



Skolkovo Institute of Science and Technology

Skolkovo Institute of Science and Technology

A DECISION SUPPORT SYSTEM FOR AGILE DEVELOPMENT
OF COMPLEX HARDWARE SYSTEMS

Doctoral Thesis

by

NICOLA GARZANITI

DOCTORAL PROGRAM IN ENGINEERING SYSTEMS

Supervisor

Associate Professor Alessandro Golkar

Co-Supervisor

Professor Clément Fortin

Moscow - 2021

© Nicola Garzaniti 2021

This page intentionally left blank

“faber est suae quisque fortunae”

Appius Claudius Caecus

This page intentionally left blank

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Candidate (Nicola Garzaniti)

Supervisor (Prof. Alessandro Golkar)

Co-Supervisor (Prof. Clément Fortin)

This page intentionally left blank

Abstract

This thesis presents a decision support system for Agile development of complex hardware systems. Agile is a general term covering different yet somehow similar methods that have evolved within the software community and find their foundations in the *Manifesto for Agile Software Development*. Over the last two decades, Agile approaches have gradually spread from the software domain to the development of *physical products*. With this spread over different engineering domains, the product development community started discussing the actual viability of implementing Agile in the context of hardware systems development.

This work aims to contribute to the current debate on the topic by developing a framework to the viability of implementing Agile vs. Traditional or Hybrid product development process. Thus, provide structuring and planning of the resulting development process. It includes an analytical approach to managing development activities and consists of three macroblocks: structuring, simulating, and planning. The framework is implemented in an integrated tool.

Within the framework development, several challenges entailed by engineering teams adopting Agile have been addressed, such as the Sprint planning, the Minimum Viable Product in the context of hardware systems, the question of the procurement and manufacturing, and the coordination aspect with potential development partners not implementing agile.

Then the framework has been applied to a set of case studies to verify its capabilities in different industry contexts. In the case studies, it has been considered the development of a space system as well as the development of a consumer product. The analysis conducted within the case studies also provided valuable insights on the contextual factors enabling Agile implementation.

System modularity, supplier selection, team composition, location, and synchronization play a key role in the feasibility of Agile/Hybrid-Agile approach implementation. Modularity is crucial for the effective implementation of Agile because it drives both the cost and time of iterations. Typically, high modularity allows for more and faster iterations. Team composition, location, and synchronization may enable or hamper the possibility of iterative, incremental development. As demonstrated in the first case study, co-located cross-functional teams represent an effective setting for implementing Agile. While, as discussed in the second case study, highly dispersed functional teams are not ideal for Agile adoption. Supplier selection and supply chain management, in general, driving the time and the cost of procuring/manufacturing physical components, affect the feasibility of iterative, incremental development. Long lead-time items or expensive custom components hinder effective Agile implementation. A misguided combination of all those factors may completely overturn potential advantages in cost or schedule brought by Agile.

Publications

Journal Articles

- **N. Garzaniti**, Z. Tekic, D. Kukulj, A. Golkar, “Review of Technology Trends in New Space Missions using a Patent Analytics Approach”, *Progress in Aerospace Sciences*, 2021, vol. 125, p. 100727, Aug. 2021, doi: 10.1016/J.PAEROSCI.2021.100727
- **N. Garzaniti**, A. Golkar and P. Maggiore, “Additive Manufacturing Evaluation Tool for Design Studies,” in *IEEE Systems Journal*, vol. 14, no. 3, pp. 4382-4393, Sept. 2020, doi: 10.1109/JSYST.2019.2939906.

Conference Papers

- **N. Garzaniti** and A. Golkar, “Performance Assessment of Agile Hardware Co-development Process,” in *2020 IEEE International Symposium on Systems Engineering (ISSE)*, 2020, pp 1-6, doi: 10.1109/ISSE49799.2020.9272209.
- A. Golkar, S. Briatore, and **N. Garzaniti**, “Lessons learnt in the deployment of scrum in space hardware development projects,” in *Proceedings of the International Astronautical Congress (IAC)*, 2019, IAC-19_D1_5_6_x51187.
- **N. Garzaniti**, C. Fortin, and A. Golkar, “Toward a Hybrid Agile Product Development Process,” in *IFIP Advances in Information and Communication Technology (PLM Conference 2019)*, 2019, vol. 565 IFIP, pp. 191–200, doi: 10.1007/978-3-030-42250-9_18.
- **N. Garzaniti**, S. Briatore, C. Fortin, and A. Golkar, “Effectiveness of the Scrum Methodology for Agile Development of Space Hardware,” in *IEEE Aerospace Conference Proceedings*, 2019, vol. 2019-March, pp. 1–8, doi: 10.1109/AERO.2019.8741892.
- A. Camps et al., “FSSCAT, the 2017 copernicus masters’ ‘ESA sentinel small satellite challenge’ winner: A federated polar and soil moisture tandem mission based on 6U Cubesats,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2018, vol. 2018-July, pp. 8285–8287, doi: 10.1109/IGARSS.2018.8518405.
- **N. Garzaniti**, A. Golkar, and C. Fortin, “Optimisation of multi-part 3D printing build strategies for lean product and process development,” in *IFIP Advances in Information and Communication Technology (PLM Conference 2018)*, 2018, vol. 540, pp. 488–497, doi: 10.1007/978-3-030-01614-2_45.

Other publications

Journal Articles

- J. A. Ruiz-de-Azua, **N. Garzaniti**, A. Golkar, A. Calveras, and A. Camps, “Towards Federated Satellite Systems and Internet of Satellites:The Federation Deployment Control Protocol,” *Remote Sensing*, vol. 13, no. 5, p. 982-1003, March 2021, doi: doi.org/10.3390/rs13050982.
- K. Latyshev, **N. Garzaniti**, E. Crawley, and A. Golkar, “Lunar human landing system architecture tradespace modeling,” *Acta Astronautica*, vol. 181, Pages 352-361, April 2021, doi: 10.1016/j.actaastro.2021.01.015.

Conference Papers

- A. Camps et al., “FSSCAT Mission description and first scientific results of the FMPL-2 onboard 3CAT-5/A,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2021, vol. 2021-July
- K. Latyshev, **N. Garzaniti**, A. Golkar, and E. F. Crawley, “Technology Roadmap for Future Human Landing Systems,” in *ASCEND 2020*, 2020, pp 1-18 doi: 10.2514/6.2020-4229.
- K. Osipova, **N. Garzaniti**, S. Briatore, and A. Golkar, “Systems architecture study of satellite constellations for internet of things connectivity”, in *Proceedings of the International Astronautical Congress (IAC)*, 2020, IAC-20,D1,2,6,x58770
- J. A. Ruiz-De-Azua et al., “Demonstration of the Federated Satellite Systems Concept for Future Earth Observation Satellite Missions,” in *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020, pp. 3574–3577, doi: 10.1109/IGARSS39084.2020.9323066.
- A. Golkar and **N. Garzaniti**, “Model based systems engineering approach to technology roadmapping,” in *Proceedings of the 2020 Summer Simulation Conference (SummerSim)*, 2020, pp. 1–12, doi: 10.5555/3427510.3427536.
- J. A. Ruiz-de-Azua et al., “Proof-of-Concept of a Federated Satellite System Between Two 6-Unit CubeSats for Distributed Earth Observation Satellite Systems,” *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019, vol. 2019-July, pp. 8871–8874, doi: 10.1109/igarss.2019.8900099.
- S. Briatore, **N. Garzaniti**, and A. Golkar, “Towards the Internet for Space: bringing cloud computing to space systems,” in *36th International Satellite Communications Systems Conference (ICSSC2018)*, 2018, pp. 38 - 42, doi: 10.1049/cp.2018.1719.

Table of Contents

Abstract	I
Publications	III
Table of Contents	V
List of Symbols and Abbreviations	IX
List of figures	XI
List of tables	XV
1 Introduction	1
1.1 Motivation.....	2
1.2 Scope of the research	5
1.3 Thesis objectives	7
1.4 Research Questions and Hypothesis	9
1.5 Thesis Structure.....	10
1.6 Research methodology.....	11
2 Background	13
2.1 Agile methodology and operational frameworks.....	13
2.2 Scrum: an overview	14
2.2.1 Roles artifacts and events	16
2.2.2 Iterations	19
2.2.3 Minimum Viable Product definition.....	21
2.2.4 Sprint planning	22
2.3 Product development processes in the literature.....	24
2.3.1 The State-Gate process	25

Table of Contents

2.3.2	Agile for hardware.....	27
2.3.3	Hybrid-Agile approaches.....	28
2.4	Catalog of gaps in the literature	29
2.5	An industry perspective	31
2.5.1	New Space and the question of the product development.....	31
2.5.2	Mission overview	32
2.5.3	Using “traditional” Scrum	34
2.5.4	Support tools.....	37
2.5.5	Scrum implementation.....	38
2.5.6	MVP: Customer feedback and technology de-risking.....	41
2.5.7	Hardware vs. Software: the question of the procurement.....	42
2.5.8	Testing and the question of quality assurance	43
2.5.9	Reactivity to unforeseen design changes.....	44
2.5.10	The human factor.....	44
2.5.11	Agile manifesto vs. complex hardware systems.....	46
2.5.12	Catalog of gaps, challenges	46
2.6	Summary of literature review and industry evaluation	48
3	A decision support system.....	51
3.1	Structuring.....	54
3.1.1	Product backlog architecture	54
3.1.2	Scoring system: time and cost estimates	56
3.1.3	Resource availability and disciplines involved.....	58
3.1.4	Hybrid-Agile in multiparty consortia	59
3.1.5	Agile implementation viability.....	60

3.2	Simulation	66
3.3	Planning	70
3.3.1	Continuous process tracking and improvement.....	74
3.3.2	MVP	76
3.4	Illustrative case	78
3.4.1	Process Structuring	78
3.4.2	Simulation.....	81
3.4.3	Planning.....	83
3.4.4	Comparison with traditional project management approaches.....	87
3.5	Validity of the decision support system.....	91
3.5.1	Process model validity.....	91
3.5.2	Data model validity	93
3.5.3	Graphical User Interface validity	95
3.5.4	DSS general validity.....	96
3.5.5	Face validity	99
3.6	Deployment in development projects.....	100
4	Cast study: New Space mission payload.....	105
4.1	General case study data.....	106
4.2	Organizational structure.....	108
4.3	Agile in the development process, motivation for Agile adoption	109
4.4	CURSIVE deployment.....	110
4.4.1	Process Structuring	110
4.4.2	Simulation.....	115
4.4.3	Planning.....	118

Table of Contents

4.5	Process Insights.....	121
4.6	Summary and interim conclusion	124
5	Cast study: A consumer product	129
5.1	General case study data.....	130
5.2	Organizational structure.....	132
5.3	Agile in the development process, motivation for Agile adoption	133
5.4	CURSIVE deployment.....	133
5.4.1	Process Structuring.....	133
5.4.2	Simulation.....	138
5.4.3	Planning.....	140
5.5	Process insights	141
5.6	Summary and interim conclusion	145
6	Conclusion.....	151
6.1	Thesis summary	151
6.2	Thesis Contributions	154
6.3	Limitations and Future Work.....	157
	References	159

List of Symbols and Abbreviations

AIT	Assembly, Integration, and Testing
CAPEX	Capital Expenditure
CPM	Critical Path Method
DAR	design-analyze-redesign
DMTR	design-manufacture-test-redesign
DRM	Design Research Methodology
DSM	Design Structure Matrix
DSS	decision support system
FCM	Fuzzy C-means
FPGA	Field Programmable Gate Array
FTE	Full-time equivalent
INCOSE	International Council on Systems Engineering
MBSE	Model-Based System Engineering
MRD	Mission Requirements Document
MVP	Minimum Viable Product,
NPD	new product development
O-ISL	Optical Inter-Satellite Link
OPEX	Operating Expense
PCB	Printed Circuit Board
PDF	Probability Density Functions
PDP	Product Development Process
PDR	Preliminary Design Review

List of Symbols and Abbreviations

PERT	Project Evaluation Review Technique
RCPS	Resource-Constrained Project Scheduling
RF	Radio frequency
SMEs	Small and Medium-sized Enterprises
SRD	System Requirements Document
TRL	Technology Readiness Level
V&V	Verification and Validation
VUCA	Uncertain, Volatile, Complex, and Ambiguous

List of figures

Figure 1. Expected value of Agile hardware development, data source (Schmidt et al., 2018b).....	3
Figure 2. Currently available Agile frameworks and practices	4
Figure 3. Relation to fields of knowledge	6
Figure 4. Life cycle stages from ISO/IEC 15288 (ISO/IEC JTC 1/SC 7, 2015).....	7
Figure 5. DRM framework adapted from (Blessing & Chakrabarti, 2009).....	12
Figure 6. Graphical representation of Scrum methodology, sources (Garzaniti et al., 2019b)	15
Figure 7. Sprint burndown chart.....	22
Figure 8. Map of the Product development literature.....	24
Figure 9. A representation of the Stage-Gate model	26
Figure 10. Use case Project Life Cycle (design, development, and integrations stages)	33
Figure 11. Functional block diagram of the payload, source (Garzaniti et al., 2019a)	34
Figure 12. User stories with scores and preliminary tasks prioritization on a physical Kanboard.	36
Figure 13. Example of Jira user interface.....	37
Figure 14. Sprint length.....	38
Figure 15. Planned story points per day for each Sprint.....	39
Figure 16. Percentage of work completed per Sprint	41
Figure 17. Example of workflow to realize an MVP, source (Garzaniti et al., 2019b).....	42
Figure 18. Qualification model vibration test (left), proto-flight model TVAC test (right).....	43
Figure 19. CURSIVE macroblocks	52
Figure 20. Workflow of proposed framework.....	53
Figure 21. Notional example of DSM <i>product backlog architecture</i>	55

Figure 22. Complementary information provided within the product backlog architecture	58
Figure 23. Hybrid product development process architecture.....	59
Figure 24. Survey data on sprint length, committed user stories and team composition	63
Figure 25. Agile Implementation Viability Chart	65
Figure 26. Notional example of activities network	66
Figure 27. Schematic representation of simulation process	67
Figure 28. Output dashboard of scenario analysis.....	69
Figure 29. Output dashboard of the planning phase.....	72
Figure 30. Continuous process tracking and improvement	74
Figure 31. Product backlog architecture updates.....	75
Figure 32. MVP mapping: V&V vs. Sprint length.....	77
Figure 33. Artistic representation of an X-Band transmitter, source (EnduroSat, 2021)	78
Figure 34. Illustrative case – product backlog architecture	79
Figure 35 Illustrative case	80
Figure 36. Illustrative case – Agile viability indexes	80
Figure 37. Mean and variance over the number of simulations	81
Figure 38. Illustrative case - Simulation output	82
Figure 39. Illustrative case - Gantt chart	83
Figure 40. Illustrative case - Sprints Backlog.....	84
Figure 41. Illustrative case - Sprint sequence and costs	85
Figure 42. Illustrative case - Tasks Time and Cost Breakdown	86
Figure 43 PERT (activities on nodes), CPM in red.....	89
Figure 44. Time distribution for RCPS	90
Figure 45. Framework for evaluating DSS validity	92

Figure 46. DSS general validity elements	96
Figure 47. General case study data collection format (1).....	102
Figure 48. General case study data collection format (2).....	103
Figure 49. Case A – Organization of the development team.....	108
Figure 50. Input DSM for development of the optical telecommunication payload	111
Figure 51. Case study A - Complementary information	112
Figure 52. Case A – Agile Implementation viability.....	113
Figure 53. Mean and variance over the number of simulations	115
Figure 54. Case A - Simulation output.....	116
Figure 55. Schedule Target vs time distribution of simulated scenarios.....	117
Figure 56. Case A- Overall Schedule.	117
Figure 57. Gantt chart of the baseline scenario meeting budget and time constraints	118
Figure 58. Case A - Sprint backlog	119
Figure 59. Sprint sequence and cost Breakdown.....	120
Figure 60. Case A - Tasks Cost Breakdown.....	121
Figure 61. Case A - Tasks Time Breakdown.....	122
Figure 62. Degree of modularity	127
Figure 63. Case B – Organization of the development teams	132
Figure 64. Input DSM for development of the household appliance	134
Figure 65. Case study B - Complementary information.....	135
Figure 66. Case B – Agile Implementation viability.....	136
Figure 67. Product DSM of the household appliance.....	137
Figure 68. Mean and variance over the number of simulations	138
Figure 69. Case B - Simulation output	139

List of figures

Figure 70. Case B- Gantt chart of the baseline scenario meeting time constraints	140
Figure 71. Case B - Comparison of different scenarios.....	141
Figure 72. Case B - Tasks Time Breakdown.....	141
Figure 73. Summary of Process insights	142
Figure 74. Suggested optimization	142
Figure 75. Case B – Simulation output of optimized process structure	143
Figure 76. Recommended strategy	144
Figure 77. Degree of modularity	149

List of tables

Table 1. Scrum framework elements.....	16
Table 2. Scrum Events.....	18
Table 3. Agile manifesto vs. hardware systems development.....	46
Table 4. Catalog of gaps, challenges	47
Table 5 Taxonomy of MVP.....	77
Table 6: Map of MVPs related to the MVP taxonomy.....	85
Table 7 PERT data (durations are presented in days).....	88
Table 8. Running time	98
Table 9. Data collection methods	101
Table 10. Map of MVPs related to the MVP taxonomy.....	123

This page intentionally left blank

“In today’s fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential. [...] This new emphasis on speed and flexibility calls for a different approach for managing new product development”.

(Takeuchi & Nonaka, 1986)

Chapter 1

Introduction

High quality, low cost, and unique selling propositions are not enough to excel in today’s competitive market. Also speed and flexibility are essential in the development of new products (Takeuchi & Nonaka, 1986).

In a 1986 edition of Harvard Business Review, Takeuchi and Nonaka published a seminal research work stating that the rules of the game in new product development were changing, highlighting the need for increased speed and flexibility. After thirty-five years from that publication, the statement seems more relevant than ever.

Organizations are under constant pressure to create and sustain their competitive advantage. Since commercial markets are moving faster and faster and product life cycles are getting shorter, time is becoming a strategic source of competitive advantage. Furthermore, the product development process is continuously bullied by customer behavior changes, technological breakthroughs, competitors’ initiatives, disruptions in the supply chain, and internal organizational contingencies.

These are the conditions project managers and development teams currently face at the beginning of a new project and are the starting point of this research work.

1.1 Motivation

Companies in all industries are increasingly facing the challenge of implementing development projects under uncertain, volatile, complex, and ambiguous (VUCA) conditions. A need for new development methodologies to deal with such circumstances is generally acknowledged.

This need is driving companies to focus on streamlining the product development processes. In particular, there is a growing interest in Agile methodologies. Agile is a general term covering diverse yet somehow similar methods that have evolved within the software community and find their foundations in the *Manifesto for Agile Software Development* (Beck et al., 2001).

This year, the Agile Manifesto is celebrating its twentieth anniversary, and over the last two decades, Agile approaches have gradually spread from the software domain to the development of *physical products*. The increasing interest toward Agile has been fired by the growing number of published success stories, which typically report that Agile has shortened the time-to-market, improved the development process efficiency, and enhanced the product fit to the customer needs (Recker et al., 2017).

Although some skepticism exists whether agility can actually realize its potential in the domain of physical product, naturally characterized by different constraints and contractions compared to software, the euphoria seems to outstrip the skepticism in many companies. Scientific literature, as well as industry reports and surveys, provide hints that a considerable number of organizations have started adopting Agile methods for physical product development in real industrial settings (Digital.ai & VersionOne Inc., 2020; KPMG, 2019).

However, while organizations started the Agile transition, there is still little evidence of consistent delivery of the purported benefits across different engineering projects. To date, there is

much anecdotal evidence on the improvements Agile provides to development process performance, but empirical proofs are mixed and hard to find (Serrador & Pinto, 2015). The question of whether Agile can truly fulfill its promise in complex hardware developments has not yet found a conclusive answer.

The lack of quantitative analyses and empirical evidence comes together with a different understanding of Agile development between academia and industry. On the one hand, academia has focused on the teams' dynamics and the social aspects of the development process (Chuang et al., 2014), which we will call *soft aspects*. On the other hand, industry interest relates to reducing development time and cost (which we will call *hard aspects*). According to a sector study on the Agile Development of Physical Products (Schmidt et al., 2018b), 73% of the interviewees expect Agile development to reduce the time-to-market, 76% expect to increase project effectiveness (Figure 1). Of course, the book title “Scrum: The art of doing twice the work in half the time” (Sutherland, 2014), one of the most famous references among corporate-level managers and consultants, certainly does not help mitigate the hype and overcome the misinterpretation.

