when it is demonstrated to and approved by the Product Owner who judges it on the basis of the Done Criteria and the User Story Acceptance Criteria.

Earned Value Analysis

Earned Value Analysis analyzes actual project performance against planned performance at a given point in time. It measures current variances in the project's schedule and cost performance and forecasts the final cost based on the determined current performance.

Empirical Process Control

An Empirical Process Control model helps make decisions based on observation and experimentation rather than on detailed upfront planning. It relies on the three main ideas of transparency, inspection, and adaptation.

Environment(s)

This refers to the identification and documentation of all the environments required to develop and test the project's deliverables.

Environment Meeting

This meeting is conducted to identify the types and number of environments required to develop, manage, and test the project's deliverables. In this meeting, the resources required to establish the required environments are also discussed.

Environment Planning Meeting

An Environment Plan Meeting is used to define a schedule/calendar of how the Scrum Teams will share environments.

Environment Schedule

Environment Schedule is a schedule/calendar of how the Scrum Teams will share the environment. It provides allocated days and time periods for each team to use each environment.

Epic(s)

Epic(s) are written in the initial stages of the project when most User Stories are high-level functionalities or product descriptions and requirements are broadly defined. They are large, unrefined User Stories in the Prioritized Product Backlog.

Estimate Range

Estimates for projects should be presented in ranges. Precise figures may give an impression of being highly accurate when in fact they may not be. In fact, estimates by definition are understood not to be precisely accurate. Estimate ranges should be based on the level of confidence the team has in each estimate.

Estimate Tasks process

In this process, the Scrum Core Team, in a Task Estimation Workshop, estimates the effort required to accomplish each task in the Task List.

Estimate User Stories

In this process, the Product Owner clarifies User Stories in order for the Scrum Master and Scrum Team to estimate the effort required to develop the functionality described in each User Story.

Estimation Criteria

The primary objective of using Estimation Criteria is to maintain relative estimation sizes and minimize the need for re-estimation. Estimation Criteria can be expressed in numerous ways, with two common examples being story points and ideal time.

Expected Monetary Value

This is a risk assessment technique where the potential monetary impact of a risk is determined based on its Expected Monetary Value (EMV). EMV is calculated by multiplying the monetary impact by the risk's probability, as approximated by the customer.

Explorer—Shopper—Vacationer—Prisoner (ESVP)

This is an exercise that can be conducted at the start of the Retrospect Sprint Meeting to understand the mind-set of the participants and set the tone for the meeting. Attendees are asked to anonymously indicate which best represents their outlook in the meeting.

External Dependencies

External dependencies are those related to tasks, activities, or products that are outside the scope of the work to be executed by the Scrum Team but are needed to complete a project task or create a project deliverable. External dependencies are usually outside the Scrum Team's control.

Fist of Five

Fist of Five is a simple and fast mechanism that can be used as an estimation practice, as well as a general group consensus building technique. After initial discussion on a given item for estimation, the Scrum Team members are each asked to vote on a scale of 1 to 5 using their fingers.

Focus Group Meetings

Focus groups assemble individuals in a guided session to provide their opinions, perceptions, or ratings of a product, service, or desired result. Focus group members have the freedom to ask questions to each other and to get clarifications on particular subjects or concepts. Through questioning, constructive criticism, and feedback, focus groups lead to a better-quality product and thereby contribute to meeting the expectations of the users.

Form Scrum Team

The Scrum Team members are identified during this process. Normally the Product Owner has the primary responsibility of selecting team members, but he or she often does so in collaboration with the Scrum Master.

Forming Stage

Forming Stage is the first stage of team formation, often considered a fun stage because everything is new and the team has not yet encountered any difficulties with the project.

Four Questions per Team

A set of questions asked in each Scrum of Scrums (SoS) Meeting. Each Scrum Team representative will provide updates from his or her team which are usually provided in the form of answers to four specific questions.

1. What has my team been working on since the last meeting?
2. What will my team do until the next meeting?
3. What were other teams counting on our team to finish that remains undone?
4. What is our team planning to do that might affect other teams?

Gap Analysis

Gap Analysis is a technique used to compare the current, actual state with some desired state and to determine how to bridge the gap between them.

Identify Environment

Identifying the number and types of environments needed because numerous Scrum Teams that will be starting and ending their Sprints on the same day.

Identify Scrum Master and Business Stakeholder(s) process

In this process, the Scrum Master and the business stakeholders are identified using specific Selection Criteria.

Identify Tasks

In this process, the Committed User Stories are broken down into specific tasks and compiled into a Task List. Tasks are identified as part of the Sprint Planning Meeting.

Industry Standards

New industry standards or changes to existing standards need to be implemented in order to maintain a viable product or service. Therefore, related User Stories need to be included in the Prioritized Program and/or Portfolio Backlog and prioritized accordingly.

Impediment

An impediment is any hindrance or hurdle that reduces the productivity of the Scrum Team.

Implement Phase

The Implement Phase includes processes related to the execution of the tasks and activities to create a project's product.

Incentive and Penalty Contract

This contract is based on the agreement that the supplier will be rewarded with a financial incentive if the project's products are delivered on time but will incur financial penalties if the delivery is late.

Incremental Delivery Contract

This contract includes inspection points at regular intervals. It helps the customer or other business stakeholders make decisions regarding product development periodically throughout the project at each inspection point. The customer can either accept the development of the product, decide to stop the development of the product, or request product modifications.

Index Cards

Index cards, often described as Story Cards, are used to track the User Stories throughout the project. This increases visibility and transparency and facilitates early discovery of any problems that may arise.

Initiate phase

This phase is composed of the processes related to initiation of a project: *Create Project Vision, Identify Scrum Master and Business Stakeholder(s), Form Scrum Team, Develop Epic(s), Create Prioritized Product Backlog, and Conduct Release Planning.*

Inspection

Inspection refers to the monitoring required to follow empirical process control, to ensure that the project deliverables conforms to the requirements.

Internal Dependencies

Internal dependencies are those dependencies between tasks, products, or activities that are under the control of the Scrum Team and within the scope of the work to be executed by the Scrum Team.

Internal Rate of Return (IRR)

Internal Rate of Return (IRR) is a discount rate on an investment in which the present value of cash inflows is made equal to the present value of cash outflows for assessing a project's rate of return. When comparing projects, one with a higher IRR is typically better.

Issues

Issues are well-defined certainties that are currently happening on the project, so there is no need to conduct a probability assessment as would be done for a risk.

Iterative Development

A Scrum principle where the product evolves through repeated cycles (Sprints), allowing for frequent feedback, risk reduction, and improved stakeholder satisfaction.

JAD Sessions

A Joint Application Design (JAD) session is a requirement gathering technique. It is a highly structured facilitated workshop which hastens the *Create Project Vision* process as it enables the business stakeholder(s) and other decision makers to come to a consensus on the scope, objectives, and other specifications of the project.

Joint Venture Contract

This contract is used when two or more parties partner to accomplish the work of a project. The parties involved in the project will both achieve some Return on Investment because the revenues or benefits generated will be shared between the parties.

Kano Analysis

Kano Analysis was developed by Noriaki Kano (1984) and involves classifying features or requirements into four categories based on customer preferences:

1. Exciters/Delighters
2. Satisfiers
3. Dissatisfiers
4. Indifferent

Laissez Faire Leader

A leadership style, where the team is left unsupervised and the leader does not interfere with their daily work activities. This often leads to a state of anarchy.

Large Core Team

The Large Core Team comprises the Chief Product Owner, Chief Scrum Master, Scrum Masters, Product Owners, and selected members of the Scrum Teams working on the large project.

Length of Sprint

Based on the various inputs including business requirements and the Release Planning Schedule, the Product Owner and the Scrum Team decide on the length of the Sprints for the project. Once determined, the length of the Sprint is usually fixed for the project. Length of Sprint is the time duration of the Sprints determined for a project.

Mandatory Dependencies

These dependencies are either inherent in the nature of the work, like physical limitations, or may be due to contractual obligations or legal requirements.

Market Study

Market Study refers to the organized research, gathering, collation, and analysis of data related to customers' preferences for products. It often includes extensive data on market trends, market segmentation, and marketing processes.

Members Selection Criteria

Members Selection Criteria are created by business stakeholders to define the Scrum Guidance Body members, their roles and responsibilities, the number of members, and their required skills and expertise.

Minimum Acceptance Criteria

The business unit defines Minimum Acceptance Criteria. They then become part of the Acceptance Criteria for any User Story for that business unit. Any functionality defined by the business unit must satisfy these Minimum Acceptance Criteria if it is to be accepted by the respective Product Owner.

Mitigated Risks

Mitigated Risks refer to the risks that are successfully addressed or mitigated by the Scrum Team during the project.

Monopoly Money

Monopoly Money is a technique that involves giving the customer "monopoly money" or "false money" equal to the amount of the project budget and asking them to distribute it among the User Stories under consideration. In this way, the customer prioritizes based on what they are willing to pay for each User Story.

MoSCoW Prioritization

The MoSCoW Prioritization scheme derives its name from the first letters of the phrases "Must have," "Should have," "Could have," and "Won't have." The labels are in decreasing order of priority with "Must have" features being those without which the product will have no value and "Won't have" features being those that, although they would be nice to have, are not necessary to be included.