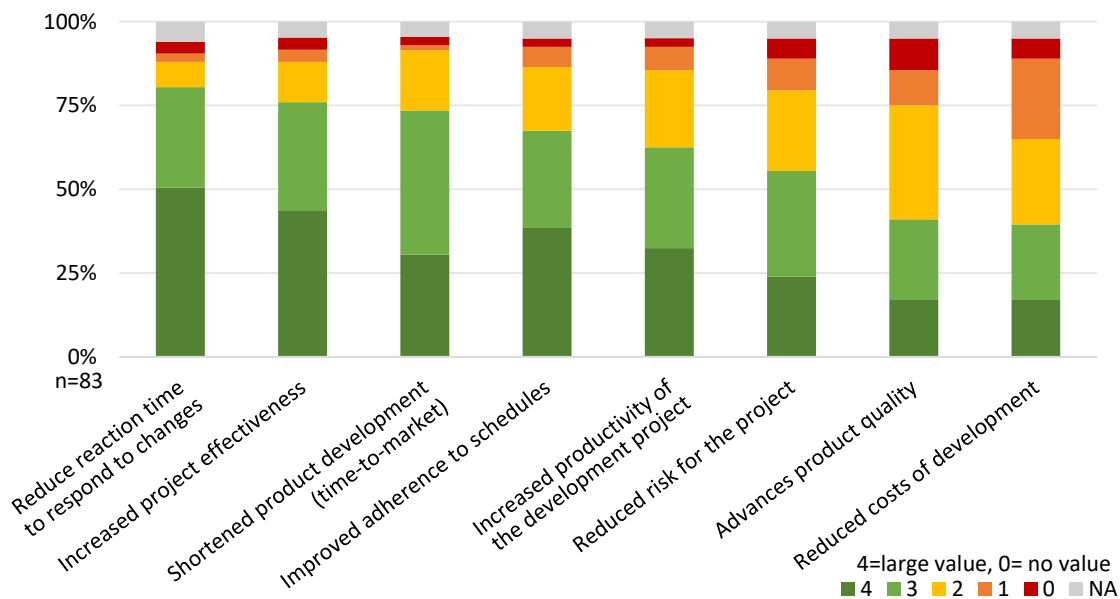


Figure 1. Expected value of Agile hardware development, data source (Schmidt et al., 2018b)

The unclear effect of Agile on project performance, combined with the need for new methods to develop projects under VUCA conditions, has led to a proliferation of Agile frameworks and practices. At the time of writing this thesis, we have identified forty-five different operational frameworks and practices (Figure 2). Each of them addresses and tries to solve specific issues pertaining to the development process (e.g., issue: continuous process improvement and incorporation of lessons learned within the ongoing project; Practice: Sprint retrospective; Framework: Scrum), and each of them can serve well some situation while may not suit some others.

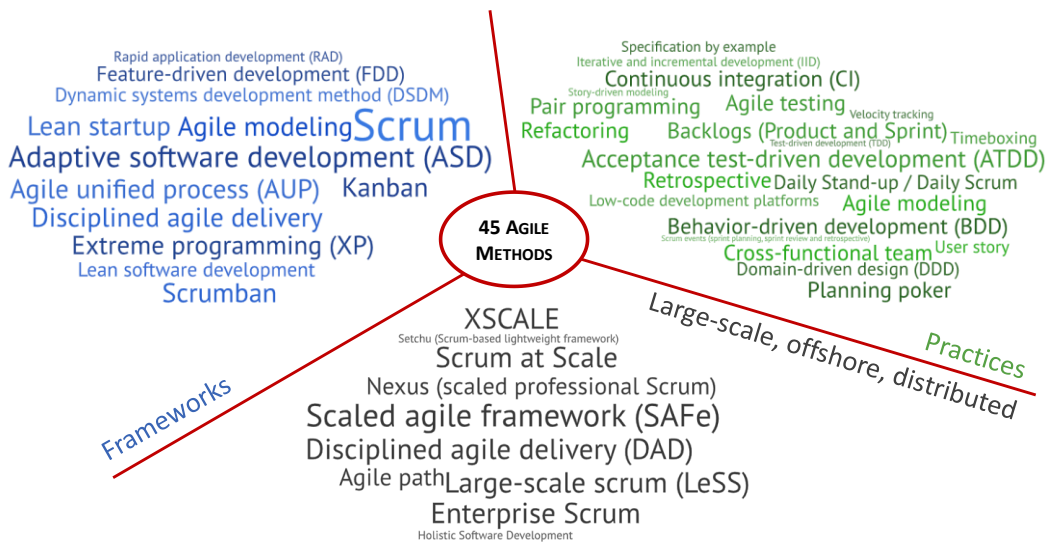


Figure 2. Currently available Agile frameworks and practices

Despite the wide selection of frameworks and practices, organizations still struggle to identify a development process structure that best fits their needs. Some organizations, enticed by the success stories hype, try to seamlessly implement Agile methods in their development process without accounting for both organizational and product boundary conditions. However, high expectations can lead to over-ambitious goal setting, and together with success stories, often comes an equal (if not greater) number of failures (Atzberger et al., 2019).

Some other companies, relying on their cultural legacy, fall back on classic plan-driven approaches such as the Stage-Gate (Cooper, 1990) approach or the V-Model (INCOSE, 2015), which often reveal to be too cumbersome and inflexible in dealing with fast-paced and ever-changing market conditions.

Both situations lead to frustration, schedule and cost overrun, waste of resources, and sometimes project failure. Empirical evidence (Garzaniti et al., 2019a; Golkar et al., 2019) suggests that the root of such troubles is not the lack of systematic methods to develop products but rather the inability to exploit available approaches tailoring the process to the specific project context and system features. Every development project is different, has its own peculiarities (i.e., product and process boundary conditions), and presents different challenges.

Therefore, it has been identified the need for a methodology to support the structuring of Agile or Hybrid-Agile product development for hardware systems (Garzaniti et al., 2019b), underpinning the decision-making process by quantitative analyses and statistical evidence. Such a decision support system shall benchmark different development process structures. It shall evaluate cost, schedule, and quality for each PDP structure and eventually offer suggestions to tailor the process to the specific project context and system features.

1.2 Scope of the research

This research is related to four fields of knowledge (Figure 3). The focus lies on the intersection of New Product Development, Project Management, Systems engineering, and design research disciplines.

Systems engineering is an “interdisciplinary approach and means to enable the realization of successful systems” (INCOSE, 2015). As a discipline, it offers valuable aids in managing

complexity in both technical and organizational processes. This work exploits the benefits of such approaches while facing the challenges of managing the lifecycle of products.

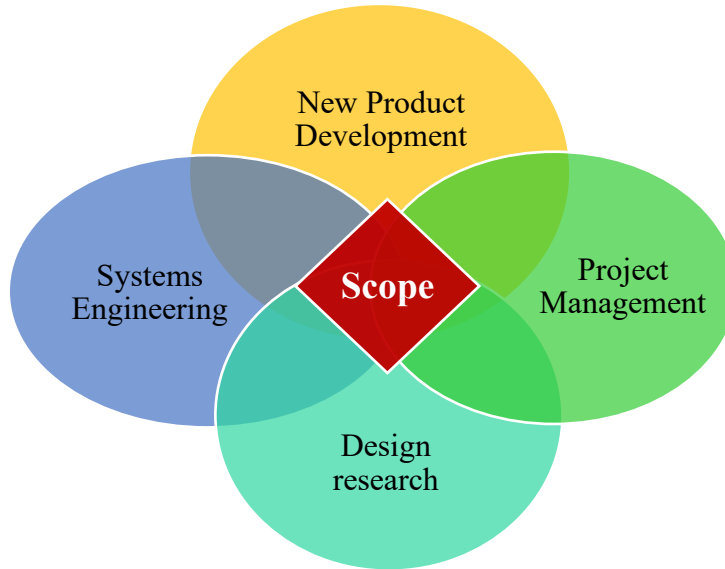


Figure 3. Relation to fields of knowledge

The second pillar of our research is the field of New Product Development (NPD). Generally speaking, NPD covers the entire process of bringing a new product to market or renewing an existing product. According to the ISO/IEC 15288, the product development stage includes all the activities required to move a product from the concept stage to the ramp-up or full-scale production stage (ISO/IEC JTC 1/SC 7, 2015). Figure 4 shows the positioning of this research within the overall product lifecycle stages. This work investigates the possibility of effectively implementing Agile development for complex hardware systems. It assesses the performance of Agile for hardware and explores the possibility of its integration with traditional development processes such as the Stage-Gate or the V-model.

The third pillar is the project management discipline. A complex development project typically involves hundreds of tasks executed by a network of professionals from various

disciplines and probably from multiple organizations. As project complexity increases, it becomes more challenging to manage the interactions among tasks, people, and all involved parties. It may be almost impossible to predict the impact of a single design decision over the whole development process. For this reason, project management techniques have traditionally played a key role in the successful execution of complex projects.

LIFE CYCLE STAGES	PURPOSE
<i>CONCEPT</i>	Identify stakeholders' needs Explore concepts Propose viable solutions
<i>DEVELOPMENT</i>	Refine system requirements Create solution description Build system Verify and validate system
<i>PRODUCTION</i>	Produce systems Inspect and test [verify]
<i>UTILIZATION</i>	Operate system to satisfy users' needs
<i>SUPPORT</i>	Provide sustained system capability
<i>RETIREMENT</i>	Store, archive, or dispose of the system

Concept stage	Development stage	Production stage	Utilization stage	Retirement stage
			Support stage	

Figure 4. Life cycle stages from ISO/IEC 15288 (ISO/IEC JTC 1/SC 7, 2015). In green the stage covered by our work.

Finally, the thesis engages in design research and adopts the Design Research Methodology (Blessing & Chakrabarti, 2009) to formulate, develop and validate our methodology and related tools.

1.3 Thesis objectives

The motivation section acknowledged the need for new development methodologies to deal with VUCA conditions. According to the current debate within the product development community,

Agile might represent a solution to such a concern. For this reason, several organizations have started adopting Agile methods in the development of physical products (Digital.ai & VersionOne Inc., 2020; KPMG, 2019). Nevertheless, despite the proliferation of Agile frameworks and practices, there is still little evidence of consistent delivery of the purported benefits (Schmidt et al., 2018b, 2019). Other companies, led by the skepticism toward Agile, fall back on traditional plan-driven approaches, which revealed to be unsuitable in dealing with fast-paced development conditions.

In light of the two outlined perspectives, the research has two specific objectives.

Goal 1

To support project managers and engineering teams in understanding when and how to use Agile within the development process of a complex hardware system.

Goal 2

To support project managers and engineering teams in structuring and executing Agile or Hybrid-Agile methods within product development projects.

This first goal is of conceptual nature and tries to address the rationale for implementing different degrees of agility within a given development process. The second goal tries to answer a clear industry need. The industry is seeking an integrated methodology, adopting transparent criteria, implemented in an easy-to-use tool.

1.4 Research Questions and Hypothesis

According to the objectives set out, this research aims to answer the following research questions.

RQ 1

How to understand when and how to use Agile methods within the development process physical systems based on the specific project context and system features?

The concept of understanding relates to the formulation of metrics and tools that, based on available project and system data, are able to provide an indication of the most suitable development process. Specifically, it is required to define metrics and related thresholds in terms of cost and time to recommend the use of Agile, Hybrid-Agile or traditional development processes. Then use this information to reason on the optimal level of tasks granularity to enable the implementation of Agile and the level of system modularity that best fit the Agile PDP.

RQ 2

How to support project managers and engineering teams in structuring and executing Agile or Hybrid-Agile methods within product development projects?

The concept of support relates to the infrastructure (methods and tools) required to enhance the effectiveness of Agile methods for physical systems development. To answer this question, we need to address the problem of minimum viable products (MVP) definition, the procurement and manufacturing management in the Agile context, the resource allocation and leveling, and the coordination aspects with consortium participants for hybrid Agile product development.

The research relies on the hypothesis that: as systems can have several architectures, and, among those, one will best fit customer needs, so product development processes can have different structures and, among those, one might best fit project targets and constraints.

1.5 Thesis Structure

The thesis is organized as follows.

Chapter 1 provides an introduction to the thesis, introduces the problem, and explains the motivation for the research work. Chapter 1 also explains the research objectives and presents the research questions.

Chapter 2 reviews the state of the art of the bodies of knowledge framing the research work. It provides the theoretical foundation of Agile theory, and particularly of Agile Scrum. It offers an overview of the product development literature, emphasizing the traditional stage-gate approach, and discusses currently available project management techniques. In Chapter 2, the literature survey is also complemented with a field research study to validate the gaps identified in the literature, thus better informing the research questions.

Chapter 3 presents the approach. It discusses the formulation of the decision support system in detail, mapping the approach steps to the gaps identified in the literature. Then, the framework is applied to an illustrative case. The objective is to demonstrate the use of such a system step by step. As an example case, a relatively simple product, like a software-defined X-Band transmitter for CubeSat, has been chosen. The example is also used to compare the proposed framework with traditional project management methods and tools. Following the illustrative case, the decision support system validity is addressed. The different validity aspects are discussed,

including the validity of the process model, the data model, the graphical user interface, as well as the general and face validity.

Chapter 4 presents the framework applied to the first case study. The objective is to verify the capability of the system. The chapter describes all the steps of the framework deployment over the project and evaluates how the system supports engineering teams, thus meets research goals and research questions. This first case study also has the objective to validate the methodology to some extent.

Chapter 5 presents the framework applied to the second case study. The objective is to verify the capability of our system in a different industry context as well as a different organizational structure. The chapter goes through the framework deployment steps over the project and presents how the system supports engineering teams, thus meets research goals and research questions.

Chapter 6 draws the conclusion. It summarizes the main outcomes of the research work, highlighting the contribution to the knowledge and the contribution to the practice. Lastly, it states the limitation of this work and sets the basis for future work.

1.6 Research methodology

The research approach adopted relies on the Design Research Methodology (DRM) (Blessing & Chakrabarti, 2009), as presented in Figure 5.

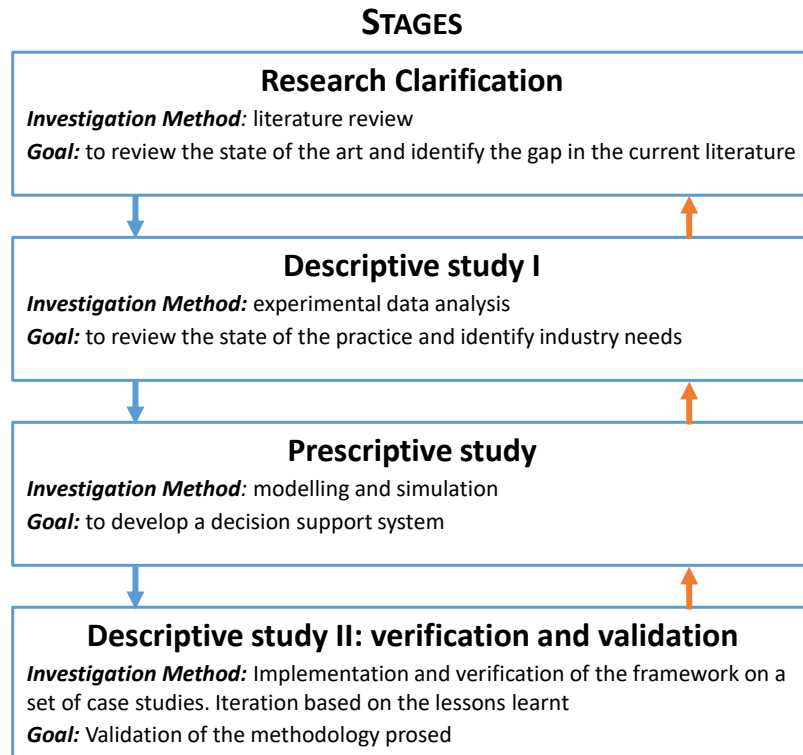


Figure 5. DRM framework adapted from (Blessing & Chakrabarti, 2009)

The chapters are mapped to the DRM methodology as follows.

1. Chapter 1 and Chapter 2 refer to the *Research Clarification* stage.
2. Section 2.5 discusses the field research and relates to the *Descriptive study I*.
3. Chapter 3 presents the approach and relates to the *Prescriptive Study I*
4. Chapter 4 and Chapter 5 present the case studies. These chapters evaluate the performance of our system and verify its capabilities. Finally, chapter 6 draws the conclusion. All these chapters relate to the *Descriptive study II*.

“Dicebat Bernardus Carnotensis nos esse quasi nanos gigantium humeris insidentes, ut possimus plura eis et remotiora videre, non utique proprii visus acumine, aut eminentia corporis, sed quia in altum subvehimur et extollimur magnitudine gigantea”.

(John of Salisbury, 1159)

Chapter 2

Background

This chapter reviews the state of the art of the bodies of knowledge framing our research work. It provides the theoretical and practical foundation of Agile theory, with a particular emphasis on Agile Scrum. The following sections review traditional product development approaches with a particular focus on the stage-gate model and discuss currently available project management techniques, contextualizing them in the problem of interest. This chapter also reports the field research study conducted to get additional insights into current challenges in implementing Agile for hardware, thus better inform the review. Following the literature review, the gap in the state-of-the-art is identified, and the scientific contribution of this work is defined.

2.1 Agile methodology and operational frameworks

Agile methodologies are based on the concept of “iterative enhancement” (Larman & Basili, 2003). Unlike the Spiral model (Boehm, 1988), Agile attempts to simplify the development practices, bringing flexibility at all levels of the lifecycle (Bott & Mesmer, 2020).

In essence, Agile is a very granular way to organize the work. It leverages short iterations and self-organized cross-functional teams. Iterations are typically self-contained mini-projects with

activities spanning from system design to manufacturing, assembling, and testing. Each iteration leads to a product release as a growing and an evolving subset of the final system. Short iterations are the key to getting early feedback from customers, discovering defects at all levels of development, and getting new knowledge for product refinement and requirements adaptation.

Agile finds its foundations in the Manifesto for Agile Software Development (Beck et al., 2001) and encompasses several operational frameworks (Al-Zewairi et al., 2017; Ozkan et al., 2020), including Adaptive Software Development (ASD), Agile Unified Process (AUP), Crystal Methods, Dynamic Systems Development Methodology (DSDM), Extreme Programming (XP), Feature Driven Development (FDD), Kanban, Scrum. This research considers the use of Scrum.

2.2 Scrum: an overview

Scrum (Schwaber & Beedle, 2001) is one of the most popular operational frameworks. The framework is constantly evolving; therefore, the two founding authors regularly publish an updated edition of their Scrum guide. The overview we provide in this section mostly relies on the latest Scrum guide published in 2020 (Schwaber & Sutherland, 2020).

Before going into details, however, we need some clarification on the nomenclature. The literature may use the word “Scrum,” referring to a method, a process, a methodology, a concept, a set of rules, and even a mindset. This fragmentation of the nomenclature generally causes some confusion.

Scrum is not a *method*. INCOSE defined method as a set of “*techniques for performing a task; in other words, it defines the “how” of each task*” (Estefan, 2008). Scrum does not offer much guidance in how to perform the actual development activities.

Scrum is not a *process* in a technical sense of the word. A process is typically based on series of actions or repetition of steps taken to achieve a particular and repeatable outcome. Since Scrum has some adapting capabilities to the project context, the term process might not be the most appropriate one.

Someone might refer to Scrum as a methodology. However, a methodology is essentially a collection of related processes, methods, and tools (Estefan, 2008), and since Scrum lacks process and methods, it is not a *methodology*. Scrum is not a *concept* but rather a practice. Scrum is a *framework* because it merely frames a set of principles, activities, and tools.

Scrum includes a set of micro-planning tools and strategies aimed at getting a working end-product quickly. The rhythm of the development is marked by short, iterative, incremental Sprints (i.e., a given development time range) with the objective to deliver a Minimum Viable Product (MVP) at the end of each Sprint (Figure 6).

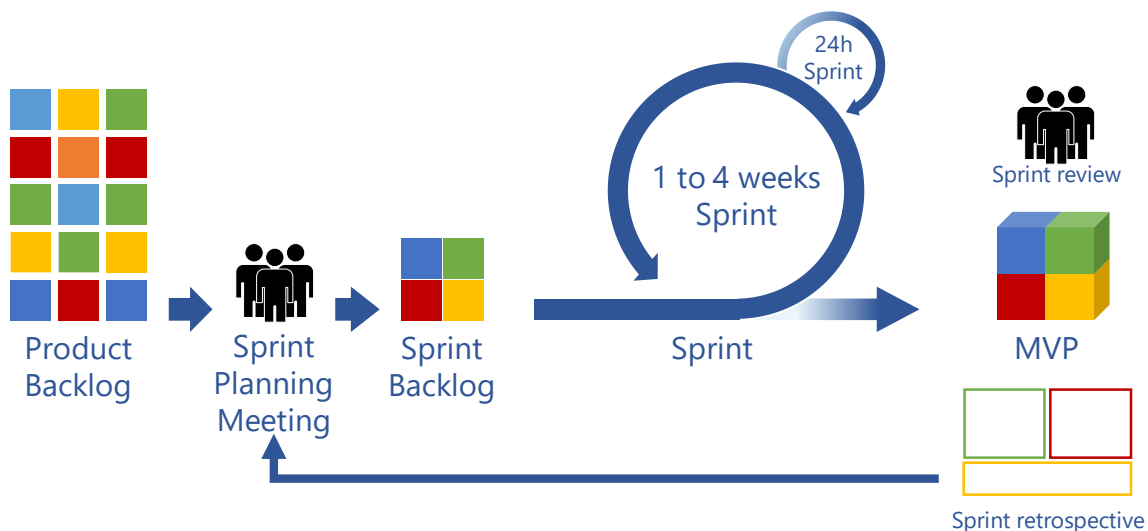


Figure 6. Graphical representation of Scrum methodology, sources (Garzaniti et al., 2019b)

2.2.1 Roles artifacts and events

The Scrum framework consists of three main elements: roles, meetings, and artifacts.

Table 1. Scrum framework elements

Roles	Product Owner, Development Team, and Scrum Master
Events	Sprint Planning, Daily Scrum, Sprint Review, and Retrospective
Artifacts	Product Backlog, the Sprint Backlog, and Product Increment

Roles: Product Owner

The Product Owner is responsible for maximizing the product value delivered to the customer. In some cases, even be the customer itself can be the Product Owner. He or she is responsible for defining the product goal and the product backlog items as well as backlog items organization. In the Scrum theory, the Product Backlog management is an exclusive prerogative of the Product Owner, but in practice, the task is sometimes delegated to the development team.

Roles: Development team

The Development Team includes the personnel who actually do the job. They are responsible for delivering the planned Minimum Viable Product at the end of each Sprint. Scrum promotes self-organizing cross-functional teams with no formal or informal hierarchic structure. Each team member is responsible for solving the tasks he or she is committed to individually. Therefore, backlog items must be granular enough to be addressed by a single person. The Development Team is often co-located to foster continuous and transparent communication amongst the team members. During a Sprint, the Development Team works only on tasks directly leading to the Sprint Goal. Sprint rescoping, i.e., add new tasks to the Sprint backlog while the Sprint is ongoing, is not allowed.

Roles: Scrum master

The Scrum Master is the enabler of Scrum implementation. He or She has to ensure that the development team follows the rules defined in the Scrum Guide. The Scrum Master can be a member of the team or external personnel. Both configurations can have advantages and drawbacks. The Scrum Master facilitates daily meetings, ensuring that the team has a clear understanding of development tasks in the ongoing Sprints and shares a common understanding of the product vision. Lastly, the Scrum Master also serves as an interface between the Development Team and the surrounding organization or relevant stakeholders.

Events: Sprint Planning meeting

All Sprints begin with a Sprint Planning Meeting. The whole Scrum Team participates in the meeting, i.e., Product Owner, Scrum Master, and Development Team. The sprint planning typically addresses three questions: 1) what is the goal of the Sprint? i.e., understating the value delivered to the customer 2.) what can be actually done? i.e., identify the activity required to achieve the sprint goal 3) how to select the job to get done? i.e., understanding development team capability.

The first two questions are addressed by negotiating between the Development Team and the Product Owner. The last question is addressed by using some qualitative supporting tools. One of the most popular is the *planning poker*. Given the criticality of this event, we will expand the literature review on the topic in Section 2.2.4

Events: Daily Scrum

The Daily Scrum is a 15-minute stand-up meeting held every day by the Development Team. The purpose of this meeting inspect progress and synchronize the work to reach the Sprint Goal. The Daily Scrum meeting is typically held every day at the same time and place

Events: Sprint Review

The Sprint Review is the meeting that concludes the Sprint. The team presents the Sprint results to relevant stakeholders and assesses the progress made toward the final product. The purpose is to inspect the work completed within the Sprint and define the way forward.

Events: Sprint Retrospective

As the Sprint Review inspects the product, so the Sprint Retrospective reviews the process. The Sprint Retrospective follows the Sprint Review and aims to identify potential refinement of the Scrum process to improve the team efficiency. The Retrospective analysis offers the opportunity to bring forward issues encountered while performing the Sprint and suggestions to improve. It became a pillar of the continuous improvement strategy known as “inspect and adapt”, constituting one of the foundations of the Scrum framework.

The Sprint

Sprints are fixed-length time boxes where the job gets done. The Sprint acts as a container for all other events (Table 2). A new Sprint starts when the previous is concluded. Various practices exist to monitor the development progress (e.g., burn-down charts)

Table 2. Scrum Events

	Sprint Planning	Daily Scrum	Sprint Review	Retrospective
Perspective	Forward-looking	Managing	Assessing	Retrospective
Focus	Task breakdown and estimation	Tasks Burn-down	The Product	The Process
Purpose	Planning	Team synchronization	Explain progress done	Improving the process
Participants	Scrum Team	Dev. Team & Scrum Master	Dev. Team & stakeholders	Dev. Team & Scrum Master
Input	Product Backlog Items	Development status	Work completed	Process-related issues
Output	Sprint Backlog	Plan for the day	Update for Product Backlog	Suggestion for process improvement

Artifact: Product Backlog

The Product Backlog is the collection of user stories, i.e., the list of the work to be done. It is a dynamic document defined and maintained by the Product Owner, and it evolves within the product development process. The Product Backlog may be ordered in several ways. Depending on the development strategy, the list can be sorted by value, risk, or priority.

Artifact: Sprint Backlog

The Sprint Backlog is a tool to visualize the work required to achieve the Sprint Goal. As the Product Backlog so the Sprint Backlog is a dynamic document that develops throughout the Sprint. It informs the Development Team on the work completed as well as the remaining work and time.

Artifact: Product increment

The Product Increment is the result of the Sprint. In earlier versions of the official Scrum Guide, the author used to give great importance to the results of each Sprint characterized as a Minimum viable product, i.e., a working part of the final product or a working prototype. However, in the latest Scrum Guide, greater emphasis is put on the “Definition of Done”. This is probably related to the difficulties of delivering a potentially shippable increment after each Sprint, and the “definition of done” represents an easier way of addressing the issue. Since the Minimum Viable Product represent one of the distinguishing features of agile approaches, particularly Scrum, we will expand the review on the topic in section 2.2.3

2.2.2 Iterations

Iterations are ubiquitous in the design and development of engineering systems (Maier & Störrle, 2011). They are also one of the cornerstones of Agile approaches (Schuh et al., 2018a). Nevertheless, managing iterations is not an easy task; reaping their benefits is not an obvious conclusion (Costa & Sobek, 2003).

In design theory literature, iterations are broadly defined as the repetition of an action or as a heuristic reasoning process depending on the perspective adopted (Wynn & Eckert, 2017). Building on these definitions, researchers developed different taxonomies to help better understand the nature of the phenomenon. Those works propose to classify iterations based on designers behavior (Wynn et al., 2007), information/task interdependencies (Smith & Eppinger, 1997a, 1997b), design process attributes (Costa & Sobek, 2003), or evolution of problem-solution space (Dorst & Cross, 2001). The reader can refer to (Wynn & Eckert, 2017) for a comprehensive review of the topic.

This research adopts a perspective complementary to existing theories. It focuses on the development stages involved in the iteration (e.g., design, prototyping, manufacturing, testing) as well as the verification/validation activities carried out (thus the technical risk reduction and TRL advancement). Two primary forms of iteration cycles have been identified: the *design-analyze-redesign (DAR)* and the *design-manufacture-test-redesign (DMTR)*.

DAR cycles are used in almost all engineering design practices. They consist of a concurrent, iterative exploration of the design space based on modelling and simulation. They usually represent an effective tradeoff between cost/time invested and technical risk mitigated (Unger & Eppinger, 2009).

DMTR cycles are a distinctive feature of flexible PDPs (e.g., Spiral, Agile). Modelling and simulations are complemented with verification and testing activities performed on the real product or a subset of it. DMTR cycles cover several phases of the product life cycle (design, manufacturing, assembly integration, and testing), thus reducing project technical risk at multiple levels. However, even if technical risk decreases and product quality theoretically improves with successive cycles, iterations significantly impact cost and schedule (Garzaniti & Golkar, 2020).

Reliably modelling such cycles becomes crucial when attempting to implement Agile in complex hardware projects. Product and process interdependencies may completely overturn advantages in cost and schedule brought by Agile (Golkar et al., 2019).

Currently available frameworks rarely model and analyze iterations in detail. Traditional network-based project management techniques (Ben Issa & Tu, 2020), such as PERT/CPM (Wiest & Levy, 1977), do not include iterations. More recent frameworks such as DSM-based simulation models characterize iterations by rework probability and rework impact (Browning & Eppinger, 2002; Cho & Eppinger, 2005), (Ma et al., 2019). Although these last ones proved to be powerful tools for analyzing process performance, the reliability of results is limited by the accuracy of input information (not always available and hard to assess). Furthermore, they all fall short in modelling iterations propagation in the later phases of the product lifecycle (e.g., procurement, manufacturing, integration, and testing), making the iterations planning and management difficult.

2.2.3 *Minimum Viable Product definition*

The primary outcome of an iteration cycle is the *Minimum Viable Product* (MVP). The concept of MVP has long been discussed in the literature (Lenarduzzi & Taibi, 2016). Scholars and professionals have analyzed different aspects associated with the MVP definition, such as the MVPs economic value (Rancic Moogk, 2012), the effectiveness of customer feedbacks depending on the MVP structure (Schmidt et al., 2018a), the role of prototyping (Bergweiler et al., 2019), (Schuh et al., 2018b), and the frequency of MVPs releases (Anderson et al., 2017).

Nevertheless, the definition of MVP within the context of complex hardware systems development is not straightforward. The physical aspect of the system hampers the implementation of a fully functional product increment. Furthermore, while there is a solid body of knowledge on

the MVP economic and market aspects, current literature still lacks perspective in capturing the MVP technical and development process aspects.

2.2.4 Sprint planning

In Scrum theory, *Sprint planning* is the event when the team defines the Sprint objective (i.e., MVP to be delivered) and lays out the work to get done (i.e., *Sprint backlog*) (McGreal, 2020). The *Sprint backlog* (Scrum.org, 2019) is a subset of the *product backlog* (Sedano et al., 2019), including the *user stories* (Lucassen et al., 2016) selected for the Sprint. In “traditional” Agile, *user stories* are defined as atomic elements to be performed by individuals, with no explicit consideration of product or process dependencies (Sedano et al., 2019). This entails several drawbacks during *Sprint planning* activities.

The teams can realize while performing the Sprint that some tasks are missing: thus, the MVP cannot be delivered. The decision to add tasks to the backlog while the Sprint is running will unavoidably result in delays and cost overrun (Figure 7). Frameworks currently available do not tackle this issue. They neither elicit nor represent user stories’ interdependencies.

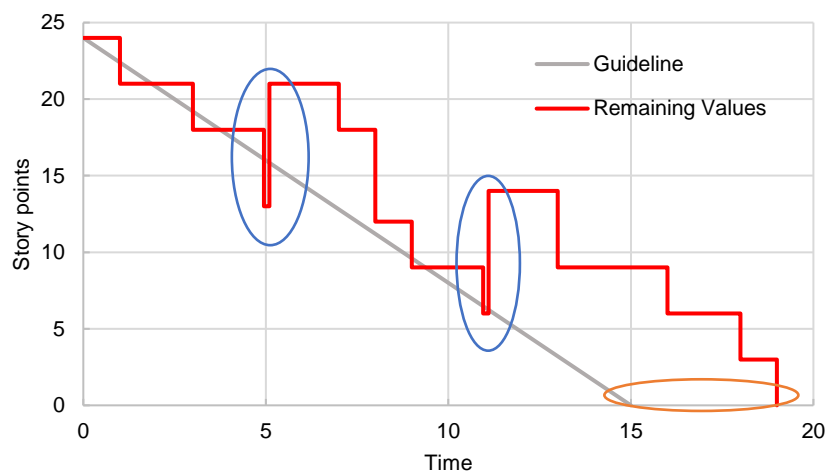


Figure 7. Sprint burndown chart - Tasks added during Sprint execution (vertical steps marked in blue) – Delays due to new task (marked in orange)

Additional challenges are also posed by the scoring system adopted to evaluate the complexity of the tasks allowing their prioritization within the Sprint time-box. The complexity index used in traditional Scrum (i.e., *story point*) does not provide any viable information to support the development process (Mahnič & Hovelja, 2012).

Recently some progress has been made in formulating parametric models for efforts estimation (Kamal Tipu & Zia, 2012), (Briatore & Golkar, 2021). However, current scoring systems do not clearly relate to cost or time, or resources needed to complete a user story. Furthermore, they still cannot provide reliable estimates and eventually bring more complexity than benefits to the process.

From an organizational standpoint, complex engineering systems are frequently developed in multiparty consortia. Each organization in the consortium runs its own agenda and adopts its preferred product development process. Therefore, Agile implementation, and specifically Sprint planning, becomes more challenging as it might require coordination with traditional systems engineering approaches and shall comply with consortium milestones.

Even if research efforts have been dedicated to these concerns (Garzaniti et al., 2019b; Ramos et al., 2013), most of the implementations in real industrial settings were not as successful as hoped (Atzberger et al., 2019).

This misalignment between literature contributions and industry results highlights a clear gap in the definition of a structured framework and a coherent methodology (encompassing processes, methods, and tools) to support the deployment of Agile for complex hardware systems development.

2.3 Product development processes in the literature

The product development literature is quite broad and relies on more than half a century of research and experience. The product development body of knowledge encompasses processes, methods, and tools aimed at delivering a new product or improving an existing one.

The literature has been surveyed, mapping the different branches of product development knowledge to frame our research, thus position our work. For this literature mapping exercise, different repositories of knowledge have been surveyed using the following keywords (and respective permutations): *product, development, process, method, approach, model, philosophy, strategy*. The survey has then been refined by adding in subsequent rounds of search the following keywords: *agile, stage, gate, spiral, concurrent, sequential, flexible*. The literature mapping showed that, in general, product development processes could be classified based on the flexibility and degree of tasks concurrency (Figure 8).

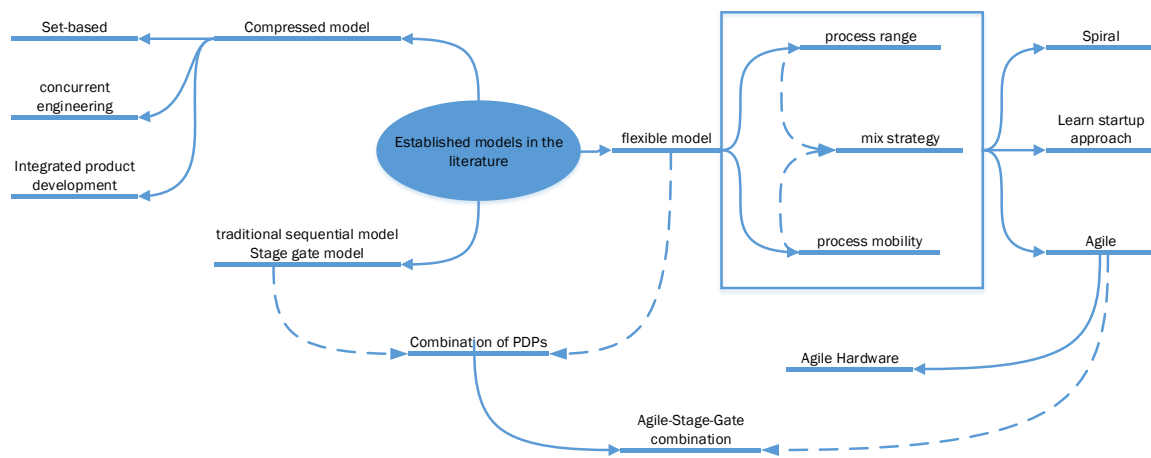


Figure 8. Map of the Product development literature

Compressed models (Iansiti, 1995) include *concurrent engineering* (Lawson & Karandikar, 1994; O’Neal, 1993; Stevens, 2015; Valle & Vazquez-Bustelo, 2009; Yassine et al.,

2003; Zhang & Daguang Zhang, 1995), *Set-based design* (Ghosh & Seering, 2014), and *Integrated product development* (Gerwin & Barrowman, 2002; Khaleeq uz Zaman et al., 2017; Sommer et al., 2014; Yingkui Gu et al., 2006). Traditional sequential models mostly refer to the *Stage-gate* (Cooper, 1990; Royce, 1970; Sethi & Iqbal, 2008). Flexible models (Boehm, 1988; Burger et al., 2017; Thomke, 1997) can be classified based on process range (Seebacher & Winkler, 2014), process mobility (Upton, 1995), and their combination (Fernandes et al., 2012). Flexible models include the *Spiral model* (Boehm, 1988; Buijs, 2003), the *Learn startup approach* (Boehm & Turner, 2005; Fazzi Bortolini et al., 2018; Ghezzi & Cavallo, 2018), and *Agile* (Douglass, 2016a; Schön et al., 2017; Schuh et al., 2018a). From these models over the year, hybrid approaches got momentum (Bianchi et al., 2018; Cooper & Sommer, 2018; Mahmoud-Jouini et al., 2017; Robey et al., 2018) and particularly the combination of *Agile-Stage-Gate combination* (Begel & Nagappan, 2007; Boehm, 2004; Cooper, 2016; Cooper & Sommer, 2018; Karlström & Runeson, 2005, 2006; Sommer et al., 2015). Despite the variety of approaches, the scope of research is limited to Agile, particularly the scrum version as presented in Section 2.2, and to the traditional Stage-Gate model, one of the most adopted approaches for a wide range of applications. The combination of the two is also considered.

2.3.1 The State-Gate process

The Stage-gate approach (Cooper, 1990), also called waterfall, phase gate, toll gate, checkpoint, or structured product development by different authors and practitioners (Unger & Eppinger, 2009), is a well-established product development process. Stage-Gate has been designed to help firms to select the right projects, and once selected, to map out the key stages, best practice activities, and roles and responsibilities as part of the project, bringing discipline to “chaotic” new product development (NPD) activities (Sethi & Iqbal, 2008).

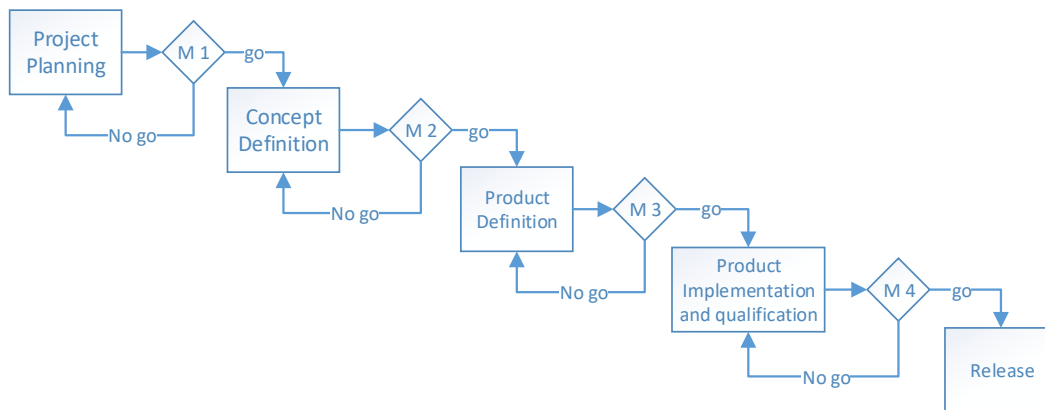


Figure 9. A representation of the Stage-Gate model

The ideal Stage-Gate process proceeds in distinct stages, from product planning to product release (Figure 9). At the end of each phase is a review, or gate, to evaluate whether the previous phase was successfully completed. If the project is reviewed positively, work proceeds to the next phase. If not, then the project iterates within that phase until it can successfully pass the review, or the project may be terminated.

The major advantage of stage-gate processes is to provide structure to the development by reaching sharp product definitions and specifications early in product development. Technical risk is reduced because narrow iterations and reviews freeze specifications early. Rigid requirements and stable product definitions help to avoid errors by avoiding midstream corrections (Cooper, 1990).

The main drawback of this product development process (PDP) is inflexibility. Narrow iterations cannot incorporate feedback from later phases. Failure may occur if early specifications and assumptions are proven wrong by subsequent market research or prototyping.

2.3.2 Agile for hardware

Agile development of physical products, Agile hardware development, and Agile development of cyber-physical systems are used as synonyms in the scientific literature. The terms refer to the development of products consisting of a combination of software and hardware elements (e.g., electronics, mechanisms, and many others).

While previous sections reviewed the literature on some specific topics related to the Agile development of physical products, this section offers a general overview of the research in the field. Due to the progressive spread of Agile in the hardware domain as well as the growing popularity such methodologies reached over the last two decades, there is a vast literature on the topic. A search on the Scopus database using the keywords “Agile” AND “Hardware” and “Agile” AND “physical” AND “product”, as of the time of writing this thesis, has yielded a total of 1459 results, covering five decades of research. Given the fast pace evolution of the topic, this thesis limited the focus to publications that occurred in the last ten years.

This wide body of knowledge addresses different aspects relevant to the Agile development of physical products. Dove & LaBarge (2014a, 2014b) provided a philological analysis on the concepts of Agile systems-engineering and agile-systems engineering, while Dove (2002) provided a philological analysis on the concepts related to the Agile enterprise. Atzberger & Paetzold (2019) analyzed the challenges of adopting agile for hardware development, updating the analysis of Ovesen & Dowlen (2012). Punkka (2012) analyzed the challenges and the opportunities of implementing Agile in the co-design of embedded systems. Schmidt et al. (2018c) offered a critical perspective on the expected vs real effects of Agile development of physical products, while Atzberger et al. (2019) reviewed how the Agile for hardware development has matured over the last decade and how the hype evolved.

Schmidt et al. (2018b, 2019) and Atzberger et al. (2020) conducted a set of empirical studies addressing different thematics related to the agile development of physical products, such as motivations, transition, potentials, and applicability. To date, these studies represent one of the most compelling sources of data in the domain of Agile for hardware.

The literature also addresses the challenges of scaling and decentralized development (Alqudah & Razali, 2016; Dikert et al., 2016; Ullah et al., 2011); however, those are issues not limited to the context of hardware development. Lastly, several authors also presented case studies on the application of Agile for developing hardware systems, such as Huang et al. (2012), Edwards (2017), or Dove et al. (2017).

Overall, the literature focuses on the challenges and opportunities entailed in adopting agile for developing physical products yet not providing enough actionable solutions to overcome such challenges.

2.3.3 Hybrid-Agile approaches

The debate is ongoing on whether Stage-Gate and Agile are compatible and complementary and how to best mix the two approaches to leverage their respective strengths and mitigate their weaknesses (Bianchi et al., 2018). To date, there is still no widely shared agreement on the topic.