Net Present Value (NPV)

Net Present Value (NPV) is a method used to determine the current net value of a future financial benefit, given an assumed inflation or interest rate.

Non-core role

Non-core roles are those roles that are not mandatorily required for the Scrum project. They may include team members who are interested in the project but have no formal role on the project team. These individuals may interface with the team but may not be responsible for the success of the project.

Norming stage

Norming is the third stage of team formation. During this stage, the team begins to mature, sort out their internal differences, and find solutions to work together. It is considered a period of adjustment.

Number of Stories

Number of Stories refers to the number of User Stories that are delivered as part of a single Sprint. It can be expressed in terms of simple count or weighted count.

Opportunities

Risks that are likely to have a positive impact on the project are referred to as opportunities.

Opportunity Cost

Opportunity cost refers to the value of the next best business option or project that was discarded in favor of the chosen project.

Organizational Deployment Methods

The deployment mechanisms of each organization tend to be different based on industry, target users, and positioning. Depending on the product being delivered, deployment can take place remotely or may involve the physical shipping or transition of an item.

Organizational Resource Matrix

The Organizational Resource Matrix is a hierarchical depiction of a combination of a functional organizational structure and a project organizational structure. Matrix organizations bring together team members for a project from different functional departments such as information technology, finance, marketing, sales, manufacturing, and other departments - and create cross-functional teams.

Paired Comparison

Paired Comparison is a technique where a list of all the User Stories in the Prioritized Product Backlog is prepared. Next, each User Story is taken individually and compared with the other User Stories in the list, one at a time. Each time two User Stories are compared, a decision is made regarding which of the two is more important. Through this process, a prioritized list of User Stories can be generated.

Pareto Analysis

This technique of assessing risk involves ranking risks by magnitude. It helps the Scrum Team address the risks in the order of potential impact of those risks on the project.

PDCA/PDSA cycle

The Plan-Do-Check-Act Cycle—also known as the Deming or Shewhart Cycle—was developed by Dr. W. Edwards Deming, considered the father of modern quality control and Dr. Walter A. Shewhart. Deming later modified Plan-Do-Check-Act to Plan-Do-Study-Act (PDSA) because he felt the term “Study” emphasized analysis rather than simply inspection, as implied by the term “Check.” Both Scrum and the Deming/Shewhart/PDCA Cycle are iterative methods that focus on continuous improvement.

Performing stage

The final stage of team formation when the team becomes its most cohesive and operates at its highest level in terms of performance. The members have evolved into an efficient team of peer professionals who are consistently productive.

Personas

Personas are highly detailed fictional characters, representative of the majority of users as well as other business stakeholders who may not directly use the end product. Personas are created to identify the needs of the target user base.

Piloting Plan

A Piloting Plan can be used to map out a pilot deployment in detail. The scope and objectives of the deployment, the target deployment user base, a deployment schedule, transition plans, required user preparation, evaluation criteria for the deployment, and other key elements related to the deployment are specified in the Pilot Plan and shared with business stakeholders.

Plan and Estimate phase

A Scrum phase that consists of processes related to planning and estimating User Stories and associated tasks, which include *Create User Stories*, *Estimate User Stories*, *Commit User Stories*, *Identify Tasks*, *Estimate Tasks*, and *Update Sprint Backlog*.

Planning for Value

Planning for Value refers to justifying and confirming the project value. The onus for determining how value is created falls on the business stakeholders (sponsor(s), customers, and/or users), while the Scrum Team concentrates on what is to be developed.

Planning Poker

Planning Poker, also called Estimation Poker, is an estimation technique which balances group thinking and individual thinking to estimate relative sizes of User Stories or the effort required to develop them.

Portfolio

A portfolio is a group of related programs and/or projects, with the objective of delivering business outcomes as defined in the Portfolio Vision Statement. The Prioritized Portfolio Backlog incorporates the Prioritized Program or Project Product Backlogs for all the programs in the portfolio.

Portfolio Product Owner

The Portfolio Product Owner defines the strategic objectives and priorities for the portfolio.

Portfolio Scrum Master

The Portfolio Scrum Master solves problems, removes impediments, facilitates, and conducts meetings for the portfolio.

Potentially Shippable Deliverables from Projects

Potentially Shippable Deliverables from projects are valuable inputs for coordination at the program or portfolio level. At the end of Sprints in projects, product increments or deliverables are completed. The User Stories included in these increments meet the Definition of Done criteria as well as their respective Acceptance Criteria.

Prioritization

Prioritizing can be defined as determining the order of things and separating what will be done now from what can be done later.

Prioritized Product Backlog

The Prioritized Product Backlog is a single requirements document that defines the project scope by providing a prioritized list of features of the product or service to be delivered by the project.

Probability Impact Grid

A grid where Risks are assessed for probability of occurrence and for potential impact on project objectives. A numerical rating is assigned to both probability and impact independently. The two values are then multiplied to derive a risk severity score, which can be used to prioritize risks.

Probability Trees

Potential events are represented in a diagram with a branch for each outcome of the events. The probability of each outcome is indicated on the appropriate branch, and these values can be used to calculate the overall impact of risk occurrence in a project.

Product

The term “product” in the *SBOK® Guide* may refer to a product, service, or other deliverable that provides value to the customer.

Product Owners Collaboration Plan

The Product Owners Collaboration Plan should define how multiple Product Owners collaborate with the Chief Product Owner.

Prioritized Product Backlog Review Meeting

A Product Backlog Review Meeting (also referred to as a Prioritized Product Backlog Refining Session) is a formal meeting during the *Refine Prioritized Product Backlog* process, which helps the Scrum Team review and gain consensus about the Prioritized Product Backlog.

Prioritized Program or Portfolio Backlog Review Meetings

At the program or portfolio level, there is representation from each project in the program or from each program in the portfolio. To streamline the meeting, it is recommended to have only one representative from each project or program attend at the program or portfolio level.

Product Owner

The Product Owner is the person responsible for maximizing business value for the project. He or she is responsible for articulating customer requirements and maintaining business justification for the project.

Program

A program is a group of related projects, with the objective of delivering business outcomes as defined in the Program Vision Statement. The Prioritized Program Backlog incorporates the Prioritized Product Backlogs for all the projects in the program.

Program and Portfolio Risks

Risks related to a portfolio or program that will also impact projects that are part of the respective portfolio or program.

Program Product Owner

The Program Product Owner defines the strategic objectives and priorities for the program.

Program Scrum Master

The Program Scrum Master solves problems, removes impediments, facilitates, and conducts meetings for the program.

Project

A project is a collaborative enterprise to either create new products or services or to deliver results as defined in the Project Vision Statement. Projects are usually impacted by constraints of time, cost, scope, quality, people, and organizational capabilities.

Project Benefits

Project benefits include all measurable improvements in a product, service or result which could be provided through successful completion of a project.

Project Budget

The project budget is a financial document which includes the cost of people, materials, and other related expenses in a project. The project budget is typically signed off by the sponsor(s) to ensure that sufficient funds are available.

Project Charter

A project charter is an official statement of the desired objectives and outcomes of the project. In many organizations, the project charter is the document that officially and formally authorizes the project, providing the team with written authority to begin project work.

Project Costs

Project costs are investment and other development costs for a project

Project Reasoning

Project reasoning includes all factors which support or contribute to the need for the project, whether positive or negative, chosen or not (e.g., inadequate capacity to meet existing and forecasted demand, decrease in customer satisfaction, low profits, and legal requirements).

Project Timescales

Timescales reflect the length or duration of a project. Timescales related to the business case also include the time over which the project's benefits will be realized.

Project Vision Meeting

A Project Vision Meeting is a meeting with the Program Business Stakeholder(s), Program Product Owner, Program Scrum Master, and Chief Product Owner. It helps identify the business context, business requirements, and business stakeholder expectations in order to develop an effective Project Vision Statement.

Project Vision Statement

The key output of the *Create Project Vision* process is a well-structured Project Vision Statement. A good Project Vision explains the business need and what the project is intended to meet rather than how it will meet the need.

Proposed Non-Functional Items for Product Backlog

Non-functional requirements may not be fully defined in the initial stages of the project and can surface during the Sprint Review or Retrospect Sprint Meetings. These items should be added to the Prioritized Product Backlog as they are discovered.

Quality

Quality is defined as the ability of the completed product or deliverables to meet the Acceptance Criteria and achieve the business value expected by the customer.

Quality Assurance

Quality assurance refers to the evaluation of processes and standards that govern quality management in a project to ensure that they continue to be relevant. Quality assurance activities are carried out as part of the work.

Quality Control

Quality control refers to the execution of the planned quality activities by the Scrum Team in the process of creating deliverables that are potentially shippable. It also includes learning from each set of completed activities in order to achieve continuous improvement.

Quality Management

Quality management in Scrum enables customers to become aware of any problems in the project early and helps them recognize if a project is going to work for them or not. Quality management in Scrum is facilitated through three interrelated activities:

1. Quality planning
2. Quality control
3. Quality assurance

Quality Planning

Quality Planning refers to identification and definition of the product required from Sprint and the project along with the Acceptance Criteria, any development methods to be followed, and the key responsibilities of Scrum Team members in regard to quality.

Recommended Scrum Guidance Body Improvements

As a result of planning for the large project, suggestions may be made to revise or enhance the Scrum Guidance Body Recommendations. If the Guidance Body accepts these suggestions, they will be incorporated as updates to the Scrum Guidance Body documentation.

Refactoring

Refactoring is a tool specific to software projects. The aim of this technique is to improve the maintainability of the existing code and make it simpler, more concise, and more flexible. Refactoring means improving the design of the present code without changing how the code behaves. It involves the following:

- Eliminating repetitive and redundant code
- Breaking methods and functions into smaller routines
- Clearly defining variables and method names
- Simplifying the code design
- Making the code easier to understand and modify

Refine Prioritized Product Backlog

An ongoing process where the Product Owner and team update and reprioritize the backlog based on feedback, changes, and learnings from the project.

Regulations

Regulations include any Federal, Local, State, or industry regulations that the program or portfolio must adhere to. Sometimes, Scrum Guidance Body recommendations may need to be updated to reflect new regulations.

Rejected Deliverables

Rejected Deliverables are the deliverables that do not meet the defined Acceptance Criteria. A list of Rejected Deliverables is maintained and updated after each Sprint Review Meeting with any deliverables that were not accepted.

Rejected Updates to the Scrum Guidance Body Recommendations

Recommended Scrum Guidance Body Improvements may not always be accepted. If the recommended improvement is rejected by the Scrum Guidance Body members, feedback that shares the reason for that rejection is provided to the relevant party.

Relative Prioritization Ranking

Relative Prioritization Ranking is a simple listing of User Stories in order of priority. It is an effective method for determining the desired User Stories for each iteration or release of the product or service.

Relative Sizing/Story Points

In addition to being used for estimating cost, Story Points may also be used for estimating the overall size of a User Story or feature. This approach assigns a story point value based on an overall assessment of the size of a User Story with consideration given to risk, amount of effort required, and level of complexity.

Release Content

This consists of essential information about the deliverables that can assist the Customer Support Team.

Release Notes

Release Notes should include external or market facing shipping criteria for the product to be delivered.

Release Planning Schedule

A Release Planning Schedule is one of the key outputs of the *Conduct Release Planning* process. A Release Planning Schedule states which deliverables are to be released to the customers, along with planned intervals, and dates for releases. There may not be a release scheduled at the end of every Sprint iteration.

Release Planning Sessions

The major objective of Release Planning Sessions is to create a Release Plan Schedule and enable the Scrum Team to have an overview of the releases and delivery schedule for the product they are developing, so that they can align with the expectations of the Product Owner and relevant Business stakeholder(s).

Release Preparation Methods

Release preparation methods are the methods used to execute the tasks identified in the Release Readiness Plan in order to get the deliverables ready to be shipped/released.

Release Prioritization Methods

Release Prioritization Methods are used to develop a Release Planning Schedule. These methods are usually industry- and/or organization-specific and are usually determined by senior management in an organization.

Release Readiness Plan

It details the steps to be taken by relevant Scrum Teams and any other individuals to confirm that the minimum requirements for release have been met, and the product or product increment is ready for release.

Release Readiness Sprint

If there is a need for specific tasks to be performed to get ready for a Release and to confirm that the minimum requirements for release have been met, these tasks are performed in a Release Readiness Sprint. A Release Readiness Sprint, if required, is only done once per Release as the last Sprint prior to the Release.

Resolved Issues

In Scrum of Scrums Meetings, Scrum Team members have the opportunity to transparently discuss issues impacting their project. This timely discussion and resolution of issues in the Scrum of Scrums Meeting improves coordination between different Scrum Teams and reduces the need for redesign and rework.

Retrospect Program or Portfolio Meeting

The Retrospect Program or Portfolio Meeting is similar to the Retrospect Release Meeting but is carried out at the program or portfolio level. The major difference is that the frequency of Retrospect Program and Portfolio Meetings is much lower than that of Retrospect Release Meetings.

Retrospect Release

In this process, which completes the project, organizational business stakeholders and Scrum Core Team members assemble to retrospect the project and identify, document, and internalize lessons learned. Often, these lessons lead to the documentation of agreed actionable improvements, to be implemented in future projects.

Retrospect Release Meeting

The Retrospect Release Meeting is a meeting to determine ways in which team collaboration and effectiveness can be improved in future projects. Positives, negatives, and potential opportunities for improvement are also discussed. This meeting is not Time-boxed and may be conducted in person or in a virtual format.

Retrospect Sprint

In this process, the Scrum Master and Scrum Team meet to discuss the lessons learned throughout the Sprint. The lessons learned are documented and can be applied to future Sprints.

Retrospect Sprint Log(s)

The Retrospect Sprint Log is a record of the opinions, discussions, and actionable items raised in a Retrospect Sprint Meeting. The Scrum Master may facilitate creation of this log with inputs from Scrum Core Team members.

Retrospect Sprint Meeting

The Retrospect Sprint Meeting is Time-boxed to 4 hours for a one-month Sprint and conducted as part of the *Retrospect Sprint* process. The length may be scaled up or down relative to the length of the Sprint. During this meeting, the Scrum Team gets together to review and reflect on the previous Sprint in terms of the processes followed, tools employed, collaboration and communication mechanisms, and other aspects relevant to the project.

Return on Investment (ROI)

Return on Investment (ROI), when used for project justification, assesses the expected net income to be gained from a project. It is calculated by deducting the expected costs or investment in a project from its expected revenue and then dividing this (net profit) by the expected costs in order to get a return rate.

Review and Retrospect Phase

The fourth phase in the Scrum process where deliverables are demonstrated to stakeholders and the team reflects on lessons learned to improve future performance.

Risk

Risk is defined as an uncertain event or set of events that can affect the objectives of a project and may contribute to its success or failure.

Risk Appetite

Risk appetite refers to how much uncertainty a business stakeholder or organization is willing to take on.

Risk Assessment

Risk assessment refers to evaluating and estimating identified risks.

Risk Attitude

The Risk Attitude of the Business stakeholder(s) determines how much risk the Business stakeholder(s) consider acceptable. This is a determining factor when they decide to take actions to mitigate potential adverse risks.

Risk Averse

Risk Averse is one of the categories of Utility Function. It refers to a Business stakeholder being unwilling to accept a risk no matter what the anticipated benefit or opportunity.

Risk Breakdown Structure

In this structure, risks are grouped based on their categories or commonalities. For example, risks may be categorized as financial, technical, or safety related.

Risk Burndown Chart

A chart that depicts cumulative project risk severity over time. The likelihood of the various risks are plotted on top of each other to show cumulative risk on the y-axis. The initial identification and evaluation of risks and the creation of the Risk Burndown Chart are done early in the project.

Risk Checklists

Risk Checklists include key points to be considered while identifying risks, common risks encountered in Scrum projects, or even categories of risks that should be addressed by the team.

Risk Communication

Risk Communication involves communicating the findings from the first four steps of Risk Management to the appropriate Business stakeholder(s) and determining their perception regarding the uncertain events.

Risk Identification

Risk Identification is a key step in Risk Management which involves using various techniques to identify all potential risks.

Risk Meeting

The Product Owner can prioritize risks by calling a meeting of the Scrum Core Team and optionally inviting relevant business stakeholders to the meeting.

Risk Mitigation

Risk Mitigation is a major step in Risk Management that involves developing an appropriate strategy to deal with a risk.

Risk Neutral

Risk Neutral is one of the categories of Utility Function that refers to a business stakeholder being neither risk averse nor risk seeking; any given decision is not affected by the level of uncertainty of the outcome.

When two scenarios carry the same level of benefit, the risk neutral business stakeholder will not be concerned if one scenario is riskier than the other.

Risk Prioritization

Risk Prioritization is a key step in Risk Management that involves prioritizing risks to be included for specific action in the Prioritized Product Backlog.

Risk Prompt Lists

Risk Prompt Lists are used in stimulating thoughts regarding the source from which risks may originate. Risk Prompt Lists for various industries and project types are available publicly.

Risk Seeking

Risk Seeking is one of the categories of Utility Function that refers to a business stakeholder being willing to accept risk even if it delivers a marginal increase in return or benefit to the project.

Risk Threshold

Risk Threshold refers to the level at which a risk is acceptable to the business stakeholder's organization. A risk will fall above or below the risk threshold. If it is below, the business stakeholder or organization is more likely to accept the risk.

Risk Tolerance

Risk Tolerance indicates the degree, amount, or volume of risk the business stakeholders will withstand.

Risk-Based Spike

Risk-Based Spikes are experiments that involve research or prototyping to better understand potential risks. In a spike, an intense two to three-day exercise is conducted (preferably at the beginning of a project before the *Develop Epic(s)* or *Create Prioritized Product Backlog* processes) to help the team determine the uncertainties that could affect the project.

Scope

The scope of a project is the total sum of all the product increments and the work required for developing the final product.

Scrum

An Agile framework used for managing complex work. It emphasizes collaboration, iteration, transparency, and adaptability to deliver high-value products efficiently.