Cooper & Sommer (2016) describe the cases of established firms benefiting from the use of Agile practices within their existing Stage-Gate systems. Some authors speculate on their potential to deliver exceptional innovation outcomes (Papadakis & Tsironis, 2018), however few studies rigorously examine how the integration of Stage-Gate and Agile affects product development performance (Batra, 2018; Bianchi et al., 2018).

Sommer et al., (2015) suggest that using a Stage-Gate model at the strategic level together with Scrum tools at the execution level increases NPD productivity, flexibility, and coordination. Dikert et al., (2016) instead indicate that the coexistence of the two approaches causes tensions at all organizational levels, bureaucracy duplication, and reward system mismatch.

Cooper (2016) provided a formal characterization of Stage-Gate and Agile based on the type of management (macro vs micro), the scope of the project, the structure of the organization, and the decision model. Even if this represents a seminal work in the field, it still does not provide clear indications on how to select the development process and manage the Stage-Gate-Agile hybridization.

Overall, beyond qualitative analyses on the effect of adopting hybrid PDPs, the literature lacks rigorous approaches to combine Agile and Stage-Gate and support the management of projects over their execution.

2.4 Catalog of gaps in the literature

Section 2.1 and Section 2.2 have provided a brief yet comprehensive picture of Agile theory, emphasizing Agile-Scrum. Section 2.3 instead briefly surveys product development literature, focusing on the Stage-gate model and its combination with Agile. This paragraph summarizes the gaps identified in the current Agile theories and practices, focusing on the challenges of implementing Agile for physical products development. Furthermore, this paragraph reports current challenges in coordinating Agile with the traditional development process. This synthesis will serve as a guide in the development of the thesis and will also be used for evaluation purposes in conclusion.

Gaps in current Agile theory

1. Lack of a taxonomy for Minimum Viable Product definition that accounts for technical and development process aspects (ref. to section 2.2.1 and Section 2.2.3).
2. Lack of a method to support the tasks' prioritization and Sprint planning activities that, accounting for user stories interdependencies, minimizes the Sprint rescoping (ref. to section 2.2.4).
3. Lack of a method to reliably evaluate the Sprint workload accounting for resource availability and allowing for proper resource allocation and leveling (ref. to section 2.2.4).
4. Lack of methods to model and evaluate the impact of iterations propagation through the development process, thus inform on the viability of given iterations (ref to Section 2.2.2)

Gaps in the relation between Agile and traditional product development processes

1. Lack of theories to reconcile traditional and Agile approaches (ref to Section 2.3.3).
2. Lack of quantitative metrics to evaluate the viability of implementing the different product development processes (ref to Section 2.3.3 and Section 2.2.2).
3. Lack of methods to coordinate the interdependencies between different organizations implementing different development processes, accounting for the complex interplay between all the stakeholders (ref to Section 2.3.3 and Section 2.2.2).

2.5 An industry perspective

To validate the gaps identified in the literature, thus better inform our research questions, the literature-based investigation on the state-of-the-art has been complemented with field research. The implementation of Agile for hardware development has been analyzed in a real industrial setting, specifically in the context of a “New Space” mission. Based on these analyses, it has been possible to quantitatively assess the effectiveness of implementing Agile to develop space systems (Garzaniti et al., 2019a) and identify challenges and opportunities deriving from the multiparty consortium dynamics (Garzaniti et al., 2019b; Golkar et al., 2019).

2.5.1 New Space and the question of the product development

In the last decades, global space activities radically changed with greater involvement of private stakeholders, hundreds of startups created worldwide, and a significant increase in the influx of private capital into space ventures. This new evolution of the space age is popularly known by the name of “New Space” (Garzaniti et al., 2021). Within this new phase of maturity of space exploitation, organizations increasingly focus their activities on economic profit, addressing the need of customers coming not exclusively from the space industry.

Due to the explicit drive for profitability, New Space ventures started exploring new business models and new product development approaches. The trend to prioritize activities with lower capital cost requirements and short time-to-market, together with the need for a faster and more adaptive response to changes in customer needs, is challenging traditional product development processes such as the V-model and the Stage-Gate model (INCOSE, 2015), and making Agile methods (Beck et al., 2001) a potential key enabler of the New Space sector.

While Agile methods are well known in the software industry, their use in hardware development is not widely explored, not to mention in the space engineering domain. The status quo is largely made of space projects that are structured and executed following the established development processes. Nevertheless, there are hints in the scientific and grey literature that New Space organizations are experimenting with Agile product development. This chapter analyzes the experience of one of these organizations that engaged with Agile to develop its product.

2.5.2 Mission overview

The use case is a nanosatellite mission developed in a multiparty consortium including six different organizations (SMEs, startups, universities, and institutional partners) spread over five countries (Camps et al., 2018). At the consortium level, the traditional *Space Flight Project Life Cycle* has been adopted (NASA, 2007). The consortium defined the overall mission requirements, the success criteria, and a high-level schedule; individual organizations implemented their portion of the project, according to this master plan, adopting their preferred product development process (PDP).

Mission requirements were formulated based on the contents of the mission proposal and have been formalized in a Mission Requirements Document (MRD). Consequently, each organization in the consortium has cascaded the MRD requirements into system requirements for their specific contribution to the project. The system requirements have been consolidated into a System Requirements Document (SRD). Based on the MRD and SRD, each organization has shaped its own PDP.

The prime contractor managed the PDP at the consortium level and summarized High-level activities and the main milestones in a traditional Gantt chart.

This schedule included:

1. Preliminary Design Review (Kick-Off + 1 month);
Success criteria: definition and finalization of all mission requirements
2. Critical Design Review (K0 + 4 months);
Success criteria: definition of interfaces and system requirements
3. FlatSat testing (K0 + 6 months)
Success criteria: successful pass of interfaces test and functional test.
4. Payload delivery to Integrator (K0 + 9 months)
Success criteria: payload delivered; performance and environmental test passed.

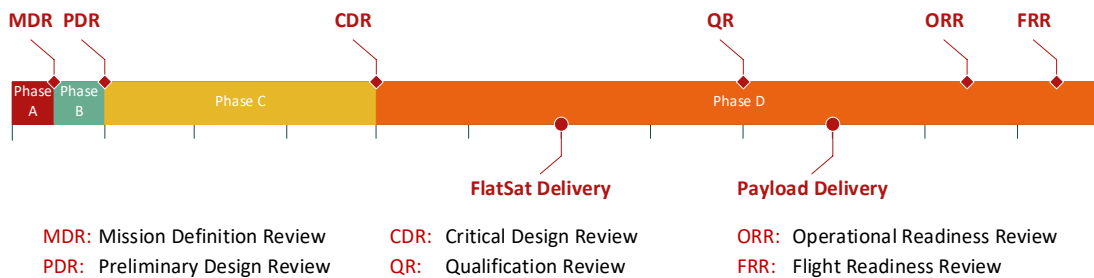


Figure 10. Use case Project Life Cycle (design, development, and integrations stages)

The first two milestones concern the design phase, while milestones three and four refer to hardware implementation and test.

In an effort to ensure delivery of its payload on a tight schedule (less than 12 months), one of the consortium participants structured the development process following an Agile approach, and particularly the Scrum framework. This organization was responsible for the developed an optical inter-satellite communication (O-ISL) payload.

A functional block diagram of the payload is presented in Figure 11. The system includes a transmitter (TX), a receiver (RX), and two electronic boards, one for command and data handling (C&DH) and one for power management (EPS). The receiver is based on a Cassegrain architecture and includes a primary mirror steering device, while the transmitter is equipped with an electromechanical solution to enable fine-pointing operations.

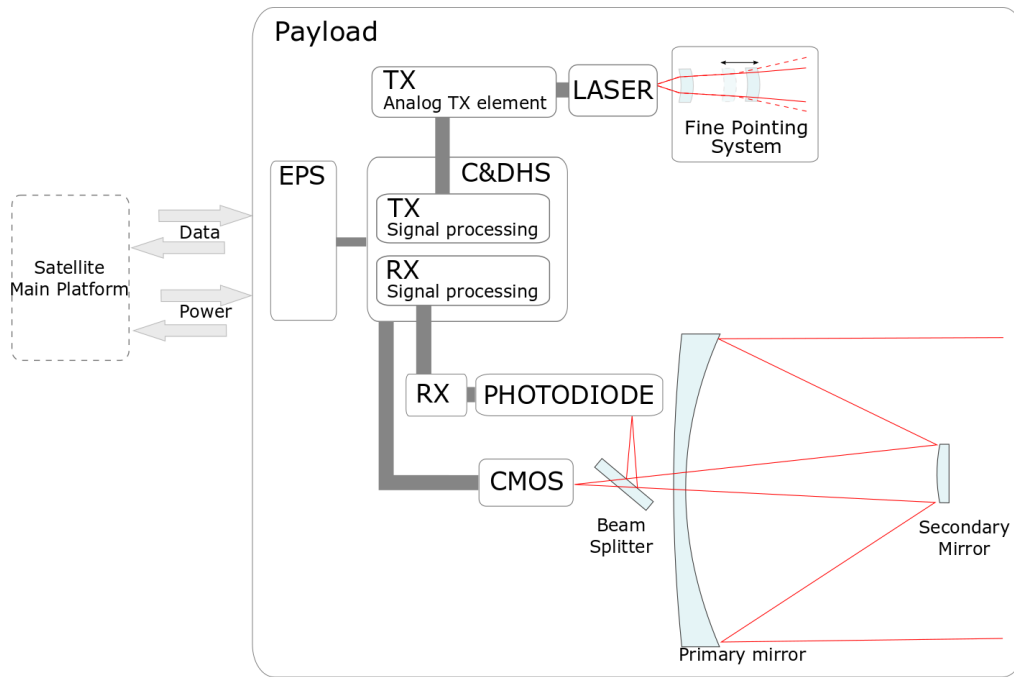


Figure 11. Functional block diagram of the payload, source (Garzaniti et al., 2019a)

2.5.3 Using “traditional” Scrum

Following Scrum, the work (i.e., design of the entire payload) has been decomposed into stories related to the development of the main subsystems of the optical terminal:

1. Optics
2. Electronics
3. Mechanics
4. Software

These high-level user stories (also known as *epics*) are the building blocks of the O-ISL terminal and could be mostly handled as independent projects. Additional epics relate to the definition and implementation of interfaces between subsystems, such as integrating the support structures with the electronics or integrating the optical trains. Each epic is characterized and decomposed into a subset of user stories. The decomposition provides structure to the development process. The collection of all the user stories constitutes the *product backlog*.

In this phase, we can already observe the first divergence from traditional Scrum and one of the limitations of Agile in hardware development projects. Scrum was initially conceived for a homogeneous environment in terms of team expertise (e.g., ability to develop software using object-oriented programming). Here, the development had to span across different disciplines (optical, mechanics, electronics), each with its own requirements, design strategies, and issues. It is not possible to use the same approach or the same personnel to design mechanical components and electronic boards.

Later on, during the *Sprint planning* meeting, such user stories, if needed, were further decomposed into smaller tasks to be ideally completed within a one-day time frame. Then they were collected in the *Sprint Backlog* setting the objective of the Sprint. A team member was assigned the role of “*Scrum master*”. Its role was to coordinate the team inputs/outputs and facilitate the execution of the Agile process.

Sprint planning was performed collaboratively among all the team members using a Fibonacci sequence (Falcon, 2016) scoring system. Tasks scoring aimed to evaluate the complexity of each task and allow activities prioritization within the Sprint length. Scoring activities are performed as follows: each team member votes providing a score in the Fibonacci sequence (1, 3, 5, 8, ...). Afterward, the median score is recorded by the Scrum master and assigned to the

task. The resulting Sprint backlog is then reviewed, prioritizing the activities according to the agreed time box. Since providing scores is an empirical activity and mostly qualitative, task effort estimation reliability tends to improve over successive sprints.

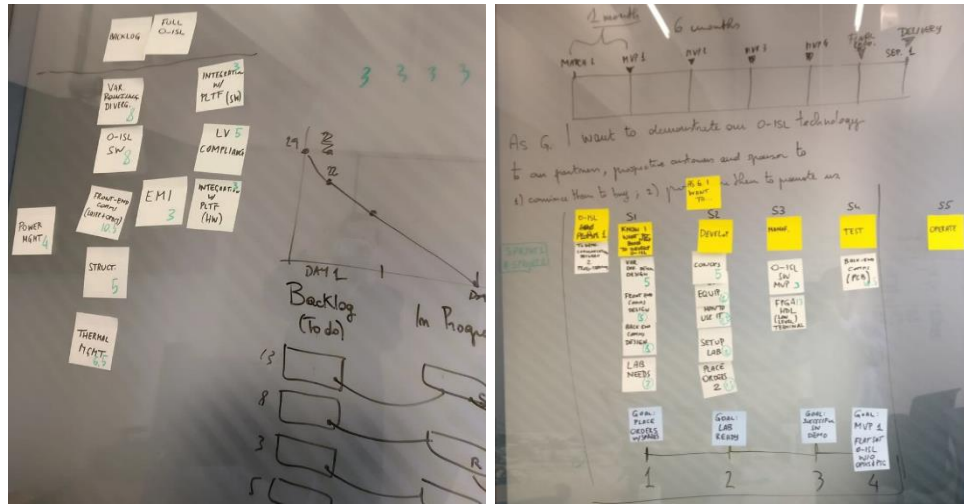


Figure 12. User stories with scores and preliminary tasks prioritization on a physical Kanboard

The development was structured in Sprints of duration between one and four weeks. Each Sprint was devoted to completing the Sprint backlog tasks and delivering a minimum viable product. Initially, one-week Sprints were adopted, but the team quickly realized that this duration was way too short for consolidating results in the development. The sprint length became an issue, particularly as the interdependencies with the supply chain and related lead times became more relevant as the design evolved from a “cocktail napkin feasibility study” (Balint & Freeman, 2017) to design models and ultimately to a physical system.

The tempo of the Sprint was marked by daily 15-minute stand-up meetings and daily close-out meetings. At the end of each Sprint, the team performed a retrospective analysis to understand what worked well, what did not work, and derive the lessons learned to improve the process in the subsequent development iterations. In order to assess the correct implementation and constantly monitor the status of the project, the team made use of a set of tracking technologies.

2.5.4 Support tools

Task organizer

Jira (Atlassian, 2021) has been used to track tasks, task scores, and development status. Jira provides a graphical representation of the development and indication of the distribution of tasks in the team (Figure 13).

Time tracker

The team used Toggl (Toggl, 2021) to track the actual time spent on each task. This information was helpful to establish a correlation between time and task complexity score, thus improving the effort estimation during subsequent sprint planning activities. For instance, on a printed circuit board (PCB) with more than 1,000 traces, measuring the time needed to complete ~100 traces allowed estimating the time required to complete the remaining 90% of the task with $\pm 5\%$ average time accuracy.

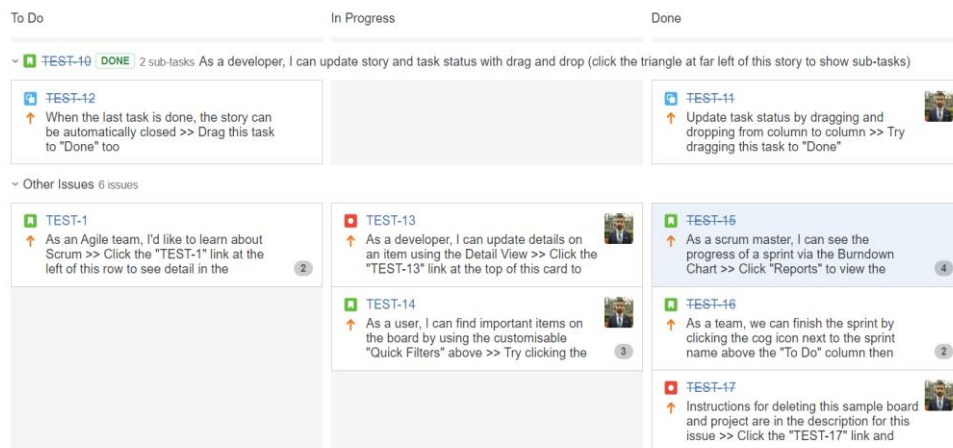


Figure 13. Example of Jira user interface

Repository monitoring

The team used GIT (Git, 2021) as the primary repository manager for all the design data. Git allows for tracking changes and manage file versions, ensuring that every team member is working on the

correct file at every given moment. Analyzing the evolution of the git-commits is also possible to roughly estimate the progression of the design.

2.5.5 Scrum implementation

This section reports the analysis of process data related to the first six months of development (i.e., FlatSat testing – milestone 3 of the project). The scope of this analysis has been limited to that period because later, the development team significantly revised the development project diverging from traditional Agile. Nevertheless, it is already enough to get preliminary results and draw interesting conclusions on the use of Scrum for space hardware development projects.

The design and implementation process to achieve the FlatSat delivery consisted of 11 Sprints with Sprint lengths ranging from 1 week to 4 weeks. The team experimented with different durations (Figure 14) to understand what can be the most effective in terms of MVP delivery.

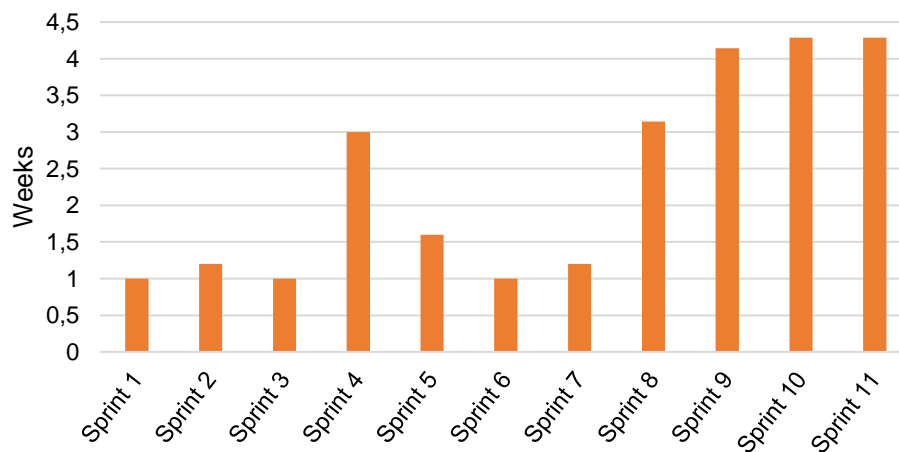


Figure 14. Sprint length

The traditional one-week Sprint always resulted ineffective, while a longer period allowed to get closer to the Sprint completion. However, such data shall not lead to wrong conclusions. It is not possible to define a statistical correlation between Sprint duration and Sprint effectiveness

measured as the percentage of completed tasks over the planned ones because of the multi-criteria nature of the problem. It shall also be noted that sprints longer than four weeks were mostly avoided because larger periods entirely overturn the benefits of early technical debt retirement.

While experimenting with different Sprint lengths, the development team also tried to find the best fit for the story points per day to be allocated. Figure 15 reports the estimated task complexity per day for each Sprint.

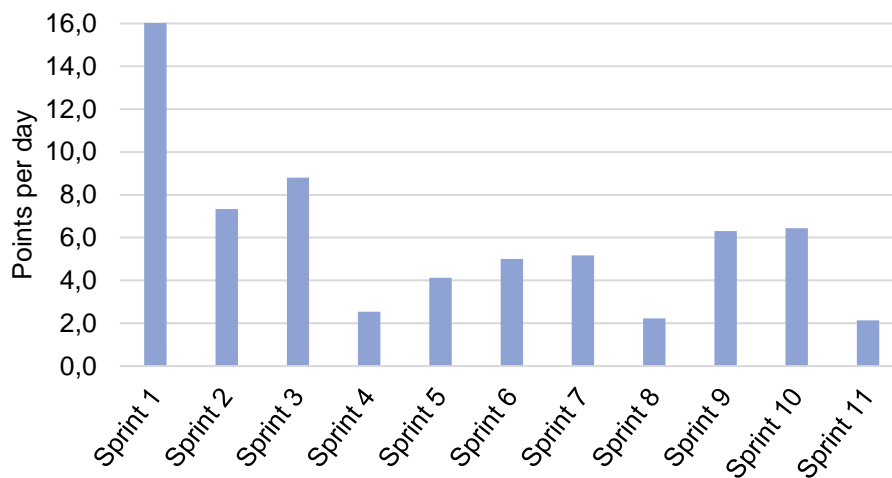


Figure 15. Planned story points per day for each Sprint

In the beginning, the team overestimated the workload but gradually refined the estimates, reaching a steady-state of about 5 points per day with a core development team of three people. It shall be highlighted that these figures apply only to the specific development team, as the scoring system is subjective and affected by biases. In addition to the learning effect, the daily story points decrease was also linked to the project evolution. The more the project got *physical*, the fewer points per day have been achievable. This situation highlights the complete unreliability and inefficiency of the traditional complexity-based scoring system: designing a component and manufacturing a component might exhibit the same task complexity but have utterly different execution times and costs.

A remarkable finding is that all Sprints failed. The team defined a *failed Sprint* as a Sprint where only a subset of the planned tasks was completed. An average of 59% of planned work (with a standard deviation of about 12%) has been completed per Sprint (Figure 16). Nevertheless, the FlatSat has been delivered according to the consortium project schedule. This kind of “failure” seems to be an issue shared by many development teams adopting Agile in hardware projects. The reasons for such a failure are three-fold.

First, engineering teams required a couple of sprints to refine the task estimation, tune sprint planning, and adapt to the Agile workflow. Engineers may often underestimate the complexity or misunderstand the interdependencies between different user stories. For example, after the Sprint began, the team might realize that tasks are missing in the backlog as they proceed with the activities. Alternatively, the team may realize that they included unnecessary tasks for delivering the given MVP. Finally, it can also happen that a task considered of moderate complexity requires a whole week, hampering the MVP delivery. The scientific literature also substantiates those statements (Feldmuller, 2018; Garzaniti et al., 2019a; Gregory et al., 2015).

Second, process disruption occurred due to interdependencies with components procurement and manufacturing lead times, which are inherent characteristics of physical systems not occurring in Agile software projects (Schmidt et al., 2019, 2017).

Third, there is still no consensus on the definition of a *successfully completed* Sprint. Completing all tasks in the backlog does not necessarily imply delivering a meaningful MVP (Schwaber & Sutherland, 2020).

For those reasons, several activities within the use case project have been rescheduled during Sprint executions. Rescheduling typically involved a percentage close to $35\% \pm 6\%$ of the total points foreseen for a given Sprint (Figure 16).

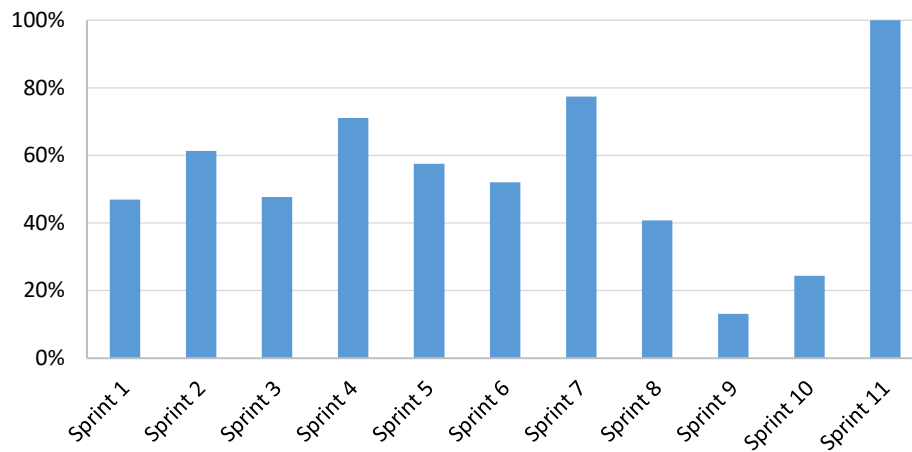


Figure 16. Percentage of work completed per Sprint

2.5.6 MVP: Customer feedback and technology de-risking

The main alleged advantage of Agile is the rapid development of products in iterative steps (MVPs), with significant involvement of the customers. In theory, they evaluate the system from the early stages of the development, thus provide feedback to align the product to their actual needs.

While this customer-centered approach can be easily implemented in the software domain, it is more challenging to include customer evaluation in the design and development of hardware systems, let alone a space-flying payload. The reason is that most valuable feedbacks come with activities occurring during assembly, integration, and testing (AIT) phases, thus during later stages of development. By then, the technical debt accumulated is quite large, and iterations would completely overturn the benefits of fast development.

However, some design strategies might help exploit this Agile feature: incremental design and rapid prototyping shall be adopted as best practices for all engineering disciplines involved (Bergweiler et al., 2019; Schuh et al., 2018b; Zink et al., 2017). In this use case, the team used 3D printing technology for rapid prototyping mechanical components. Incremental PCB prototyping (Figure 17) has been used instead for electronics and avionics elements.

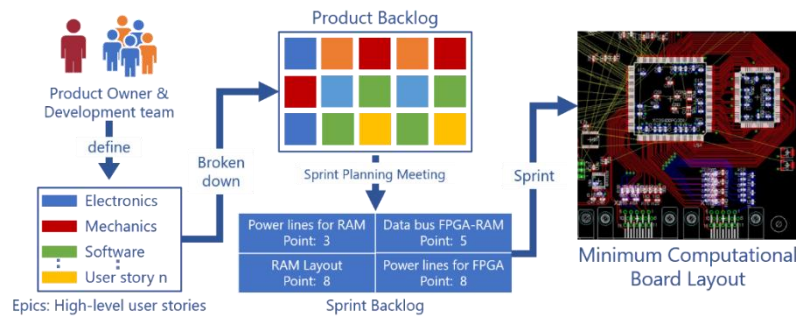


Figure 17. Example of workflow to realize an MVP, source (Garzaniti et al., 2019b)

Such an approach allowed the team to better align with the customer and investigate potential AIT issues in the early stages of the development (Deininger et al., 2019; Schmidt et al., 2018a). This latter point, we believe, is the most valuable and where Agile for hardware can make the difference. Prototyping can help de-risking interface issues, manufacturability, or assemblability problems, and the technology in general. The challenge will then become defining such MVP sequence to increase the maturity of the product subsets over time.

2.5.7 Hardware vs. Software: the question of the procurement

Scrum was initially conceived for software. In software, all the development depends on the team and relates to the Sprint structure. Therefore, the Scrum Master can constantly assess the development status, add or remove resources, and optimize the process. If procurement of external components is required, it is usually in the form of libraries, or software packages in general, that are already immediately (or rapidly) available and ready to use (Brhel et al., 2015).

In the hardware domain, most elements need to go through a lengthy procurement process, often involving manufacturing activities. Except in rare cases, the manufacturing is outsourced, thus outside the direct control of the team (Wei et al., 2021). This dependency introduces uncertainty. In worst cases, it may result in delays spanning over multiple Sprints and cost overruns (Garzaniti et al., 2019a). The problem can be partially mitigated by adopting Model-Based Systems

Engineering (MBSE) approaches and tools (Darpel et al., 2020; Meißner et al., 2021). However, this would make the process lose momentum, introducing additional complexity due to reconciling two quite far methodologies (Magdaleno et al., 2012; Ross et al., 2008).

Nevertheless, considering the procurement question cannot be avoided. A strategy to address delays and external dependencies with the supply chain may consist of scheduling the workflow accounting for suppliers' lead times and conceiving different scenarios with shifted AIT Sprints depending on early or late delivery of components. However, it would require having a dependencies structure of all the user stories to do so.

2.5.8 Testing and the question of quality assurance

A critical phase in hardware systems development, and specifically in space products, is Verification and Validation (V&V). Scrum and Agile, by nature, lack this feature by design because, according to Agile theory, testing as a stand-alone phase can be avoided in favor of incremental development and continuous feedback (Douglass, 2016b). Using the product from early phases, customers provide feedback on functionality and typically report bugs and problems. However, it is not acceptable to replicate the same approach for space flying products. Rigorous testing is essential to identify issues that cannot be fixed after the spacecraft is launched in orbit.



Figure 18. Qualification model vibration test (left), proto-flight model TVAC test (right)

In the use case presented in this chapter, the team introduced *testing user stories* for each verification activity expected. Testing tasks were introduced at multiple levels, during the design, using simulations, and during assembling and integration, using rapid prototyping or proto-flight model development approaches. For instance, the team has dedicated an entire Sprint to test a set of subsystems, with user stories fully dedicated to testing (Figure 18).

2.5.9 Reactivity to unforeseen design changes

Flexibility is one of the strengths of Scrum and Agile in general, making design changes during project implementation easier than traditional development approaches (Diebold & Dahlem, 2014).

For instance, in the use case, a later modification happened in adopting the RS422 bus for internal communication and control of the different payloads. While the generic bus characteristics were discussed during the Preliminary Design Review (PDR), such as choosing the data protocol and the power interfaces, the consortium did not analyze in-depth all the implications of connecting multiple payload interfaces to the same bus. Solving the problem required a hardware modification when electronics boards were already finalized and sent for manufacturing. A highly modular design and task flexibility allowed the team to quickly solve the issue, update the schematics and layouts, and send the new version for manufacturing (Schuh et al., 2017). That would have probably been harder in a traditional stage-gate scenario (Fricke & Schulz, 2005).

2.5.10 The human factor

Even though significant efforts were devoted to quantitative evaluating process performance, with numerous attempts in adapting team behavior, it is clear that significant uncertainties remain when dealing with people.

Task completion honesty

The reporting on the status of each activity was managed by the person in charge of the activity itself. The situation led to arbitrary definitions of the “completion” concept that were not meeting both the Sprint objective and the user story goal. While figuring well in numbers on the tracking tools, this approach generally resulted in a 32% increase in time compared to similar tasks (same score) where a more rigorous evaluation was adopted. This scenario has been observed on 12 tasks over the project.

Sprint duration and task spreading

Dealing with hardware or having external interdependencies may introduce complexities that require allocating some activities over multiple Sprints. An example is the purchase of electronics components that, including order placement and shipment, can take up to 2 weeks. While the most obvious approach would be to spread the task among multiple Sprints, this turned out to be a bad strategy. In addition to damaging the integrity of task completion analysis and complicating accurate statistics, the situation led to a mathematical worsening of story points estimation on future Sprints, with an impact that reached a 17% error in full Sprint complexity (Figure 15, Figure 16).

Data suggests that the practice adopted by the team to mitigate the issue was to plan for longer Sprints to accommodate all necessary activities (Figure 14). However, there is an upper limit when the process would unduly diverge from Scrum becoming de facto a traditional stage-gate.

Time-based scoring

The first approach adopted was a Fibonacci sequence scoring system, evaluating task complexity based on people's votes. This voting method proved to be not very effective in sizing the time required for completing a task. For this reason, the team has moved to a more straightforward time-based scoring system. This method was used from Sprints 5 to 7 and seemed to correspond to increased points per day completed (as shown in Figure 15 and Figure 16).

2.5.11 Agile manifesto vs. complex hardware systems

Building on lessons learned in the implementation of Scrum in the use case, Table 3 summarizes potential conflicts between Agile manifesto values (Beck et al., 2001) and space systems, or complex physical systems in general, development (Garzaniti et al., 2019b; Golkar et al., 2019).

Table 3. Agile manifesto vs. hardware systems development

Agile Manifesto	Potential Conflicts
Individuals and interactions over processes and tools	Complex systems development, such as space products, is traditionally highly process-driven due to the high capital costs involved and required mission assurance standards.
Working software over comprehensive documentation	Documentation is critical to ensure collaboration and avoid any misunderstanding among multiple organizations and complex supply chains.
Customer collaboration over contract negotiation	Mission requirement documents (MRD) and System requirement documents (SRD) are typically part of the contractual agreement between customers and mission integrators. MRD and SRDs are frozen under configuration control at the initial phases of the development.
Responding to change over following a plan	Cost of changes increases significantly at later stages of the development due to the high costs involved in rework or requalification of space hardware.

2.5.12 Catalog of gaps, challenges

Section 2.5 has provided a comprehensive picture of an Agile-Scrum implementation in an industrial setting. Table 4 summarizes the gaps and the challenges identified, reports potential threats, and highlights the needs of engineering teams.

Table 4. Catalog of gaps, challenges

ID	Gaps, challenges	Threats	Engineering teams needs
1	The scoring system is strongly linked to the human factor <i>Section 2.5.10</i>	scoring system subjective and affected by biases not allowing for reliable planning	A reliable scoring system correlated to time or cost estimates
2	Effective Sprint planning (length) <i>Section 2.5.5</i>	Schedule disruption	A method for defining the Sprint length based on activities inherent characteristics and Minimum Viable Product objectives
3	Effective Sprint planning (contents) <i>Section 2.5.5</i>	Sprint rescoping, cost and schedule overrun.	An architecture that models the dependencies among user stories, allowing for task prioritization and ensuring the inclusion of all required tasks.
4	Meaningful definition of Sprint objective (i.e., MVPs sequence structure) <i>Section 2.5.6 & 2.5.8</i>	Reworking the same product subset without improving the technology readiness of the system	A taxonomy to define the MVPs and map the MVPs over the product maturity evolution
5	Management of procurement and manufacturing <i>Section 2.5.7</i>	Schedule disruption, development process interruption, cost overrun	An architecture that includes procurement and manufacturing in the <i>product backlog</i> modeling their interdependencies with other user stories.
6	Resource allocation and leveling a non-homogenous development environment. <i>Section 2.5.5 & 2.5.10</i>	Unbalanced development teams, understaffing, and overhead of available personnel	A model the accounts for resource allocation and leveling during planning activities evaluating different project implementation scenarios.
7	Coordination aspects with different stakeholders such as consortium participants, customers, and suppliers. <i>Section 2.5.9</i>	Interface mismatches or failures between system elements	A model that consistently maps work packages (if any) and external input/output to user stories and accounts for their interdependencies

2.6 Summary of literature review and industry evaluation

Previous sections reviewed the bodies of knowledge framing the research work, offering an overview of the current state of the art. Sections 2.1 and 2.2 provided the theoretical and practical foundation of Agile theory, with a particular emphasis on Agile Scrum. Section 2.3 briefly discussed the literature on product development approaches with a particular focus on the Stage-Gate model, Agile for hardware and their combination. It also discussed currently available project management techniques, contextualizing them in the problem of interest. Section 2.5 reported the field research study conducted to get additional insights into current challenges in implementing Agile for hardware, thus better inform the review.

Following the literature review and the industry study, sections 2.4 and 2.5.12 presented the gap in the state-of-the-art that has been identified, shaping the contribution of this work. Specifically, gaps in current Agile theory, as well as Agile relation/combination with the traditional approaches, have been identified.

The following list reports the gaps in the Agile theory mapping them to the challenges faced by the industry and summarized in Table 4

- a) Lack of a taxonomy for Minimum Viable Product definition that accounts for technical and development process aspects (as presented in section 2.2.1 and Section 2.2.3, mapped on the industry challenge Table 4, ID 4)
- b) Lack of a method to support the tasks' prioritization and Sprint planning activities that, accounting for user stories interdependencies, minimizes the Sprint rescoping (as presented in section 2.2.4, mapped on the industry challenges Table 4, ID 1 & 3).

- c) Lack of a method to reliably evaluate the Sprint workload accounting for resource availability and allowing for proper resource allocation and levelling (as presented in section 2.2.4, mapped on the industry challenge Table 4, ID 2 & 6).
- d) Lack of methods to model and evaluate the impact of iterations propagation through the development process, thus inform on the viability of given iterations (as presented in Section 2.2.2, mapped on the industry challenges Table 4, ID 3 & 5).)

The following list reports the gaps in the Agile relation/combination with the traditional approaches mapping them to the challenges faced by the industry and summarized in Table 4

- a) Lack of theories to reconcile traditional and Agile approaches (as presented in Section 2.3.2, mapped on the industry challenge Table 4, ID 7).
- b) Lack of quantitative metrics to evaluate the viability of implementing the different product development processes (as presented in Section 2.3.2 and Section 2.2.2, mapped over all the industry challenges presented in Table 4).
- c) Lack of methods to coordinate the interdependencies between different organizations implementing different development processes, accounting for the complex interplay between all the stakeholders (as presented in Section 2.3.2 and Section 2.2.2, mapped on the industry challenges Table 4, ID 5 & 7).

This thesis aims to fill those gaps enriching current Agile theory and developing a decision support system (including methods and tools) to support project managers and engineering teams in overcoming the challenges mentioned above.

This page intentionally left blank

*“All models are wrong but
some are useful”.*
(Box, 1979)

Chapter 3

A decision support system

Chapter 1 described the motivation for this research and outlined the research goals. Specifically, it has been identified the need for a decision support system to assist project managers and design teams in structuring and planning the Agile or Hybrid-Agile development process, navigating programmatic and technical tradeoffs. Chapter 2 surveyed the literature identifying challenges in implementing Agile development for physical products, setting the boundary condition for our decision-making problem.

This chapter addresses the need identified in Chapter 1, developing the decision support system. This decision support system for Agile development of hardware systems (CURSIVE) includes an analytical approach to managing development activities within a hardware project. CURSIVE consists of three macroblocks: *structuring*, *simulating*, and *planning*, implemented in an integrated tool (Figure 19). This setting allows CURSIVE to deal efficiently with typical projects structure (Archibald, 2003; ISO/IEC JTC 1/SC 7, 2015)

The *structuring* macroblock refers to the methods and tools for reasoning about the structure of the decision problem. If we consider the development activities as our decision variables, the structuring problem includes: defining or evaluating the interconnections between the

different decision variables (tasks dependencies); determining the order in which decisions are to be addressed (tasks order); evaluating the quality of the variables in describing the problem of interest (tasks granularity).

The *simulating* block investigates the feasible solutions satisfying the problem constraints and evaluates the overall process performance for different variables combinations and values.

The *planning* macroblock refers to methods and tools for further investigate potential target solutions offering an actionable plan.

These macroblocks are integrated with a cross-block layer responsible for knowledge representation. The *representing or viewing layer* (different names are used depending on whether we refer to input or output data) includes methods and tools to formally represent the problem in a way understandable by decision-makers and interpretable by computers. This layer is also responsible for presenting decision-support information derived from structuring, simulating, and planning blocks in a human-understandable format.

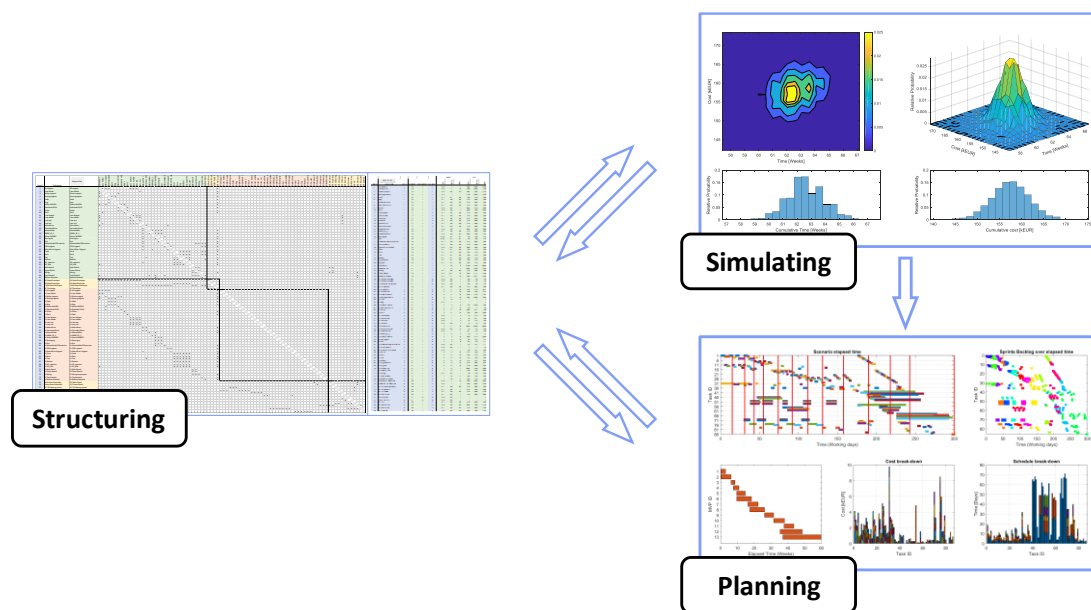


Figure 19. CURSIVE macroblocks

Figure 20 summarizes the overall workflow of the proposed framework. Each macroblock consists of a set of activities (square shapes) producing a set of artifacts (rectangle with a wavy base). Additional data (rhomboid shape) might be required to perform some activities. In the following sections, we describe in detail each of the proposed steps.

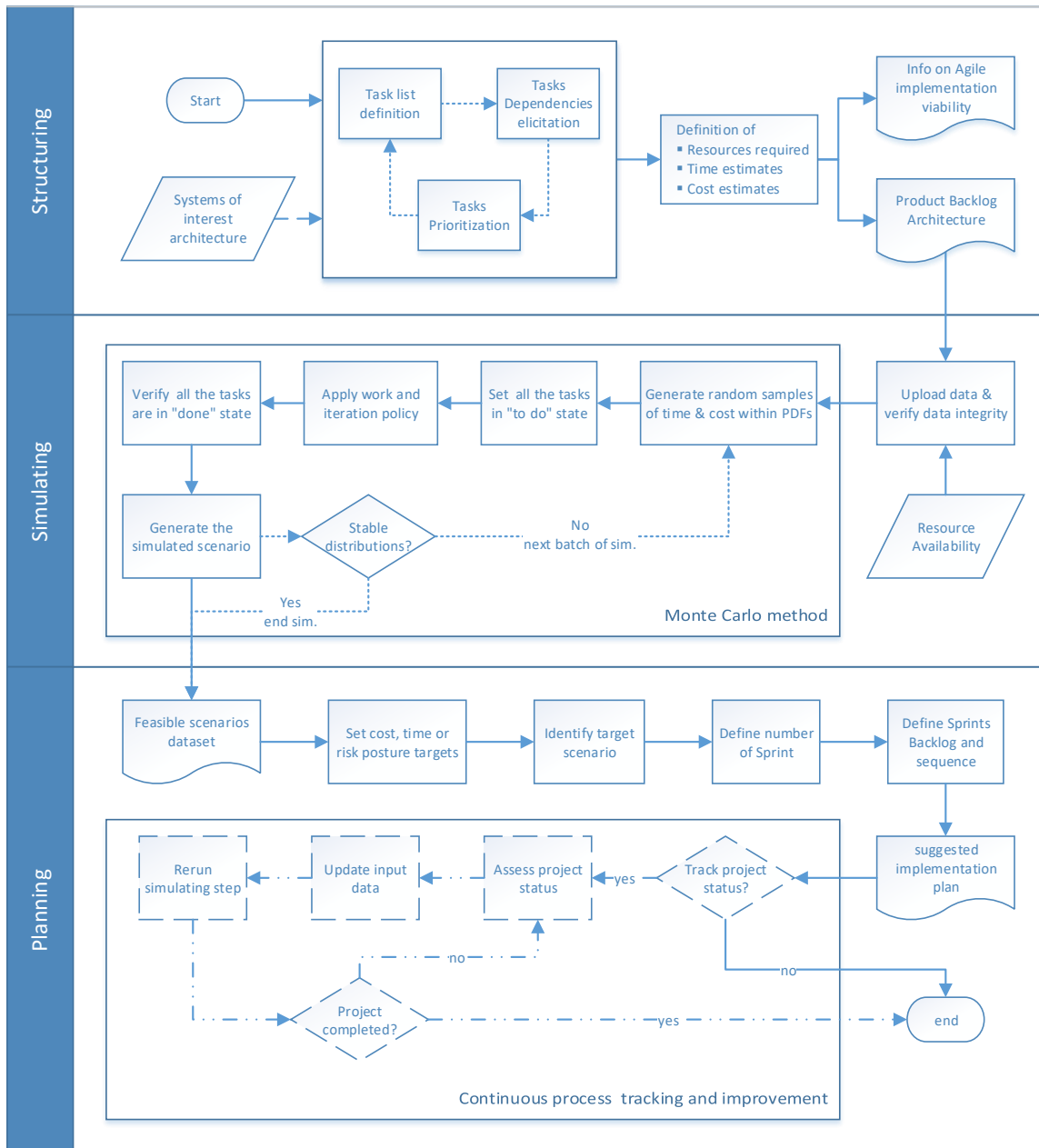


Figure 20. Workflow of proposed framework

3.1 Structuring

This first stage of CURSIVE aims to overcome some of the critical challenges identified in Chapter 2. With reference to the catalog of gaps summarized in Table 4, this section addresses the following challenges:

- Providing a reliable scoring system correlated to time or cost estimates (ID 1);
- Defining an architecture model to characterize the dependencies among user stories, allowing for task prioritization and ensuring the inclusion of all required tasks (ID 3);
- Introducing procurement and manufacturing activities in the architecture of the *product backlog* modeling their interdependencies with other user stories (ID 5);
- Defining a model to consistently map work packages (coming from an upper organizational layer, if any) and external input/output to user stories and accounts for their interdependencies (ID 7);

By overcoming those challenges, this thesis contributes to filling the gap in Agile theory and its coordination aspects with traditional product development processes identified in Section 2.4. The approach proposed in this first stage significantly contributes to answering the first research question (specifically Section 3.1.5) and contributes to answering the structuring problem of the second research question (particularly Section 3.1.1).