Scrum Aspects

Key areas that must be addressed in every Scrum project: Organization, Business Justification, Quality, Change, and Risk. These aspects support the application of Scrum principles.

Scrum Core Team

Consists of the Product Owner, Scrum Master, and Scrum Team. They are directly responsible for delivering value through Scrum processes.

Scrum Guidance Body

The Scrum Guidance Body (SGB) is an optional role. It consists of a group of documents and/or a group of experts who are typically involved with defining objectives related to quality, government regulations, security, and other key organizational parameters.

Scrum Guidance Body Expertise

Scrum Guidance Body Expertise relates to documented rules and regulations, development guidelines, or standards, and best practices.

Scrum Guidance Body Meetings

The Scrum Guidance Body meets regularly to discuss the potential need for an update of the Scrum Guidance Body Recommendations (e.g., recommended improvements from Retrospectives and other processes, updated regulations, etc.). The Scrum Guidance Body decides the frequency of the meetings based on the specific needs of the enterprise.

Scrum Guidance Body Members

The Scrum Guidance Body (SGB) members can include Scrum experts, selected Scrum Masters, Product Owners and team members (on all levels). However, there should be a limit on the number of members that a SGB can have in order to ensure that it remains relevant and does not become prescriptive in nature.

Scrum Master

The Scrum Master is one of the Scrum Core Team roles. He or she facilitates creation of the project's deliverables, manages risks, changes, and impediments during the *Conduct Daily Standup*, *Retrospect Sprint*, and other Scrum processes.

Scrum Masters/Scrum Teams Collaboration Plan

The Scrum Masters/Scrum Teams Collaboration Plan defines how the numerous Scrum Masters and Scrum Teams collaborate with each other to provide highest value in the shortest possible time.

Scrum Meetings or Ceremonies

Key events including Sprint Planning, Daily Standups, Sprint Reviews, and Sprint Retrospectives. These enable communication, transparency, and alignment in Scrum projects.

Scrum of Scrums Meeting

A Scrum of Scrums Meeting is a crucial component when implementing Scrum for large-scale projects. Typically, there is one representative in the meeting from each Scrum Team—usually the Scrum Master—but it is also common for anyone from the Scrum Team to attend the meeting if required. The meeting is facilitated by the Chief Scrum Master and aims to address coordination and integration among the various Scrum Teams.

Scrum of Scrum of Scrums Meeting

At the program and especially at the portfolio level, it makes sense to have another layer of meetings. Representatives from relevant or interrelated programs and projects in the program or portfolio meet at regular intervals, or as required. In attendance there would be representatives from each of the Scrum of Scrums meetings. This additional level of meetings is called the Scrum of Scrum of Scrums (SoSoS).

Scrum Principles

Six non-negotiable guidelines that form the foundation of Scrum: Empirical Process Control, Self-Organization, Collaboration, Value-based Prioritization, Time-boxing, and Iterative Development.

Scrum Processes

Nineteen essential activities grouped under five phases (Initiate, Plan and Estimate, Implement, Review and Retrospect, and Release) that guide execution of Scrum projects.

Scrum Team

The Scrum Team is one the Scrum Core Team roles. The Scrum Team works on creating the deliverables of the project and contributes to realizing business value for all business stakeholders and the project.

Scrum Team Lessons Learned

The self-organizing and empowered Scrum Team is expected to learn from mistakes made during a Sprint and these lessons learned help the teams improve their performance in future Sprints.

Scrum Team Representatives

A representative nominated by the team to represent them in the Scrum of Scrums (SoS) Meetings based on who can best fulfill the role depending on current issues and circumstances.

Scrumboard

Scrumboard is a tool used by the Scrum Team to plan and track progress during each Sprint. The Scrumboard contains four columns to indicate the progress of the estimated tasks for the Sprint: a To Do column for tasks not yet started, an In Progress column for the tasks started but not yet completed, a Testing column for tasks completed but in the process of being tested, and a Done column for the tasks that have been completed and successfully tested.

Self-organization

Scrum practices encourage the idea that employees are self-motivated and seek to accept greater responsibility. Hence, they deliver much greater value when self-organized.

Shared Resources

Shared resources can include people, environment, and equipment that are needed by all or some of the Scrum Teams working on the project. In a large project, the shared resources may be limited and are needed by all or some of the Scrum Teams at the same time.

Ship Deliverables

In this process, Accepted Deliverables are delivered or transitioned to the relevant Business stakeholder(s). A formal Working Deliverables Agreement documents the successful completion of the Sprint.

Simple Schemes

Simple Schemes involve labeling items as Priority “1”, “2”, “3” or “High”, “Medium” and “Low” and so on. Although this is a simple and straightforward approach, it can become problematic because there is often a tendency to label everything as Priority “1” or “High”.

Skills Requirement Matrix

The skills requirement matrix, also known as a competency framework, is used to assess skill gaps and training requirements for team members. A skills matrix maps the skills, capabilities, and interest levels of team members in using those skills and capabilities on a project. Using this matrix, the organization can assess any skill gaps in team members and identify the employees who will need further training in a particular area or competency.

Speed Boat

Speed Boat is a technique that can be used to conduct the Retrospect Sprint Meeting. Team members play the role of the crew on a Speed Boat. The boat must reach an island, which is symbolic of Project Vision. The attendees use sticky notes to record engines and anchors. Engines are things which help them reach the island, while anchors are things that are hindering them from reaching the island. This exercise is Time-boxed to a few minutes.

Sponsor

The sponsor is the individual or the organization that provides resources and support for the project. The sponsor is also the business stakeholder to whom Scrum team members are accountable. The sponsor may be one or more individuals or an organizational group.

Sprint

A Sprint is a Time-boxed iteration of one to four weeks in duration during which the Scrum Team works on and creates the Sprint Deliverables.

Sprint Backlog

Sprint Backlog is a list of the tasks to be executed by the Scrum Team in the current Sprint.

Sprint Burndown Chart

Sprint Burndown Chart is a graph that depicts the amount of work remaining in the ongoing Sprint.

Sprint Deliverables

Sprint Deliverables refer to product increments or deliverables that are completed at the end of each Sprint.

Sprint Planning Meeting

Sprint Planning Meeting is conducted at the beginning of a Sprint as part of the *Update Sprint Backlog* process. It is Time-boxed to eight hours for a one-month Sprint and is divided into two parts - Objective Definition and Task Estimation.

Sprint Retrospective

Held at the end of a Sprint, this meeting allows the team to reflect on what went well, what didn't, and how to improve in the next Sprint.

Sprint Review Meeting

The Sprint Review Meeting is Time-boxed to four hours for a one-month Sprint and can be scaled according to the length of the Sprint. During the Sprint Review Meeting, the Scrum Team presents the deliverables of the current Sprint to the Product Owner, who may accept or reject the deliverables.

Sprint Tracking Tools

Sprint Tracking Tools are used to track the progress of a Sprint and to know where the Scrum Team stands in terms of completing the tasks in the Sprint Backlog. A variety of tools can be used to track the work in a Sprint, but one of the most common is a Scrumboard, also known as a task board or progress chart.

Sprint Velocity

Sprint Velocity is the rate at which the team can complete the work in a Sprint. It is usually expressed in the same units as those used for estimating, normally story points or ideal time.

Stakeholder Analysis

A standard stakeholder analysis is used to identify the business stakeholders on program and portfolio levels. Further details related to program or portfolio business stakeholders may be identified as personas in the Create and Refine Program or Portfolio Backlog process.

Storming Stage

The second stage of team formation where the team begins trying to accomplish the work. However, power struggles may occur and there is often chaos or confusion among team members.

Story Mapping

Story Mapping is a technique to provide a visual outline of the product and its key components. Story Mapping, first formulated by Jeff Patton (2005), is commonly used to illustrate product roadmaps. Story maps depict the sequence of product development iterations and map out which features will be included in the first, second, third, and subsequent releases.

Supporting Leader

Supporting leaders employ listening, empathy, commitment, and insight while sharing power and authority with team members. Supporting leaders are stewards who achieve results by focusing on the needs of the team. This style is the embodiment of the Scrum Master role.

Sustainable Pace

Sustainable Pace is the pace at which the team can work and comfortably maintain. It translates to increased employee satisfaction, stability, and increased estimation accuracy, all of which leads to increased customer satisfaction.

SWOT Analysis

SWOT is a structured approach to project planning that helps evaluate the strengths, weaknesses, opportunities, and threats related to a project. This type of analysis helps identify both the internal and the external factors that could impact the project.

Target Customers for Release

Not every release will target all business stakeholders or users. The business stakeholders may choose to limit certain releases to a subset of users. The Release Plan specifies the Target Customers for Release.

Task Estimation Workshop

Task Estimation Workshops enable the Scrum Team to estimate the effort required to complete a task or set of tasks and to estimate the people effort and other resources required to carry out the tasks within a given Sprint.

Task List

This is a comprehensive list that contains all the tasks which the Scrum Team has committed to for the current Sprint. It contains descriptions of each task and may also contain task estimates.

Task-Oriented Leader

Task-Oriented Leaders enforce task completion and adherence to deadlines.

Team Building Plan

Since a Scrum Team is cross-functional, each member needs to participate actively in all aspects of the project. The Scrum Master should identify potential issues that could crop up with team members and try to address them diligently in the Team Building Plan in order to maintain an effective team.

Team Calendar

A Team Calendar contains information regarding availability of team members including information related to employee vacations, leaves, notable events, and holidays.

Team Expertise

Team Expertise refers to the expertise of the Scrum Team members to understand the User Stories and Tasks in the Sprint Backlog in order to create the final deliverables. Team Expertise is used to assess the inputs needed to execute the planned work of the project.

Team Specialization

In a large project, Team Specialization may be required. There are three dimensions of Team Specialization. The first dimension is the need for accomplishing specific tasks. The second dimension is the need for unique skills of single team members. The third dimension is that there may be limitations in team flexibility.

Technical Debt

Technical Debt (also referred to as design debt or code debt) refers to the work that teams prioritize lower, omit, or do not complete as they work towards creating the primary deliverables associated with the project's product. Technical Debt accrues and must be paid in the future.

Theory X

Theory X leaders assume that employees are inherently unmotivated and will avoid work if possible, warranting an authoritarian style of management.

Theory Y

Theory Y leaders assume that employees are self-motivated and seek to accept greater responsibility. Theory Y involves a more participative management style.

Theory Z

Theory Z leaders assume that employees will focus on self-actualization when their basic necessities are met.

Threats

Threats are risks that could affect the project in a negative manner.

Three Daily Questions

Three Daily Questions used in Daily Standup Meetings which are facilitated by the Scrum Master, where each Scrum Team member provides information in the form of answers to three specific questions:

- What have I done since the last meeting?
- What do I plan to do before the next meeting?
- What impediments or obstacles (if any) am I currently facing?

Time-boxing

Time-boxing refers to setting short periods of time for work to be done. If the work undertaken remains incomplete at the end of the Time-box, it is moved into a subsequent Time-box. Time-boxes provide the structure needed for Scrum projects, which have an element of uncertainty, are dynamic in nature, and are prone to frequent changes.

Transparency

Transparency allows all facets of any Scrum process to be observed by anyone. Sharing all information leads to a high trust environment.

Unapproved Change Requests

Requests for changes are usually submitted as Change Requests. Change Requests remain unapproved until they get formally approved.

Updated Program Product Backlog

A Program Product Backlog that undergoes periodic refining to incorporate changes and new requirements.

Updated Scrum Guidance Body Membership

As a result of assessing the Scrum Guidance Body membership, new members may be included in the Scrum Guidance Body and existing members may be removed or leave the Scrum Guidance Body.

Update Sprint Backlog

In this process, the Scrum Core Team holds Sprint Planning Meetings where the group creates a Sprint Backlog containing all tasks to be completed in the Sprint.

Updated Implementation Deadlines for Projects

Implementation deadlines for projects may be updated to reflect the impact of new or changed User Stories that need to modify or introduce new requirements.

Updated Prioritized Program or Portfolio Backlog

The Prioritized Program or Portfolio Backlog may be updated with new User Stories, new Change Requests, new identified risks, updated User Stories, or reprioritization of existing User Stories.

User

Users are the individuals or the organization that directly uses the project's product, service, or other results. Like customers, for any organization, there can be both internal and external users. In some cases, customers and users may be the same.

User Group Meetings

User Group Meetings involve relevant Business stakeholder(s), primarily users or customers of the product. They provide the Scrum Core Team with first-hand information about user expectations. This helps in formulating the Acceptance Criteria for the product and provides valuable insights for developing Epics.

User Stories

User Stories adhere to a specific, predefined structure and are a simplistic way of documenting the requirements and desired end-user functionality. The requirements expressed in User Stories are short, simple, and easy-to-understand statements resulting in enhanced communication among the business stakeholders and better estimations by the team.

User Story Acceptance Criteria

Every User Story has associated Acceptance Criteria. User Stories are subjective, so the Acceptance Criteria provide the objectivity required for the User Story to be considered as Done or not Done during the Sprint Review providing clarity to the team on what is expected of a User Story.

User Story Workshops

User Story Workshops are held as part of the *Develop Epic(s)* process. The Scrum Master facilitates these sessions. The entire Scrum Core Team is involved and at times it is desirable to include other Business stakeholder(s).

User Story Writing Expertise

The Product Owner, based on his or her interaction with the business stakeholders, business knowledge and expertise, and inputs from the team, develops User Stories that form the initial Prioritized Product Backlog for the project.

Utility Function

Utility Function is a model used for measuring business stakeholder risk preference or attitude toward risk. It defines the Business stakeholder(s)' level or willingness to accept risk.

Value-based Prioritization

A principle that ensures the highest-value features or requirements are addressed first. It drives early delivery of value and helps optimize ROI.

Value Stream Mapping

Value Stream Mapping uses flowcharts to illustrate the flow of information needed to complete a process and may be used to streamline a process by helping to determine non-value-adding elements.

Vendor

Vendors include external individuals or organizations that provide products and services that are not within the core competencies of the project organization.

Voice of the Customer (VOC)

The Voice of the Customer (VOC) can be referred to as the explicit and implicit requirements of the customer, which must be understood prior to the designing of a product or service. The Product Owner represents the Voice of the Customer.

War Room

War Room is the commonly used term to describe the location where all Scrum Team members working are located. Normally, it is designed in such a way that team members can move around freely, work, and communicate easily because they are located in close proximity to each other.

Wideband Delphi Technique

Wideband Delphi is a group-based estimation technique for determining how much work is involved and how long it will take to complete. Individuals within a team anonymously provide estimations for each feature and the initial estimates are then plotted on a chart. The team then discusses the factors that influenced their estimates and proceed to a second round of estimation. This process is repeated until the estimates of individuals are close to each other and a consensus for the final estimate can be reached.

Working Deliverables

This output is the final shippable deliverable for which the project was sanctioned.

Working Deliverables Agreement

Deliverables that meet the Acceptance Criteria receive formal business sign-off and approval from the customer or the sponsor.

This Page is Intentionally Left Blank.

INDEX

1

100-point method, 185

6

6SigmaStudy.com, 9

A

Acceptance Criteria, 102

Accepted User Stories, 271

Adaptability, 4

Adaptation, 29

Affinity Estimation, 214

Agile Manifesto, 34

Agreed Actionable Improvements, 276

AI, 7

Applicable Contracts, 176

Appropriation, 34

Approved Change Requests, 117, 176

Approved Changes, 181

Articulation, 34

Aspects, 10

Assertive, 76

Assigned Action Items and Due Dates, 276, 356

Authorized Training Partners, 9

Autocratic, 75

Awareness, 34

B

Back-ups, 173

BALearning.com, 7, 9

BARG™, 7

BARG™, 7, 8, 9

Brainstorming, 137

Business Analysis, 7, 8, 9

Business Analysis Reference Guide, 7, 8, 9

Business Analysis Reference Guide (BARG™), 7, 8, 9

Business Analyst, 7

Business Analysts, 8

Business case, 85

Business Justification, 10, 14, 79, 82, 83, 84, 85, 86, 87, 96, 372

Business needs, 84

Business Requirements, 184

Business Stakeholder(s), 13, 55, 120

business stakeholders, 1, 3, 4, 6, 12, 13, 15, 16, 18, 19, 23, 24, 25, 27, 29, 31, 32, 34, 42, 51, 55, 56, 57, 58, 62, 63, 65, 79, 81, 83, 84, 88, 92, 96, 99, 100, 102, 104, 106, 108, 109, 113, 115, 117, 120, 121, 123, 124, 125, 126, 127, 128, 130, 131, 133, 136, 137, 138, 142, 144, 145, 147, 149, 150, 155, 156, 157, 162, 164, 165, 166, 173, 174, 176, 178, 179, 181, 186, 191, 194, 196, 197, 199, 200, 206, 208, 239, 240, 247, 256, 257, 263, 264, 266, 279, 280, 282, 284, 285, 286, 287, 293, 294, 296, 297, 299, 300, 307, 310, 312, 315, 316, 318, 319, 320, 321, 322, 325, 328, 331, 332, 333, 334, 337, 339, 347, 352, 354, 355, 362, 372, 373, 374, 375, 377, 380, 381, 383, 385, 391, 393, 394, 396, 398, 399, 402

Business Stakeholders, 257

Business value, 101

Business value delivered, 236

C

Change, 15, 115, 117

Change Requests, 16, 117

Chief Product Owner, 63

Chief Scrum Master, 63

Coaching, 76

Collaboration, 12, 23, 34

Collaboration Plan, 173

Collective ownership, 4

Colocation, 37

Commit User Stories, 4, 17, 18, 20, 44, 47, 57, 59, 60, 83, 126, 199, 200, 217, 218, 219, 260, 304, 374, 385

Committed User Stories, 220, 227

Communication Plan, 285

Communication Techniques, 259

Communications Plan, 286

Company Vision, 155

Conduct Daily Standup, 4, 17, 19, 20, 25, 34, 35, 44, 45, 59, 60, 83, 85, 121, 130, 176, 240, 250, 251, 374, 395
inputs, 251
outputs, 254
tools, 252