3.1.1 *Product backlog architecture*

Agile does not have a process architecture as typically defined in systems engineering literature (Eppinger & Browning, 2018) but relies on the product backlog leaving the structuring exercise to

the team. This situation entails several challenges and adds uncertainties to the process (e.g., Sprint rescope, schedule disruption, development interruption, interface mismatches)

To overcome these challenges, leveraging the existent product backlog artifact, this thesis introduces the concept of *product backlog architecture* defined as “the collection of the user stories supplemented by the elicitation of their interactions”.

The design structure matrix (DSM) is used to represent the product backlog architecture in a compact yet exhaustive format (Eppinger & Browning, 2018). A DSM is a square matrix in which cells on the diagonal represent the tasks, while marks in off-diagonal cells indicate activities interfaces. For each task, marks in the row indicate its inputs (sub-diagonal marks), while marks in the column define its feedbacks (super-diagonal marks). Activity names or acronyms are typically listed in an additional column on the left of the matrix (Figure 21).

	VD	LH	Mtr_s	Brg_s	Shaft	Ring	Motor 1	G_h	Pinion	Rack	Lb	Lm	Lt	Lo	MM	SM	Mh	Motor 2	Brg tip	Nut	MH PCB_c	PEEK_s	M M Supp	OS F	OS B	OS T	OS B	OS_VD_o	OS_VD_s	MB	PB	T Diss	B Diss		
VD	•																																		
LH	•	•																																	
Mtr_s	•		•																																
Brg_s	•			•																															
Shaft					•																														
Ring						•																													
Motor 1	•						•																												
G_h								•																											
Pinion									•																										
Rack										•																									
Lb											•																								
Lm												•																							
Lt													•																						
Lo														•																					
MM															•																				
SM																•																			
Mh																	•																		
Motor 2																		•																	
Brg tip																			•																
Nut																				•															
MH PCB_c																					•														
PEEK_s																						•													
M M Supp																							•												
OS F																								•											
OS B																									•										
OS T																										•									
OS B																										•									
OS_VD_o																											•								
OS_VD_s																												•							
MB																																			
PB																																			
T Diss																																			
B Diss																																			

Figure 21. Notional example of DSM product backlog architecture

The DSM provides a simple way to visualize the structure of an activity network and compare alternative process architecture. The scientific literature provides several methods as well as algorithms to analyze and optimize process DSM models (Eppinger & Browning, 2018).

Within the product backlog definition, project participants are also asked to provide additional data about time and cost (discussed in section 3.1.2), resources required to perform the activities (discussed in section 3.1.3), and Work Breakdown reference from the consortium plan, if any (discussed in section 3.1.4).

3.1.2 Scoring system: time and cost estimates

One of the key elements for efficient planning, thus a successful implementation of Agile, is having reliable task effort estimates. Agile frameworks do not prescribe a unique method for teams to quantify the efforts, but they generally use some abstract metrics. Standard estimating practices include numeric sizing, t-shirt sizes, and the Fibonacci sequence.

While the implementation process of such methods undoubtedly represents an effective team-building activity, the outcome is not always meaningful. The team members might not always share the same understanding of the metric or scale used. Furthermore, in hardware projects, where different disciplines are involved, complexity indexes might not be comparable.

This thesis proposes to use a time and cost-based scoring system as a common language shared by the whole team across all the disciplines. Moreover, instead of deterministic estimates, stochastic variables are used to account for uncertainty.

To define those estimates, a combination of expertise-based approaches and data-driven approaches are used. Expertise-based approaches are adopted when no quantified, empirical data are available. They represent a practical, low time-consuming, and efficient solution. In this case,

the development team is asked to provide a set of time and cost estimates in a Delphi-like method (Nowack et al., 2011; Rowe & Wright, 1999). Data-driven approaches are used instead when statistics from previous development projects or detailed information on the current one is available.

These estimates have to be provided with their probability distribution to represent the uncertainty associated with activity time and cost. The literature proposes different probability density functions (PDFs) to represent such uncertainty (Hajdu & Bokor, 2014). The model proposed here adopts triangular distributions, which are simple to estimate from typically available programmatic data. Their definition requires three data points per task: Lower Boundary (LB), Most Likely Value (ML), and Upper Boundary (UP). The area under the probability density functions (PDF) has been normalized and set equal to one. The model assumes that task durations are not interdependent, and each activity's time PDF also accounts for internal reworks. The expected PDP time is build based on all the considerations above.

Since cost is strongly related to time, the cost PDF for a given activity usually has a shape similar to the schedule one. We define for each task a correlation function based on the activity cost-time elasticity and organization capability (e.g., supply chain activities - longer lead time is usually associated with lower cost; design activities – mainly OPEX).

The boundaries of cost estimates are based on historical data or expert opinion and can be deterministic or random variables. The overall PD cost is evaluated depending on the PDP architecture accounting for all activities interactions. PDFs provide additional information on the probabilities of different outcomes expressing a perception of the uncertainty. These estimates are summarized in the *product backlog architecture* as additional columns alongside the DSM (Figure 22).

Task ID	Prepare Data Task Name	WP ID	Expertise	# Exp needed	weeks			Currency		
					min	mode	max	min	mode	max
1	VD	1	M	1	0.6	1	2	960	1600	3200
2	L H	1	M	1	0.6	1	1.4	960	1600	2240
3	Mtr_s	1	M	1	0.4	0.6	0.8	640	960	1280
4	Brg_s	1	M	1	0.8	1	2	1280	1600	3200
5	Shaft	1	M	1	0.1	0.2	0.6	160	320	960
6	Ring	1	M	1	0.2	0.4	0.6	320	640	960
7	Motor 1	1	M	1	0.2	0.4	0.6	320	640	960
8	G_h	1	M	1	0.2	0.4	0.6	320	640	960
9	Pn	1	M	1	0.4	0.8	1	640	1280	1600
10	Rk	1	M	1	0.4	0.6	1	640	960	1600
11	Lb	1	O	1	0.2	0.4	0.6	320	640	960
12	Lm	1	O	1	0.2	0.4	0.6	320	640	960
13	Lt	1	O	1	0.2	0.4	0.6	320	640	960
14	Lo	1	O	1	0.2	0.4	0.6	320	640	960
15	MM	1	O	1	0.2	0.4	0.6	320	640	960
16	SM	1	O	1	0.2	0.4	0.6	320	640	960
31	Main Board	1	E	1	2	3.6	4	3200	5760	6400
32	Power Board	1	E	1	1	1.8	2	1600	2880	3200
40	Pr VD	3	p	1	8	12	16	936	1170	1697
41	Pr L H	3	p	1	8	12	16	760	950	1378
42	Pr Mtr_s	3	p	1	8	12	16	696	870	1262
43	Pr Brg_s	3	p	1	8	12	16	800	1000	1450
44	Pr Shaft	3	p	1	3	4	8	2	2	5

Figure 22. Complementary information provided within the product backlog architecture work-packages traceability, expertise required, cost and time estimates.

3.1.3 Resource availability and disciplines involved

Agile for hardware significantly differs from software version because of non-homogenous development environments requiring expertise from different disciplines. For this reason, while formulating user stories is also important to define the field of knowledge associated with them and include in the team all the expertise needed.

CURSIVE includes expertise information in the *product backlog architecture* alongside time and cost estimates (Figure 22). To be highlighted, it does not pre-assign the task to people but only marks the discipline related to the task, including internal resources, i.e., development team, and external resources, i.e., procurement and manufacturing. Those data will be used during the project execution simulation, ensuring correct resource allocation and leveling.

3.1.4 Hybrid-Agile in multiparty consortia

A multi-tier architecture has been formulated to manage a Hybrid-Agile PDP ensuring efficient coordination between the consortium adopting stage-gate and the participants implementing Agile (Garzaniti et al., 2019b). The key feature of the proposed method is the *coordination interface* that reconciles the deliverables and activities of the two approaches (Figure 23).

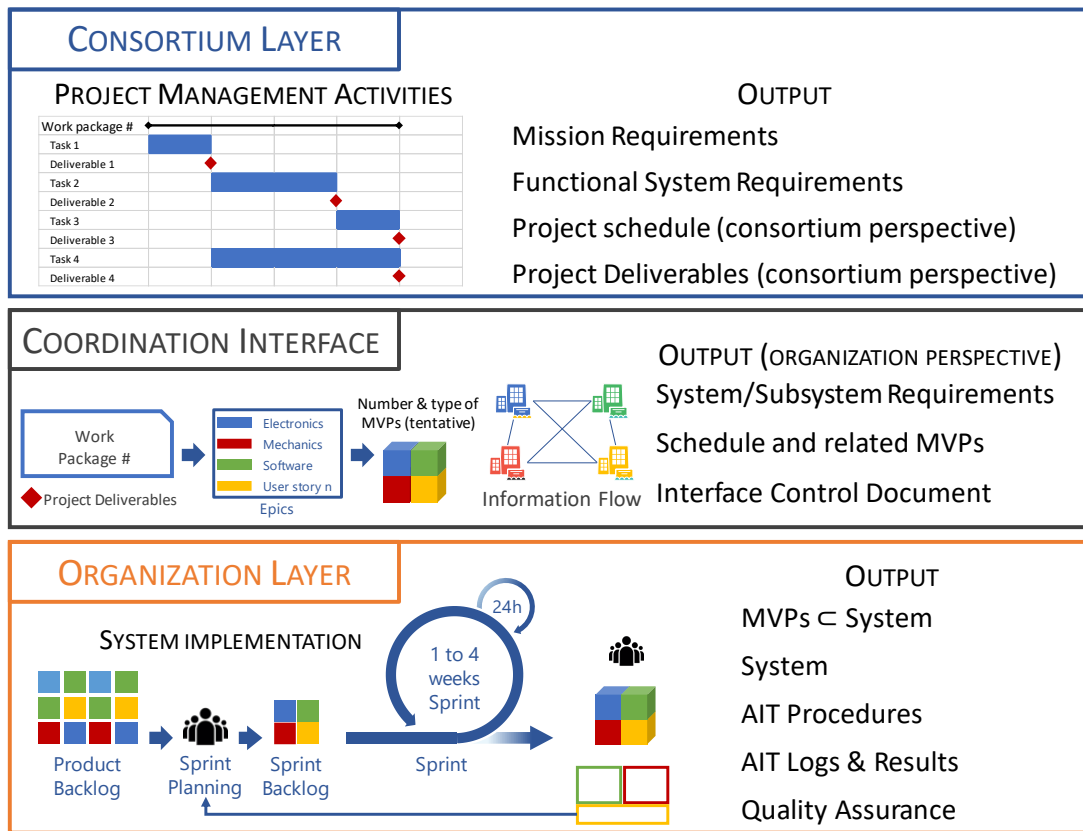


Figure 23. Hybrid product development process architecture

The *consortium layer* sits at the top layer of the architecture. The consortium, as a coordinating agent of the PDP, provides overall management of the project. It is in charge of the governance of the project. The consortium members collegially define the mission requirements, the functional system requirements, the interfaces among all parties involved in the project and agree on project reviews and deliverables. The consortium is also responsible for strategic decisions

throughout the lifecycle of the project. Consortia can operate using either Agile or stage-gate PDP. In our research, we focus on consortia operating using traditional Stage-Gate processes. Due to its structured nature, stage-gate provides natural means of coordination through decision gates and milestones, and it has been proven to work with large, complex organizations.

At the bottom layer of the architecture, we have the *organization layer*. This layer includes all the organizations participating in the project. Each project participant operates its own PDP and coordinates with others through interfaces with the consortium layer.

The coordination is implemented through a *coordination interface* defined at the organization level. Each organization participating in the project maps the consortium work packages, milestones, and deliverables on their internal means of project management, such as product backlog and minimum viable products (MVPs) in the case of Agile. This layer can also implement information coordination through direct interfaces between the participating organizations. However, we neglect the latter in the current setting, focusing on the main structured means of coordination.

3.1.5 Agile implementation viability

Two cornerstones of Agile methodologies, and specifically the Scrum version, are the iterative enhancement of products and the implementation of projects through a sequence of Sprints (Schwaber & Sutherland, 2020). While these distinctive features of Agile are highly beneficial for getting early customer feedback and derisking products from both technical and business perspectives, they might come at not negligible cost or time expense. Furthermore, the implementation of some hardware-specific activities within a Sprint timebox (typically ranging from a couple of weeks to one month) can be challenging, if not unfeasible at all, due to the physical aspect of the system (Garzaniti et al., 2019a).

Therefore, the Agile viability indexes (AV) are introduced. They are a set of non-dimensional metrics, ranging from 0 to 1, aimed at assessing the viability of implementing Agile in a given project based on time (AV^T) and cost (AV^C) data.

The time-based index provides information at the task level (1) and project level (2), accounting for both Sprint feasibility (SV) and iteration viability (IV). These two core aspects of Agile development, Sprint feasibility and iteration viability, are averaged into a unique metric (AV^T). The task-specific index AV_i^T uses the most likely value of time estimates, T_{MLi} , expressed in weeks as reference. The threshold level adopted in eq (1) formulation relates to the recommended maximum Sprint, i.e., four weeks. This value is consistent with the latest Scrum guide (Schwaber & Sutherland, 2020), stating that Sprints “are fixed length events of one month or less to create consistency”. For T_{MLi} values higher than recommended Sprint length, AV_i^T focuses on providing information on the extent to which a Sprint is unfeasible.

$$AV_i^T = \begin{cases} \frac{1}{2}(SV_i + IV_i) & \text{if } T_{MLi} \leq 4 \\ SV_i & \text{if } T_{MLi} > 4 \end{cases} \quad (1)$$

$$AV_{prj}^T = \frac{\sum_n AV_i^T}{n} \quad (2)$$

AV_{prj}^T provides a rough indication of the most suitable development process for the considered project: pure Agile ($AV_{prj}^T > 0.75$), Hybrid-Agile ($0.25 < AV_{prj}^T \leq 0.75$), or stage-gate ($AV_{prj}^T \leq 0.25$). The thresholds are correlated to the indexes formulation contributing to the AV^T , as discussed in detail in the following paragraphs.

AV_i^T offers task-specific information ($i = 1, \dots, n$ with $n =$ number of tasks), supporting decision-makers in defining the detailed development process structure that best fits the project characteristics (e.g., identify activities for which iterations would introduce a high risk of schedule overrun; identify the set of activities implementable in Sprints).

The Sprint viability index (SV_i) assesses the feasibility of completing an activity within the Sprint time-boxed periods (3).

$$SV_i = \begin{cases} -0.250 \cdot T_{MLi} + 1 & \text{if } 0 \leq T_{MLi} \leq 2 \\ -0.125 \cdot T_{MLi} + 0.75 & \text{if } 2 < T_{MLi} \leq 4 \\ -0.25 \cdot \frac{T_{MLi} - \max(T_{MLi})}{\max(T_{MLi}) - 4} & \text{if } T_{MLi} > 4 \end{cases} \quad (3)$$

As mentioned before, T_{MLi} is the most likely value of time estimates of each task i expressed in weeks. It is used as a reference metric in calculating the index. Three different ranges are set for the T_{MLi} , formalizing the shared understanding among practitioners regarding Sprints duration, as reported in the scientific literature (Atzberger et al., 2020; Schmidt et al., 2018b, 2019), as well as the industry surveys (Age-of-Product.com, 2018; Saat Network GmbH, 2008, 2011). The maximum recommended length for a Sprint is four weeks, as reported in the latest Scrum guide (Schwaber & Sutherland, 2020). Beyond this threshold, the process falls back on traditional approaches. Below four weeks, shorter times are associated with higher Sprint efficiency, allowing for faster MVPs evolution, thus convergence to the final product. The function approximating the Sprint Viability index has been estimated based on the industry surveys data (Age-of-Product.com, 2018; Saat Network GmbH, 2008, 2011). Data reveal that practitioners' consensus has converged over the last decade in considering the two weeks Sprint as more effective, thus the widest adopted (Figure 24-a). Data also suggest that practitioners typically allocate from four to thirteen user stories per Sprint (Figure 24-b), with a team composed mainly of seven or more members (Figure 24-c).

Of course, those surveys exhibit few limitations. First, not all surveys report a careful and transparent demographics selection and analysis; therefore, data might be biased based on the job position and experience of the survey participants. Second, surveys address different sectors, thus

including different products with different levels of product physicality. Nevertheless, those also are the most recent and comprehensive data currently available.

Building on those data, the SV_i index has been formulated to yield a value between 1 and 0.75 for T_{MLi} lower than or equal to 2 weeks, a value between 0.25 and 0.75 for T_{MLi} between 2 and 4 weeks, and a value between 0.25 and 0 for T_{MLi} greater than four weeks. These ranges are then used to provide recommendations for both project and task-specific indexes: Agile ($AV^T > 0.75$), Hybrid-Agile ($0.25 < AV^T \leq 0.75$), or stage-gate ($AV^T \leq 0.25$).