Conduct Release Planning, 17, 18, 20, 27, 55, 57, 59, 60, 125, 149, 150, 191, 192, 193, 196, 259, 301, 323, 374, 381, 390
inputs, 192
outputs, 196
tools, 194

Conduct Release Planning process, 191
Confirm benefits realization, 85, 95
 Conflict Management, 74
 Techniques, 74
Continuous delivery of value, 4
Continuous feedback, 4
Continuous improvement, 4, 15
 Continuous integration, 109, 123
Continuous value justification, 85, 92
Core roles, 13, 53
 Create Deliverables, 17, 19, 29, 31, 32, 34, 44, 47, 57, 59, 60, 81, 121, 130, 142, 147, 176, 240, 243, 244, 247, 248, 249, 306, 324, 375
 inputs, 244
 outputs, 248
 tools, 247
Create Prioritized Product Backlog, 4, 17, 18, 27, 29, 34, 42, 55, 57, 59, 72, 83, 118, 124, 125, 137, 140, 141, 142, 149, 150, 182, 183, 197, 209, 259, 301, 307, 323, 375, 378, 381, 394
 inputs, 183
 outputs, 189
 tools, 185
 Create Prioritized Product Backlog process, 182
 Create Project Vision, 16, 17, 20, 31, 34, 57, 149, 150, 153, 154, 156, 158, 162, 174, 297, 321, 375, 381, 382, 388
 inputs, 155
 outputs, 158
 tools, 156
 Create User Stories, 17, 18, 20, 29, 31, 47, 57, 59, 60, 72, 83, 126, 199, 200, 203, 204, 205, 209, 259, 303, 375, 385
 inputs, 204
 outputs, 208
 tools, 206
 Create User Stories process, 203
 Create/Refine Prioritized Program or Portfolio Backlog
 inputs, 344
 Create/Refine Prioritized Program or Portfolio Backlog
 tools, 347
 Create/Update Program or Portfolio Releases inputs, 350
 Create/Update Program or Portfolio Releases outputs, 353
 Create/Update Program or Portfolio Releases process, 350
 Create/Update Program or Portfolio Releases tools, 352
 Cross-functional, 122
 Cumulative Flow Diagram (CFD), 93
Customer, 56
Customer centric, 4

Customer Value-based Prioritization, 89, 123
 customers, 6, 13, 15, 16, 55, 56, 57, 71, 81, 88, 90, 95, 100, 108, 113, 117, 120, 123, 137, 156, 165, 166, 178, 179, 187, 196, 333, 360, 362, 376, 383, 385, 389, 390, 402
Customers, 83

D

Daily Standup Meeting, 45, 252
 Daily Standup Meetings, 3, 12, 25, 29, 31, 32, 38, 122, 142, 239, 246, 253, 255, 306, 401
 Data Flow Diagram
 Implement phase, 262
 Initiate phase, 198
 Decomposition, 224
 Definition of Done, 104
 Definition of Ready, 103, 190
 Delegating, 75
Delighters, 90
 Demonstrate and Validate Sprint, 4, 17, 19, 20, 25, 27, 29, 31, 32, 45, 47, 85, 112, 120, 127, 137, 209, 229, 246, 264, 266, 267, 269, 270, 273, 308, 325, 377
 inputs, 268
 outputs, 271
 tools, 269
 Demonstrate and Validate Sprint process, 266
 Demonstrations, 95
Dependencies, 189, 190, 227
 Dependency Determination, 186, 224
 Design Patterns, 248
 Develop Epic(s), 16, 18, 27, 29, 31, 34, 55, 57, 59, 60, 72, 83, 118, 142, 149, 150, 174, 175, 176, 178, 181, 209, 259, 300, 322, 377, 381, 394, 402
Develop Epic(s), 18, 150
 inputs, 175
 outputs, 181
 tools, 178
Development in Phases Contract, 177
 DevOps, 8
Directing, 75
Discretionary dependencies, 187
Dissatisfiers, 90
 Done Criteria, 190
 Done success rate, 275

E

Effective Deliverables, 4
Efficient development process, 4
Empirical Process Control, 12, 23, 25

Enhance, 8
 Epic(s), 181
 Estimate Tasks, 17, 18, 20, 29, 31, 44, 47, 57, 59, 60, 199, 200, 228, 229, 230, 232, 260, 304, 379, 385
 inputs, 229
 outputs, 232
 tools, 230
 Estimate Tasks process, 228
 Estimate User Stories, 17, 18, 20, 32, 47, 57, 59, 60, 83, 126, 186, 199, 200, 210, 211, 212, 228, 260, 303, 379, 385
 Estimated User Stories, 216, 218
Estimates, 189
 Estimation Criteria, 231
 Estimation effectiveness, 275
 Estimation Methods, 186
Exciters, 90
Expected Monetary Value (EMV), 140
 Expert Advice from HR, 165, 170
 Explorer – Shopper – Vacationer – Prisoner (ESVP), 274
External dependencies, 187
 External stakeholders, 108

F

Faster problem resolution, 4
Fist of Five, 213
 Flexibility, 119
 Focus Group Meetings, 178, 206
 Form Scrum Team, 16, 18, 57, 59, 60, 149, 150, 167, 168, 299, 300, 322, 380, 381
 inputs, 169
 outputs, 173
 tools, 170
 Forming, 73

G

Gap Analysis, 157

H

High trust environment, 4
High velocity, 4
 High-Level Estimates for Epics, 190
 Holiday Calendar, 193
 HR Theories, 73

I

Identified Business Stakeholder(s), 166

Identified Product Owner, 158
 Identified Risks, 181
 Identified Scrum Master, 166
 Identified Scrum Team, 173
Identify Scrum Master and Business Stakeholder(s), 18, 150, 162
 inputs, 164
 outputs, 166
 tools, 165
 Identify Tasks, 17, 18, 20, 31, 32, 44, 47, 57, 59, 60, 126, 199, 200, 222, 223, 260, 304, 381, 385
 inputs, 223
 outputs, 226
 tools, 224
 Identify Tasks process, 222
 Impediment Log, 246
 Implement, 8, 19, 239
Incentive and Penalty Contract, 177
Incremental Delivery Contract, 176
Indifferent, 90
 Initiate, 17, 149
 Initiate phase, 17, 85, 149, 150, 152, 296, 316, 321, 381
Innovative environment, 4
 Inspection, 27
 Integrating Change, 124
Internal dependencies, 187
 Internal Rate of Return (IRR), 88
 Internal stakeholders, 108
 Issues, 135
Iterative Development, 13, 24, 47, 120

J

JAD Sessions, 156
Joint Venture Contract, 177

K

Kanban, 8
 KanbanStudy.com, 9
 Kano analysis, 90

L

Laissez Faire, 75
 Large Project Communications Plan, 315
 Laws and Regulations, 176
 Leadership Styles, 75
 Length of Sprint, 125, 196, 219
 Lose/Lose, 75
 Lose/Win, 74

M

Mandatory dependencies, 187
 Market Study, 156
 Metrics and Measuring Techniques, 275
 Minimum Done Criteria, 106
 Minimum Marketable Features (MMF), 91
 Mitigated Risks, 249
Monopoly money, 89
 MoSCoW prioritization, 89, 185
 Motivated Scrum Team, 255
Motivation, 4

N

Net Present Value (NPV), 88
Non-core roles, 13, 53, 55
 Norming, 73
Number of stories, 236

O

OKRs, 8
 OKRStudy.com, 9
 Opportunity cost, 84
 Organization, 13, 51
 Organizational Deployment Methods, 284
 Organizational Resource Matrix, 164

P

Paired comparison, 185
Pareto analysis, 139
 PDCA Cycle, 110
 Peer feedback, 275
 People Availability and Commitment, 164
 People Costs, 171
 People Requirements, 164
 Performing, 73
 Personas, 181, 349
 Personnel Selection, 61
 Piloting Plan, 284
 Plan and Estimate, 18, 199
 Plan-Do-Check-Act (PDCA) Cycle, 110
 Plan-Do-Study-Act Cycle, 110
 Planning for value, 88
Portfolio, 65
 Portfolio Product Owner, 6, 58, 67, 71, 82, 96, 106, 112, 118, 128, 130, 145, 147, 156, 320, 321, 322, 323, 325, 327, 328, 331, 333, 336, 345, 347, 348, 351, 352, 354, 355, 372, 386