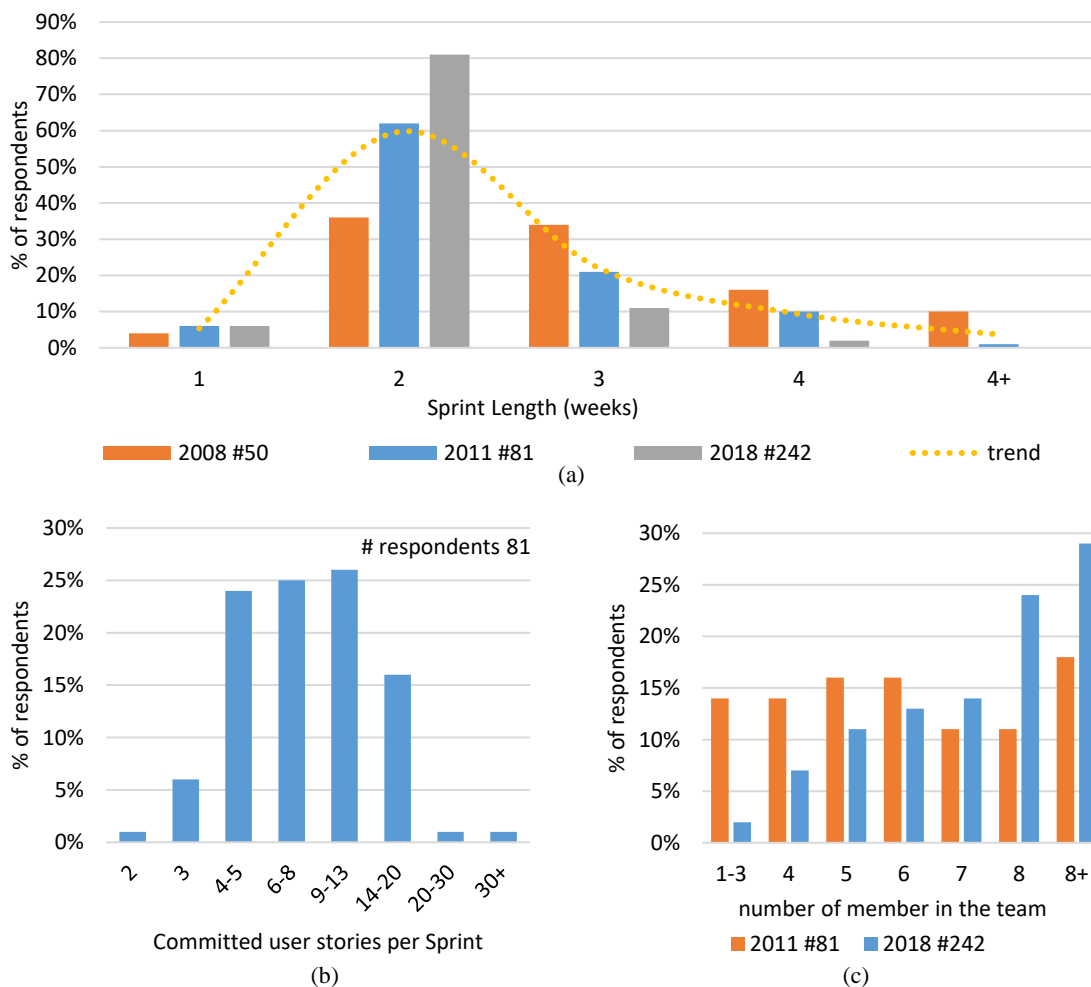


Figure 24. Survey data on sprint length, committed user stories and team composition

The iteration viability index (IV_i) evaluates the relative weight of a task over the project (4). It provides an indication of task granularity, thus the effect of a potential iteration of the given task on the project schedule. The IV_i index ranges from 0 to 1, and it is formulated to provide a closer to one value for shorter T_{MLi} (one corresponds to the shorter task in the project) and a closer to 0 value for longer T_{MLi} . This formulation, leveraging the relative weight of a task over the product development, allows for a project-independent index that can be used across a variety of projects.

$$IV_i = 1 - \left[\frac{\frac{T_{MLi}}{\sum T_{MLi}} - \min\left(\frac{T_{MLi}}{\sum T_{MLi}}\right)}{\max\left(\frac{T_{MLi}}{\sum T_{MLi}}\right) - \min\left(\frac{T_{MLi}}{\sum T_{MLi}}\right)} \right] \quad (4)$$

The cost-based Agile viability index (5). accounts for the relative weight of each task compared to the average task (i.e., the median value of task cost samples \tilde{C}_{ML}). It provides information on the impact on the project cost of a potential iteration of the given task.

$$AV_i^C = \begin{cases} \frac{0.5 \cdot [C_{MLi} - \tilde{C}_{ML}]}{\min(C_{MLi}) - \tilde{C}_{ML}} + 0.5 & \text{if } C_{MLi} \leq \tilde{C}_{ML} \\ \frac{0.5 \cdot [C_{MLi} - \max(C_{MLi})]}{\tilde{C}_{ML} - \max(C_{MLi})} & \text{if } C_{MLi} > \tilde{C}_{ML} \end{cases} \quad (5)$$

The AV_i^C index ranges from 0 to 1, and it is formulated to provide a closer to one value for lower C_{MLi} (one corresponds to the less expensive task in the project) and a closer to 0 value for higher C_{MLi} . This AV_i^C formulation leveraging the relative weight of a task over to the average task allows for a project-independent index that can be used across a variety of projects. The cost-based index at the project level (AV_{prj}^C) can be derived using AV_i^C in (2). The same threshold values presented for AV_{pr}^T are applied to AV_{prj}^C to have a unified formulation. However, in the case of the cost indexes, the thresholds are intended as a rough indication. Agile teams might select a different threshold based on different reference metrics, such as a percentage of the R&D budget or the total project budget.

These metrics, providing a measure of tasks' intrinsic characteristics, can be used as a proxy to understand if both work and system decomposition (i.e., task granularity and system modularity) are suitable for Agile implementation. In support of this statement, there is a solid body of knowledge developed over the last twenty years relating product modularity and product and process granularity (Chiriac et al., 2011; Eckert et al., 2015; Maier et al., 2017; Sosa et al., 2003). Specifically, the reader can refer to two seminal research works discussing the relation between modularity and granularity in engineering systems design and their effect on process cost and time performance (Maier et al., 2017, 2015). A detailed discussion of those concepts is out of the scope of this thesis as it would require separate research. Instead, the work presented here leverages previous research in the field and adopts those concepts in the context of Agile development of physical products. The relation between those concepts and the Agile implementation suitability is further discussed in the case studies and summarized in their interim conclusion.

Agile viability indexes data are summarized in a chart as in Figure 25 for easy reading and interpretation. Based on these data, teams can refine user stories definition and have a first understanding of when and how to use Agile within the development project.

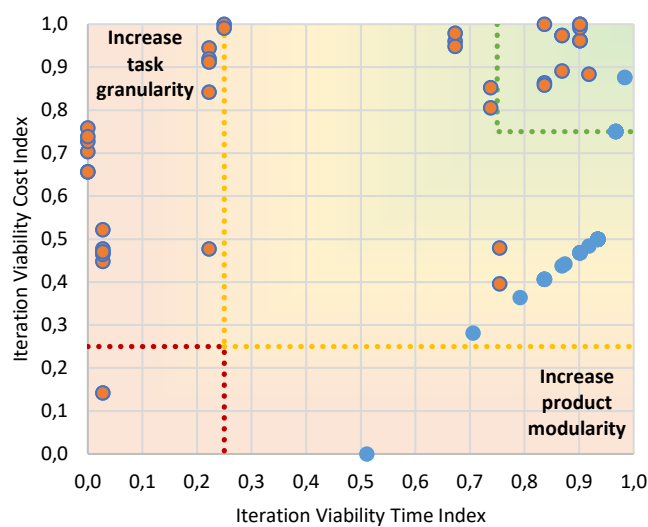


Figure 25. Agile Implementation Viability Chart each point represents a task

3.2 Simulation

In the framework proposed in this thesis, the product development process (PDP) is characterized as a network of activities exchanging information and deliverables (Figure 26). The network topology is defined by the *product backlog architecture*, which serves as an adjacency matrix. Activities generate an output based on given inputs. A change in the inputs requires revising the activity. The rework affects the outcomes of the activity and potentially propagates through other tasks.

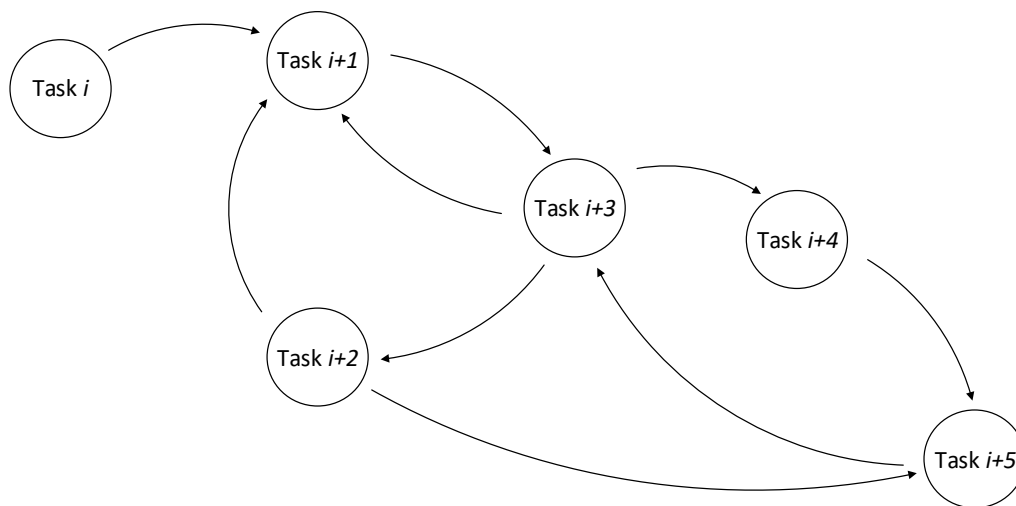


Figure 26. Notional example of activities network

While input changes always require revising the task (to verify the consistency of data exchanged), the impact of this new flow of information may vary. When an iteration occurs, an upstream task i is reworked due to feedback from downstream activities ($i + n$). The different impact of this rework is reflected in the time and the related cost required to complete the task again, as well as the number of downstream activities affected by it. The model also considers that iterate executions of the same activity take less time than the original duration (i.e., learning factor). The cost is estimated accordingly.

The model uses a discrete event simulation to compute PD process time and cost for different batches of inputs (Figure 27). Each simulation run begins at system state 0 (all tasks in the product backlog). Activities duration and the related cost for each simulation are randomly sampled within PDFs boundaries fitting to the probability distributions shapes (we have also implemented the Latin Hypercube method (Tang, 1993) if users prefer a different sampling approach).

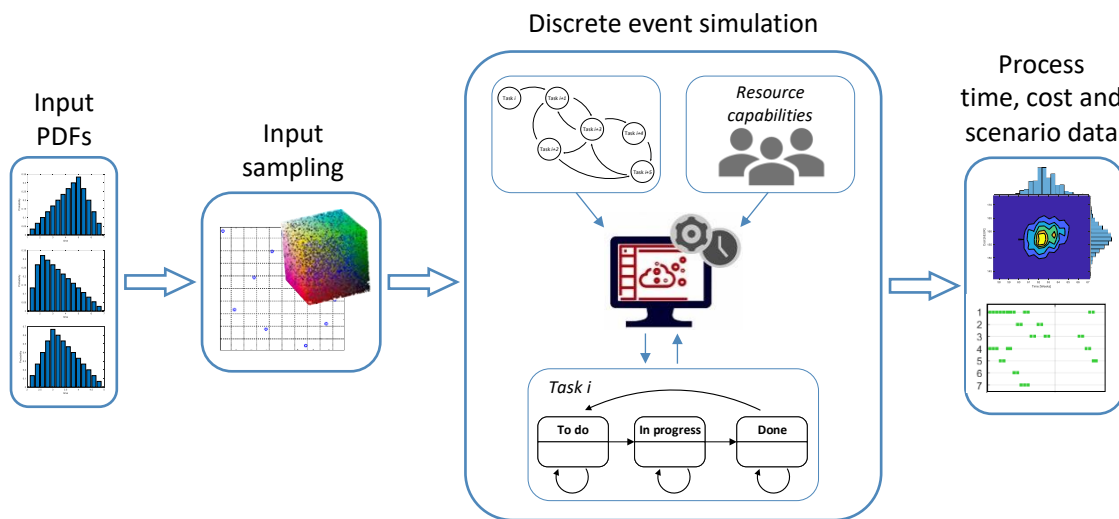


Figure 27. Schematic representation of simulation process

The simulation performs tasks using an *Agile Kanban board* (Rodríguez et al., 2018). Initially, all activities are in the “to do” state; the simulation runs till all the tasks move in the “done” state. In each system state, the model determines the activities to be done, verifying the availability/readiness of the inputs and the availability of resources. If input requirements are satisfied, the task is moved “in progress”. If input requirements are met, the task is moved “in progress”. The same task is then moved “in done” according to its working elapsed time. Once an activity ends, the cost of the work done is added to the cumulative cost. The total elapsed time represents the process duration.

Algorithm 1. Process execution for each run of the simulation

Input: task list, DSM, time and cost estimates, task type, resource

Output: Process Time, Process Cost, state transition sequence {Time} and {Cost}

def Task.state: to do=0, in progress=0.5, done=1

1 Randomly sample time and cost of each activity

2 Initialize variables and set all Task.state=0

3 **While** not all Task.state==1 **do**

4 | **For** all tasks in the tasks list **do**

5 | | Identify the tasks where Task.state=0

6 | | Identify the set of tasks that meet precedence and resources constraints

7 | | Set Task.state =0.5 and allocate the resources

8 | | Work on the tasks (increment Task.progress by time step t)

9 | | **If** Task.progress=100%, **then**

10 | | | set Task.state=1 and empty the resource

11 | | **End**

12 | | **If** Task.type==(procurement or AIT) and alert is present **then**

13 | | | Set Task.state=0 and empty the alert

14 | | **End**

15 | | Look for iteration in upstream activities generated by completed tasks

16 | | **If** an iteration is triggered **then**

17 | | | Set the upstream Task.state=0

18 | | | Update time and cost estimates samples

19 | | | propagate iteration effect to procurement and AIT tasks

20 | | | **If** those AIT and procurement Task.state==1 **then**

21 | | | | Set Task.state=0

22 | | | | Update time and cost estimates samples

23 | | | **Else**

24 | | | | Set a procurement or AIT alert

25 | | | **End**

26 | | **End**

27 | **End**

28 **End**

29 **Return** Process Time, Process Cost, state transition sequence {Time} and {Cost}

The simulation adopts a Monte Carlo approach (Theodoridis, 2020). It runs several times and generates a set of pairs cost, C , and time, T , samples. The collection of all C and T samples respectively form cost and time distribution. Together, they constitute a joint cost-schedule distribution. Several runs (s) are required to get stable distributions. Batches of simulation, n , are run until both expected value and standard deviation of the T and C distributions stabilize within precision ε according to equations (6) and (7) reported here only for cost.

$$\frac{|\sigma_{C,s}^2 - \sigma_{C,s-n}^2|}{\sigma_{C,s-n}^2} < \varepsilon \quad (6)$$

$$\frac{|E[C_s] - E[C_{s-n}]|}{E[C_{s-n}]} < \varepsilon \quad (7)$$

The results of this phase are summarized in a chart reporting joint time and cost probability distribution plots of the simulated scenarios (Figure 28). The summary is handy to quickly compare process cost and time against budget and schedule target, inform on potential risks of schedule or cost overrun, and eventually support selecting the best value-at-risk solution to comply with project targets.

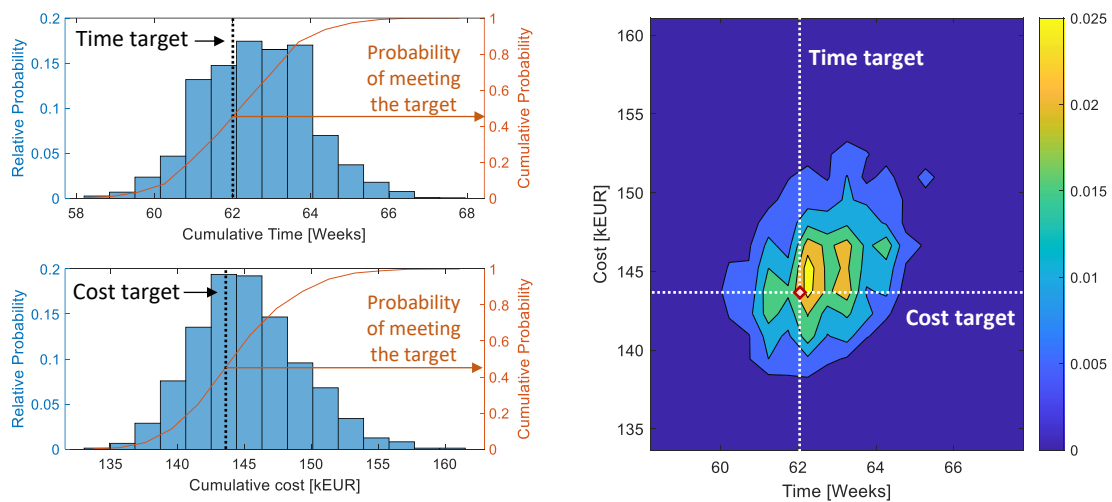


Figure 28. Output dashboard of scenario analysis

3.3 Planning

In most projects, budget and schedule targets are set a priori; therefore, agility in the PDP shall comply with these constraints. The objective is then to fine-tune all the process implementation variables to meet the cost and time requirements.

The challenges in this undertaking are 1) Defining an optimal number of Sprints; 2) Defining the Sprint backlog and duration 3) Defining the Sprint sequence, thus MVPs transition strategy. Results of this process are then summarized in a dashboard of easy interpretation.

The model proposed here considers the schedule associated with each scenario resulting from the simulation as a set of data points. Thus, the Sprint definition becomes a clustering problem. In statistics, and specifically in statistical and exploratory data analysis, clustering or cluster analysis is the task of grouping a set of objects (data points) so that the objects in the same group, called a *cluster*, are more similar to each other than those in other groups (Shannon, 2007).

Several methods are available to tackle clustering problems (Butta et al., 2021; Chander & Vijaya, 2021). Each of them involves some challenges in defining the set of parameters required by the given method to solve the clustering task. In this work, given the type of data we have to handle, we adopt a partitioning-based clustering technique, relying on squared error (Chander & Vijaya, 2021).

In such kinds of methods, the first step is to identify the optimal number of clusters. This is a recurring issue in clustering analysis and a separate question from the clustering process itself. Even if the literature addressed this concern (Davies & Bouldin, 1979; De Amorim & Hennig, 2015; Pimentel & de Carvalho, 2020; Ünlü & Xanthopoulos, 2019), there is still no definitive

answer to the question. The optimal number of clusters is somehow subjective and depends on the method used to measure similarities and the parameters used for partitioning.

To evaluate the optimal number of clusters, CURSIVE uses a heuristic method based on the *silhouette* score (Rousseeuw, 1987). Our algorithm evaluates the *silhouette* values for different numbers of clusters and finds the solution where adding a cluster no longer results in significantly better data modeling. The approach attempts to maximize the median of the silhouette over each cluster for a given number of clusters, minimizing its standard deviation. It has also been implemented the same heuristic method using a different metric: the *within-cluster sum of point-to-centroid distances*. The comparison of results derived from different metrics is used for validation purposes (Sugar et al., 2003).

As a second step, the framework uses the *fuzzy C-means algorithm* (Berget et al., 2008; Zhang et al., 2020) to define the Sprints backlog. Compared to crisp clustering methods, which assign every object to a unique cluster, fuzzy clustering provides estimates on the degree of membership of each object to each cluster (i.e., the probability of membership to the different groups). The *fuzzy C-means algorithm* (FCM) aims to minimize the objective function J as in (8)

$$J = \sum_{i=1}^N \sum_{j=1}^C w_{ij}^m d_{ij}^2 \quad m > 1 \quad (8)$$

where $w_{ij}^m \in [0, 1]$ are the membership values telling the degree to which element, x_i , belongs to the cluster, c_j , and d_{ij} are the distances between the point x_i and the centroid c_j . The index $i=1, \dots, N$ refers to the object number, while the index $j=1, \dots, C$ refers to the cluster number. The hyper-parameter m is called *fuzzifier* and determines the fuzziness of the clustering. Typically it is set equal to 2, as this value has proven to give good results with the FCM (Berget et al., 2008).

Data are clustered based on process elapsed time, cost, task dependencies, and resource availability. As inclusion criteria in the clustering process, thus Sprint definition, the Agile viability indexes are used. Excluded tasks are then mapped on the suggested Sprints. The outcome of the Sprint definition process offers information on the set of tasks to be performed within each Sprint, the Sprints' length and cost, and the resources required to perform the activities. Every Sprint has its own goal. Therefore, Sprints' lengths, as well as the Agile team, may vary within the development. The framework accounts for all those questions and suggests a solution that best fits the scenario constraints.

The last step in the Sprint planning process consists of defining the Sprint sequence (i.e., the MVPs delivery sequence) and the related transition strategy. Building on the cluster analysis results, the framework suggests an optimal Sprints sequence taking into account the Sprints interdependencies (including product interdependencies and relevant lifecycle phases such as design, procurement, assembly, integration, testing, and validation).

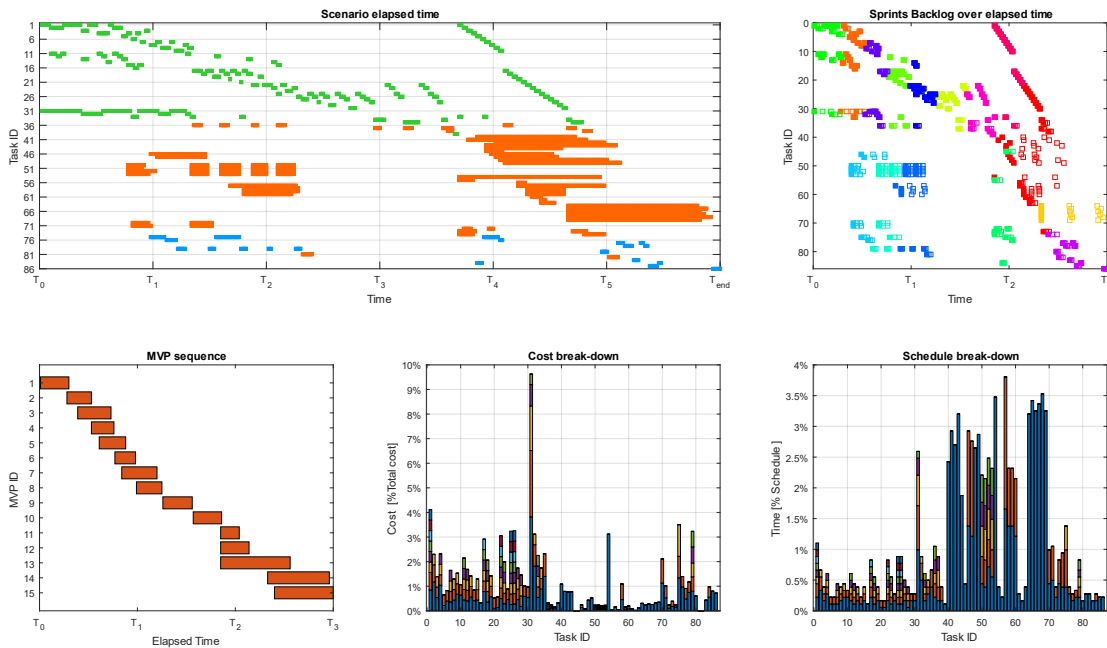


Figure 29. Output dashboard of the planning phase

The output dashboard (Figure 29) reports the traditional Gantt chart of the process (top left corner), a Sprint backlog mapped over the Gantt chart (top right corner), the number and sequence of Sprint (bottom left corner), and the detailed time and cost breakdown structure for activity sequence and iterations (bottom right side).