Portfolio Scrum Master, 346
Portfolio Scrum Master, 6, 68, 156, 321, 322, 323, 328, 331, 346
 Portfolio Scrum Master, 346
 Portfolios, 1, 23, 51, 62, 69, 72, 79, 99, 115, 128, 133, 145, 146, 149, 199, 239, 263, 279, 321
 Previous Project Information, 177
 Previous Sprint Velocity, 219
 Previous Work Day Experience, 252
Principles, 10, 23
 Prioritization Methods, 185
 Prioritized Product Backlog, 2, 15, 18, 19, 22, 25, 31, 34, 35, 37, 42, 44, 47, 57, 58, 59, 60, 62, 63, 65, 71, 79, 81, 83, 85, 89, 93, 96, 99, 100, 102, 103, 106, 109, 110, 112, 118, 119, 120, 122, 123, 124, 126, 127, 128, 130, 131, 137, 140, 141, 142, 145, 148, 149, 150, 181, 182, 184, 185, 186, 187, 189, 190, 193, 196, 197, 200, 203, 204, 206, 208, 209, 212, 214, 216, 219, 220, 237, 239, 240, 246, 247, 249, 256, 257, 258, 259, 261, 269, 271, 276, 290, 294, 298, 300, 301, 307, 309, 310, 311, 312, 313, 316, 318, 323, 325, 345, 371, 373, 374, 375, 376, 379, 384, 386, 387, 388, 394, 402
 Prioritized Product Backlog Review Meetings, 259
Prioritized Program or Portfolio Backlog Review Meetings, 347
 Prioritized Program or Portfolio Backlog Review Meetings outputs, 348
Probability Impact grid, 139
Probability trees, 138
 processes, 1, 8
Processes, 10
 Product, 1
 Product Backlog Items (PBIs), 127
 Product Backlog Review Meeting, 127
 Product Owner, 6, 13, 54, 57, 82
 Product Releases, 286
 Products, 8
Program, 65
 Program Product Owner, 6, 58, 63, 67, 71, 82, 83, 96, 106, 112, 118, 128, 130, 145, 147, 156, 320, 328, 331, 333, 337, 346, 347, 351, 352, 354, 387, 388
Program Scrum Master, 346
 Program Scrum Master, 6, 64, 68, 156, 346
 Program Scrum Master, 346
 Program Scrum Master, 352
 programs, 3, 1, 5, 14, 21, 23, 51, 65, 66, 67, 68, 69, 71, 79, 82, 83, 87, 96, 99, 110, 115, 128, 130, 133, 145, 147, 149, 156, 199, 239, 263, 279, 319, 320, 321, 328, 331, 332, 334, 337, 338, 339, 340, 341, 342, 343, 345, 348, 374, 386
 Programs, 128, 145

Progress to release/launch, 275
 Project, 2
 Project benefits, 84
 Project Budget, 158, 161
 Project Business Case, 155
 Project Charter, 158
 Project costs, 85
 Project reasoning, 84
Project timescales, 85
 Project Vision, 2
 Project Vision Meeting, 156
 Project Vision Statement, 2, 158
 projects, 3, 1, 2, 4, 5, 6, 10, 11, 14, 15, 16, 17, 19, 21, 23, 38, 42, 47, 51, 53, 54, 55, 56, 62, 63, 64, 65, 66, 67, 68, 69, 71, 74, 75, 76, 77, 79, 81, 82, 83, 85, 87, 88, 92, 96, 97, 99, 101, 103, 109, 110, 112, 115, 117, 121, 122, 125, 126, 128, 131, 133, 137, 145, 149, 156, 165, 166, 173, 176, 177, 186, 187, 189, 194, 196, 199, 212, 230, 236, 239, 247, 263, 279, 289, 293, 294, 296, 297, 299, 300, 301, 304, 309, 310, 311, 312, 313, 315, 316, 317, 318, 319, 320, 321, 322, 324, 326, 327, 328, 332, 333, 334, 337, 338, 339, 340, 341, 342, 343, 345, 347, 348, 349, 350, 351, 352, 353, 361, 362, 365, 373, 374, 379, 382, 386, 387, 389, 391, 392, 393, 401
 Proof of Concept, 155
 Proposed Non-Functional Items, 276
 Prototypes, 95

Q

Quality, 15, 99, 100
 Quality assurance, 110
 Quality control, 110
 Quality management, 108
 Quality planning, 108
 Questionnaires, 179

R

Recommended Scrum Guidance Body Improvements, 349
 Refactoring, 247
Refine Prioritized Product Backlog, 4, 17, 19, 20, 27, 29, 34, 42, 47, 57, 59, 60, 85, 103, 123, 124, 125, 127, 137, 140, 141, 214, 240, 256, 257, 258, 259, 307, 325, 387, 390
 inputs, 257
 outputs, 261
 tools, 259
Refine Prioritized Product Backlog, 4
 Refine Prioritized Product Backlog process, 256

Refined Prioritized Product Backlog, 197
 Rejected User Stories, 258, 271
 Relative prioritization ranking, 90
 Release, 19, 20, 279
 Release Planning Schedule, 18, 25, 57, 59, 149, 150, 191, 193, 194, 196, 246, 258, 261, 268, 271, 284, 301, 352, 374, 383, 390, 391
 Release Planning Sessions, 194
 Release Prioritization Methods, 194
 Release Readiness Plan, 314
 Requirements churn, 16
 Resource Costing, 166
 Resource Requirements, 170
Retrospect Program or Portfolio Meetings, 355
 Retrospect Program or Portfolio Releases inputs, 354
 Retrospect Program or Portfolio Releases outputs, 356
 Retrospect Program or Portfolio Releases process, 354
 Retrospect Program or Portfolio Releases tools, 355
Retrospect Release, 4, 17, 19, 20, 29, 58, 59, 60, 85, 120, 137, 280, 287, 288, 289, 290, 311, 327, 355, 391, 392
 inputs, 288
 Retrospect Release Meeting, 289
 Retrospect Release outputs, 290
 Retrospect Release process, 287
 Retrospect Release tools, 289
Retrospect Sprint, 3, 4, 12, 17, 19, 20, 23, 25, 29, 34, 35, 45, 55, 59, 60, 83, 85, 120, 121, 137, 258, 264, 272, 273, 274, 275, 276, 277, 289, 290, 308, 309, 317, 325, 371, 379, 388, 392, 395, 397
 inputs, 273
 outputs, 276
 tools, 274
 Retrospect Sprint Log(s), 276
Retrospect Sprint Meeting, 45, 274
 Retrospect Sprint process, 272
 Return on Investment (ROI), 87
 Review and Retrospect, 19, 263
 Review and Update Scrum Guidance Body inputs, 341
 Review and Update Scrum Guidance Body outputs, 343
 Review and Update Scrum Guidance Body process, 341
 Review and Update Scrum Guidance Body tools, 342
 Review feedback ratings, 275
 Risk, 16, 133, 135, 189
 Risk appetite, 136
 Risk assessment, 138
 Risk attitude, 136
 Risk averse, 136
Risk Breakdown Structure (RBS), 137
 Risk Burndown Chart, 143
Risk checklists, 137
 Risk communication, 142

Risk identification, 137
 Risk management, 137
 procedure, 137
Risk meeting, 138
 Risk mitigation, 141
 Risk neutral, 136
 Risk prioritization, 140
 Risk prompt lists, 137
 Risk seeking, 136
 Risk threshold, 136
 Risk tolerance, 136
 Risk-based spike, 142
 Risks, 84
 Roles Guide, 11, 23, 24, 51, 52, 63, 64, 67, 68, 79, 80, 99,
 115, 116, 133, 134

S

Satisfiers, 90
SBOK® Guide, 1, 2, 3, 1, 2, 3, 6, 10, 11, 12, 16, 23, 51, 63,
 64, 67, 68, 79, 99, 115, 133, 149, 150, 199, 200, 239,
 263, 279, 293, 319, 386
 Scalability of Scrum, 5
 Scaled Scrum Master Certified (SSMC™), 6
 Scaled Scrum Product Owner Certified (SSPOC™), 6
 Scope, 100
 Scrum, 2, 8
 Scrum aspects, 1, 11, 12, 13, 293, 319
 Scrum Aspects, 13
 Scrum Core Team, 6, 121, 175
 Scrum Developer Certified (SDC®), 6
 Scrum for Large Projects, 5, 11, 17, 21, 63, 64, 149, 199,
 239, 263, 279
 Scrum for the Enterprise, 5, 11, 17, 21, 66, 67, 68, 149,
 199, 239, 263, 279, 328, 330
 Scrum Guidance Body, 1, 6, 14, 15, 19, 29, 31, 38, 45, 56,
 62, 66, 68, 71, 82, 87, 88, 96, 101, 103, 105, 106, 108,
 109, 110, 112, 117, 122, 130, 142, 147, 150, 156, 165,
 170, 176, 177, 179, 182, 184, 187, 190, 193, 205, 207,
 212, 219, 230, 247, 248, 259, 264, 269, 270, 272, 274,
 275, 276, 277, 284, 289, 290, 294, 312, 315, 316, 317,
 318, 320, 328, 334, 337, 339, 340, 341, 342, 343, 346,
 347, 349, 354, 355, 356, 371, 383, 389, 390, 395, 401
 Scrum Guidance Body (SGB), 1, 14, 56, 83, 87, 156, 342,
 395
 Scrum Guidance Body Expertise, 179, 187, 207, 248, 270,
 275, 289
 Scrum Guidance Body Recommendations, 156, 177, 184,
 193, 205, 212, 219, 230, 247, 259, 269, 274, 284, 289
 Scrum Master, 6, 11, 13, 14, 16, 18, 19, 21, 23, 24, 31,
 34, 35, 44, 47, 51, 52, 54, 55, 56, 57, 59, 62, 63, 64,

68, 71, 72, 73, 74, 75, 77, 79, 80, 83, 96, 99, 106, 112,
 115, 116, 121, 126, 127, 130, 133, 134, 142, 147, 149,
 150, 156, 158, 162, 163, 164, 165, 166, 167, 169, 171,
 173, 175, 178, 179, 186, 199, 200, 209, 210, 239, 240,
 243, 246, 251, 252, 253, 255, 259, 264, 272, 273, 274,
 276, 277, 284, 288, 289, 293, 298, 299, 301, 302, 305,
 306, 308, 309, 310, 311, 312, 313, 315, 316, 317, 321,
 322, 323, 327, 328, 331, 333, 337, 348, 351, 352, 354,
 355, 373, 379, 380, 381, 383, 386, 387, 388, 392, 395,
 396, 399, 400, 401, 402
 Scrum Master Certified (SMC®), 6
 Scrum of Scrums (SoS) Meeting, 317, 338
 Scrum principles, 1, 11, 12, 20, 23, 24, 59, 148, 171, 259,
 293, 319
 Scrum processes, 1, 4, 6, 10, 11, 13, 16, 21, 29, 37, 44,
 47, 54, 57, 59, 60, 62, 68, 71, 74, 83, 85, 89, 112, 115,
 137, 149, 166, 176, 293, 296, 319, 321, 328, 331, 395
 Scrum Processes, 16
 Scrum Product Owner Certified (SPOC®), 6
 Scrum project, 1, 2, 11, 13, 14, 15, 16, 21, 44, 51, 53, 55,
 56, 63, 73, 74, 75, 92, 95, 100, 108, 113, 115, 117,
 123, 126, 137, 144, 149, 157, 166, 170, 173, 196, 237,
 239, 259, 264, 266, 279, 296, 298, 300, 301, 302, 303,
 304, 305, 306, 307, 308, 309, 310, 314, 316, 318, 320,
 375, 384
 Scrum Project Tool, 37, 38, 171, 180, 187, 194, 207, 219,
 221, 224, 226, 231, 236, 246, 248, 253, 259, 260, 275,
 285, 290, 299, 300, 301, 303, 304, 306, 307, 313, 315,
 316, 318, 339, 348, 352
 Scrum projects, 6
 Scrum Team, 6, 13, 54, 60, 83
 Size, 61
 Scrum Team Lessons Learned, 276
 Scrum Team Selection, 214, 270
 Scrum Team Selection Criteria, 170
 Scrum Teams, 5
 Scrumboard, 37, 226
 Scrumboard, 220
 Scrumboard, 244
 SCRUMstudy Agile Master Certified (SAMC™), 6
 Selection Criteria, 165
Self-organization, 12, 23, 31, 123
Senior Management, 121
 services, 8
Setup, 8
 Ship Deliverables, 4, 17, 19, 58, 85, 280, 282, 283, 284,
 310, 318, 327, 397
 inputs, 283
 outputs, 286
 tools, 284
 Ship Deliverables process, 282

Simple schemes, 89
 Simulations, 95
 Skills Requirement Matrix, 165
 Speed Boat, 275
Sponsor, 56, 83
 sponsors, 6, 13, 16, 56, 71, 117, 120, 165, 166, 170, 333, 361
 Sprint, 3, 44
 Sprint Backlog, 18, 19, 25, 37, 44, 47, 57, 59, 60, 103, 109, 119, 127, 144, 199, 200, 217, 219, 220, 226, 232, 233, 235, 237, 240, 243, 244, 245, 247, 268, 304, 375, 398, 400, 401
 Sprint Burndown or Burnup Chart, 92, 237
 Sprint Deliverables, 248
Sprint Planning Meeting, 44, 235
 Sprint Planning Meetings, 12, 20, 25, 38, 214, 219, 224, 226, 230, 232, 235, 401
 Sprint Planning Meetings*, 230
Sprint Review Meeting, 45, 269
 Sprint Review Meetings, 12, 25, 38, 246, 269, 270, 308
 Sprint Tracking Metrics, 236
 Sprint Tracking Tools, 235
 Stability, 119
 Stakeholder Meeting, 2
 Storming, 73
 Story Mapping, 91
 supporting leadership, 31
 Supporting Leadership, 75, 76
 Supporting Services, 56
Sustainable pace, 4, 109
 SWOT Analysis, 157

T

Target Customers for Release, 196
 Task List, 226
 Task Planning Meetings, 224
 Task-oriented, 76
 Team Building Plan, 173
 Team Calendar, 235
 Team Expertise, 247
 Team morale ratings, 275
 Team velocity, 275
 Technical debt, 109
Theory X, 77
Theory Y, 77
Theory Z, 77
 Three Daily Questions, 253
Time-boxing, 12, 23, 44, 122
Traditional Project Management, 1, 22, 24, 30, 49, 52, 72, 80, 97, 99, 113, 115, 131, 133, 148

Training, 166, 171
Transparency, 4, 25
 Trial Project, 155
 Tuckman's Model of Group Dynamics, 73

U

Unapproved Change Requests, 117, 176
Uncertainty, 189
 Update Sprint Backlog, 17, 18, 20, 32, 44, 47, 57, 59, 60, 127, 199, 200, 233, 234, 304, 314, 385, 398, 401
 inputs, 234
 outputs, 237
 tools, 235
Updated Dependencies, 340
Updated Impediment Logs, 340
 Updated or Refined Personas, 209
 Updated Prioritized Product Backlog, 209, 261
 Updated Release Planning Schedule, 261
 Updated Scrum Guidance Body Recommendations, 277
 Updated Scrumboard, 232, 248
 Updated Sprint Backlog process, 233
 Updated Task List, 232
 User Group Meetings, 178
 User or Customer Interviews, 179
 User Stories, 2, 3, 18, 19, 25, 31, 32, 38, 42, 44, 45, 47, 54, 57, 58, 59, 60, 71, 83, 85, 89, 90, 91, 92, 93, 96, 100, 102, 103, 104, 106, 108, 109, 110, 112, 113, 118, 123, 125, 126, 127, 130, 141, 142, 148, 173, 178, 181, 184, 185, 186, 189, 190, 196, 197, 199, 200, 203, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 216, 217, 219, 220, 222, 223, 224, 226, 227, 228, 229, 230, 231, 232, 235, 236, 237, 239, 240, 245, 246, 247, 248, 249, 256, 259, 261, 264, 266, 269, 271, 275, 276, 279, 280, 282, 283, 290, 301, 302, 303, 304, 305, 306, 308, 309, 311, 313, 314, 316, 317, 318, 341, 345, 346, 347, 348, 351, 353, 371, 372, 374, 375, 376, 377, 378, 379, 381, 383, 384, 385, 386, 390, 400, 401, 402
 User Stories Acceptance Criteria, 211
 User Story Acceptance Criteria, 102, 209, 223
 User Story Acceptance/Rejection, 269
 User Story Workshops, 178
 User Story Writing Expertise, 206
 users, 6, 13, 15, 16, 55, 56, 57, 71, 81, 88, 100, 106, 112, 113, 117, 120, 137, 165, 166, 178, 179, 181, 187, 196, 206, 284, 333, 361, 364, 380, 384, 385, 399, 402
Users, 56, 83
Utility Function, 136

V

Value, 189

Value Stream Mapping, 89

Value-based Prioritization, 12, 23, 42

Value-driven Delivery, 15, 79, 81

Velocity, 236

Vendors, 14, 56

Video Conferencing, 253

VMedu, 9

VMedu Inc, 9

VMedu.com, 9

Voice of the Customer, 58

W

War Room, 253

Wideband Delphi, 213

Win/Lose, 75

Win/Win, 74

Working Deliverables, 286

Working Deliverables Agreement, 286

The Essential Guide to Successfully Deliver Projects using Scrum

A Guide to the Scrum Body of Knowledge (SBOK® Guide) provides comprehensive guidelines for the successful implementation of Scrum—the most popular Agile product development and project delivery approach. Defined in the SBOK® Guide as a flexible framework, Scrum can be applied to portfolios, programs, or projects of any size or complexity across industries to deliver products, services, or other results.

This Fifth Edition is based on the collective knowledge gained from thousands of projects across diverse organizations and industries. It reflects contributions from a large number of experts in Scrum and project delivery. Feedback from the global Scrum community played a vital role in shaping improvements and additions, making the SBOK® Guide a truly collaborative effort.

Unlike other Scrum references, the SBOK® Guide is available for free on [Scrumstudy.com](https://www.scrumstudy.com), along with free certifications, webinars, videos, and study guides. It is ideal for professionals seeking a foundational understanding of Business Analysis or exploring careers in related fields. The Guide addresses real-life challenges faced by Scrum practitioners and explains how to solve them using modern tools and Artificial Intelligence (AI).

Designed to be accessible and engaging, the SBOK® Guide follows the 80-20 rule—80% of key concepts can be learned by reading just 20% of the content, with the remainder available for deeper reference. It is more readable than most Scrum books, which are often either too simplistic or overly detailed.

The SBOK® Guide serves as a reference for both experienced practitioners and those with no prior knowledge of Scrum or project delivery methods. Organized for easy navigation, the SBOK® Guide aims to inform, support, and inspire all readers through its rich, collaborative content.