The approach proposed in this last stage answers the second research question, with a particular focus on the process execution support problem. With reference to the catalog of gaps summarized in Table 4, this section has addressed the following challenges:

- To provide a method for defining the Sprint length based on activities' inherent characteristics and Minimum Viable Product objectives. (ID 2).
- To provide a method for defining Sprint backlog, accounting for dependencies among user stories, allowing for task prioritization, and ensuring the inclusion of all required tasks (ID 3).
- To provide a method for managing procurement and manufacturing activities, accounting for interdependencies with other user stories (ID 5).
- To provide a method to handle resource allocation and leveling in a non-homogenous development environment (ID 6)

It shall be remarked that the dashboards provided by this system do not aim to replace team decision-making activities. Instead, the suggested Sprint backlog and MVPs sequence is meant to be used within the Sprint planning meetings as a starting point for the discussion. The solution proposed here aims to support the team's decision-making by offering quantitative analyses and specific metrics to benchmark all the possible alternatives.

3.3.1 Continuous process tracking and improvement

One of the distinctive characteristics of Agile methodologies is the adaption of the development process as the development progresses. CURSIVE accounts for it by including a continuous process tracking and improvement feature as illustrated in the Figure 20 flowchart and recalled in Figure 30 below.

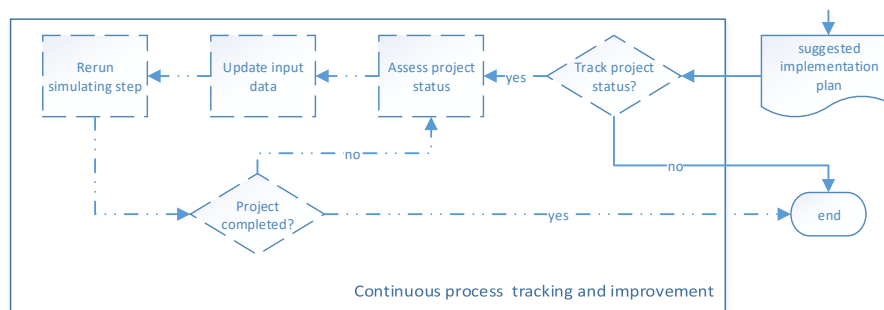


Figure 30. Continuous process tracking and improvement

Since CURSIVE is a data-driven model, the adaptation of the process as it progresses is ensured by updating the input data as they become available and rerunning the simulation and planning steps. While the project is running, end-users can first track and assess the project status by comparing the forecast with the actual implementation. Then, if major deviations occur, they define an updated version of the product backlog architecture by a) considering a subset of the initial product backlog, b) updating the task list, c) updating tasks dependencies d) updating the time and cost estimates, e) permutations of the above (Figure 31). Lastly, end users shall re-execute simulation and planning stages. Same principles and approaches on project structuring, Sprint planning and MVPs definition (as presented in sections 3.2, 3.3) are applied. The new output will offer an updated perspective on the PD, accounting for the new knowledge acquired while developing the product as well as the context evolution.

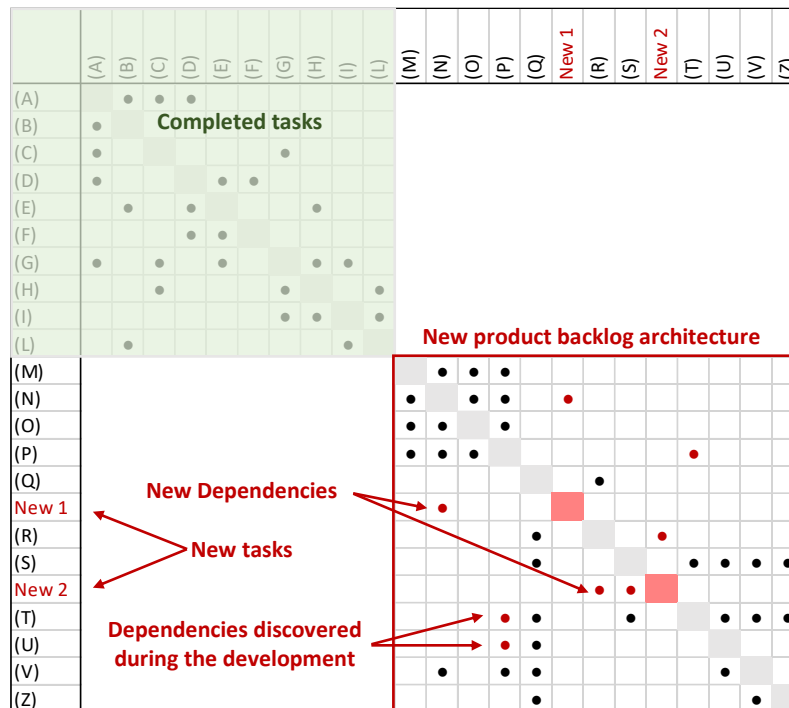


Figure 31. Product backlog architecture updates

The product backlog architecture follows the same rules of the Scrum product backlog artifact in terms of document dynamism, ownership, and maintenance strategy. The main difference relies on the elicitation of task interactions and the integration with quantitative time/cost estimates and engineering discipline information.

The framework proposed here also includes Sprint review and retrospective events as defined in the Scrum theory. Nevertheless, those two essential features are not discussed in detail because CURSIVE totally embraces the practice proposed in the Scrum guide (Schwaber & Sutherland, 2020). The main difference between traditional Scrum and CURSIVE is the aids used during the Sprints review and retrospective. The latter adopts structuring, simulation and planning tools, including visual aids as presented in the previous sections of this chapter.

3.3.2 MVP

As discussed in the literature review (Section 2.2.3), the MVP definition in the context of physical products is still an open question. The constraint of physicality hampers the implementation of a fully functional product increment at each iteration. Moreover, current literature still lacks perspective in capturing the MVP technical and development process aspects.

In an attempt to provide a formulation that fits better in the context of hardware systems and focuses more on the development process, we define the MVP as “*a complete and testable deliverable able to mitigate the technical risk associated with the product or a subset of it*”.

According to (Unger & Eppinger, 2009), *technical risk* is defined as the uncertainty related to whether a new product is technologically feasible and whether it will perform as expected, given precise product specifications. This uncertainty on both feasibility and performance can be mitigated at different levels of the product development process through verification activities. Each layer of verification activities will lead to an improvement in the maturity of the product till eventually reaching a system ready to be deployed in the operational environment. Therefore, it has been decided to use the verification and validation activities (thus the technical risk retired) as a proxy for the different levels of product maturity.

Given the capital intensity required to develop complex physical systems and the typically large procurement times, *Sprint planning* shall account for verification and validation activities performed on each MVP to ensure consistent improvement of product maturity over the Sprints sequence.

To provide engineering teams with a unified system to correlated MVPs (and their characteristics), verification and validation (V&V) activities, and product maturity, an MVP taxonomy has been developed (Table 5).

Table 5 Taxonomy of MVP

MVP			Acceptance criteria: objective on MVP	
Fidelity	Artifacts	Representation mode	V&V	Activities
Low	Diagram	Digital/Abstract	Verification	Analysis
Low	Numerical Model	Digital/Abstract	Verification	Numerical simulation
Medium/High	Digital Mockup	Digital	Verification	Analysis, Simulation
Medium	Physical Mockup	Physical	Verification	Physical inspection, Functional Test
Medium	Lab setup	Physical	Verification/Validation	Functional Test, Performance Test
High	Product Subset	Physical	Verification/Validation	Functional Test, Performance Test
High	Product	Physical	Validation	Performance Test, Day-in-the-life

Such a model, mapping verification and validation activities on the MVP artifacts, enables teams to trade off engineering efforts required to produce a given MVP and the risk retired by the Sprint outcome. Figure 32 offers a notional example of a potential tradeoff among Sprint length, V&V activities and MVP artifacts.

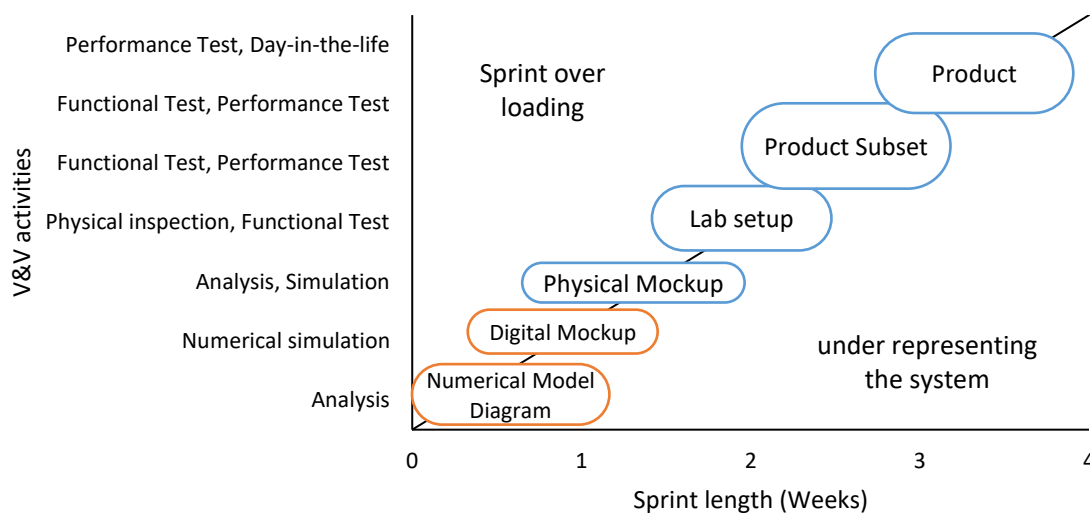


Figure 32. MVP mapping: V&V vs. Sprint length

3.4 Illustrative case

In this section, the decision support system is applied to an illustrative example. The objective is to demonstrate the use of such a system step by step. As an example case¹, it has been chosen a software-defined X-Band transmitter for CubeSat. Even if it is a relatively simple product, its development might not be that straightforward.



Figure 33. Artistic representation of an X-Band transmitter, source (EnduroSat, 2021)

3.4.1 Process Structuring

The first step consists of identifying all the activities required to develop such a system and define the dependencies among those activities. In structuring the process, the development team has to account for design, procurement or manufacturing, as well as verification and validation activities.

Starting from a system architecture model or a system block diagram, the development team can brainstorm on the activities required to develop the product and list them. For instance, the X-band transmitter would include a baseband signal processing unit (FPGA), a digital-to-analog converter, a baseband filter, a phase-locked loop oscillator, a mixer, an X-Band filter, some amplifier stages, and some input/output interfaces.

¹ The author would like to thank his colleague Simone Briatore for providing valuable insights and details on the input data used in developing this illustrative case.

Building on this list, the team can define activities related to the development of each component. When the team feels confident with the level of granularity of activities decomposition, it can start collecting the task in a DSM defining the interdependencies. Figure 34 summarizes the resulting *product backlog architecture*. It includes design activities (marked in green), procurement or manufacturing (marked in orange), and AIT activities (marked in blue).

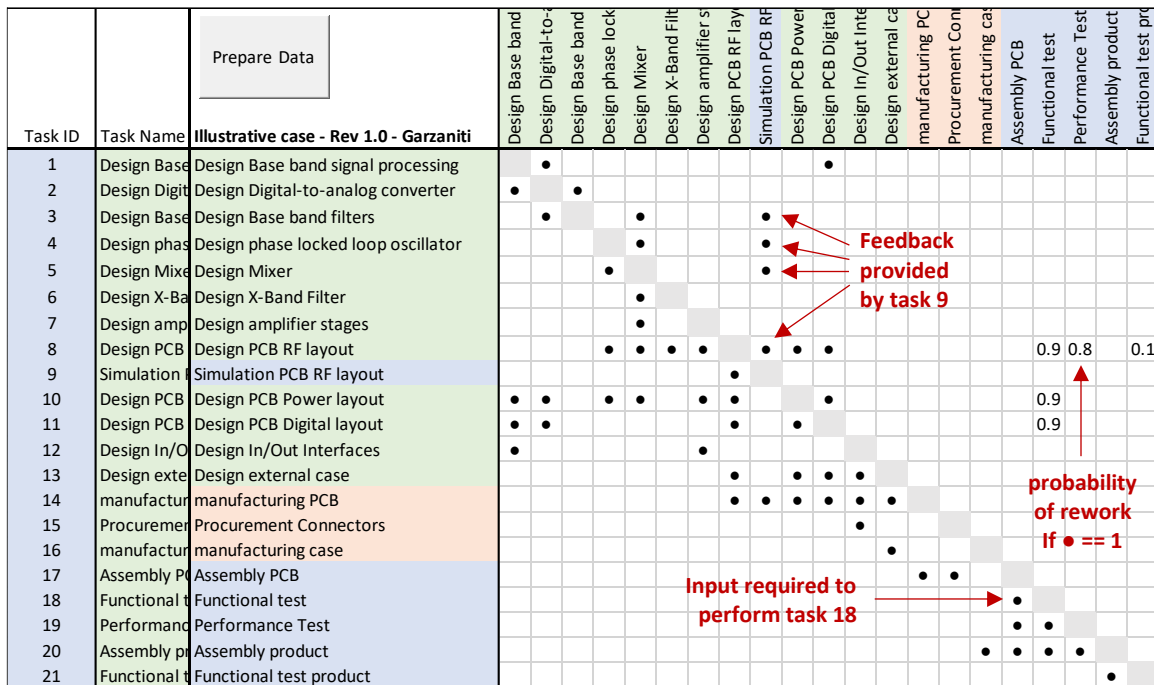


Figure 34. Illustrative case – product backlog architecture

As a second step, the team is asked to provide additional data about the time, cost, and resources required to perform the activities. The information can be based on historical data, experts’ opinion, or their combination. Each team member is asked to provide three values for both time and cost, corresponding to the optimistic value, the most likely value, and the worst-case value.

The team iterates on the effort estimation till it reaches the consensus. Then, the values are summarized in a table and attached to the product backlog architecture as complementary information (Figure 35). This data package constitutes the input for the simulation step.

Task ID	Prepare Data Task Name	WP ID	Expertise	# Exp needed	Exp code	time (day)			k EUR		
						min	mode	max	min	mode	max
1	Design Base band signal processing (FPGA)	1	E	1	1	5.0	5.0	10.0	1.0	1.0	2.0
2	Design Digital-to-analog converter	1	E	1	1	1.0	1.0	3.0	0.2	0.2	0.6
3	Design Base band filters	1	E	1	1	1.0	1.0	3.0	0.2	0.2	0.6
4	Design phase locked loop oscillator	1	E	1	1	1.0	1.0	5.0	0.2	0.2	1.0
5	Design Mixer	1	E	1	1	1.0	1.0	2.0	0.2	0.2	0.4
6	Design X-Band Filter	1	E	1	1	1.0	1.0	3.0	0.2	0.2	0.6
7	Design amplifier stages	1	E	1	1	2.0	3.0	5.0	0.4	0.6	1.0
8	Design PCB RF layout	1	E	1	1	5.0	5.0	10.0	1.0	1.0	2.0
9	Simulation PCB RF layout	1	AIT	1	5	5.0	7.0	10.0	1.0	1.4	2.0
10	Design PCB Power layout	1	E	1	1	2.0	3.0	4.0	0.4	0.6	0.8
11	Design PCB Digital layout	1	E	1	1	2.0	3.0	5.0	0.4	0.6	1.0
12	Design In/Out Interfaces	1	E	1	1	2.0	3.0	4.0	0.4	0.6	0.8
13	Design external case	1	M	1	2	1.0	2.0	4.0	0.2	0.4	0.8
14	manufacturing PCB	1	p	1	4	5.0	10.0	10.0	2.0	2.5	3.0
15	Procurement Connectors	1	p	1	4	3.0	5.0	10.0	0.1	0.1	0.1
16	manufacturing case	1	p	1	4	3.0	10.0	15.0	0.2	0.3	0.4
17	Assembly PCB	1	AIT	1	5	2.0	3.0	5.0	0.4	0.6	1.0
18	Functional test	1	AIT	1	5	5.0	5.0	10.0	1.0	1.0	2.0
19	Performance Test	1	AIT	1	5	5.0	5.0	10.0	1.0	1.0	2.0
20	Assembly product	1	AIT	1	5	1.0	2.0	2.0	0.2	0.4	0.4
21	Functional test product	1	AIT	1	5	2.0	3.0	5.0	0.4	0.6	1.0

Figure 35 Illustrative case – Complementary information attached to the product backlog architecture

While defining the product backlog architecture, CURSIVE provides the team with information on the viability of implementing Agile. The framework evaluates a viability index for each task in terms of time (Figure 36- left) and cost (Figure 36- right). Those data allow project participants to optimize the process structure already at this stage, before the simulation.

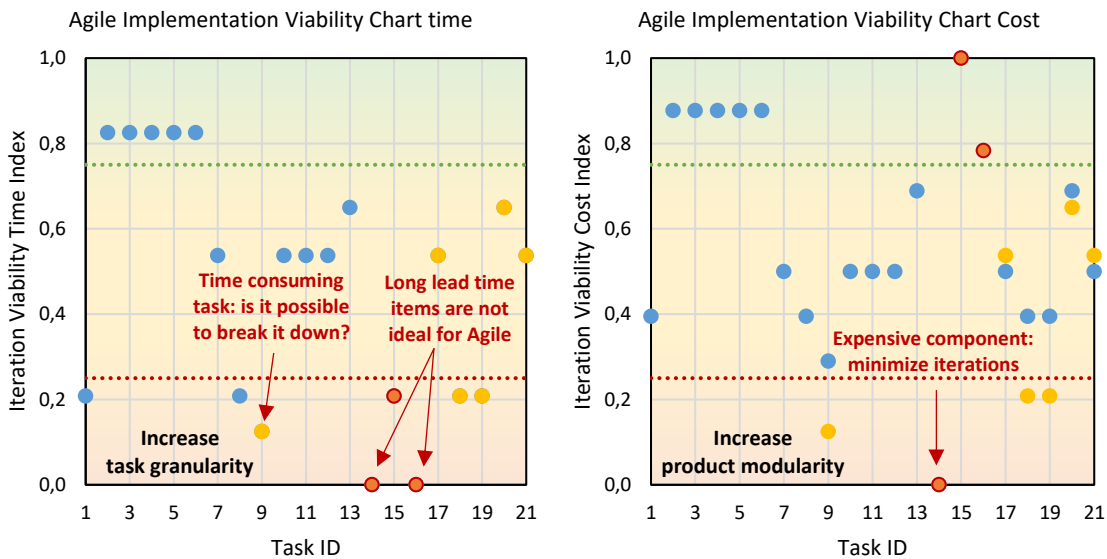


Figure 36. Illustrative case – Agile viability indexes

For instance, the procurement of the PCB seems to be inefficient in terms of both time and cost. A possible strategy is to increase the system modularity, meaning having different PCBs for the RF block and the power block. This would allow the team to iterate more on the specific PCB subsets at a lower cost. Naturally, this decision comes together with other technical considerations on the efficiency of the overall systems and performance implications.

For the illustrative case purposes, let us suppose that the team prefers to have a more integrated system because more compact and more power-efficient, thus keeping the process structure defined in Figure 34.

3.4.2 Simulation

CURSIVE runs 3000 simulations (in batches, s , of 500) to stabilize both mean and variance of time and cost distributions within precision, $\varepsilon = 10^{-4}$ (Figure 37), according to equation (6), (7).

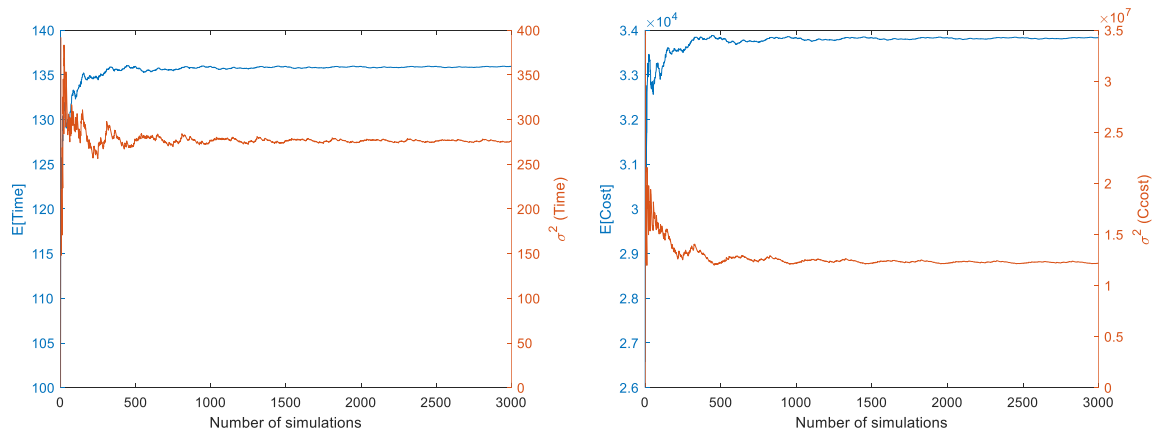


Figure 37. Mean and variance over the number of simulations for time (right) and cost (left) distributions

As a result of the simulation, the team gets distributions of time and cost. Figure 38 (left side) shows the probability density functions (PDF) of simulated cost and schedule outcomes. Figure 38 (right side) shows the joint PDF resulting from paired cost and schedule outcomes.

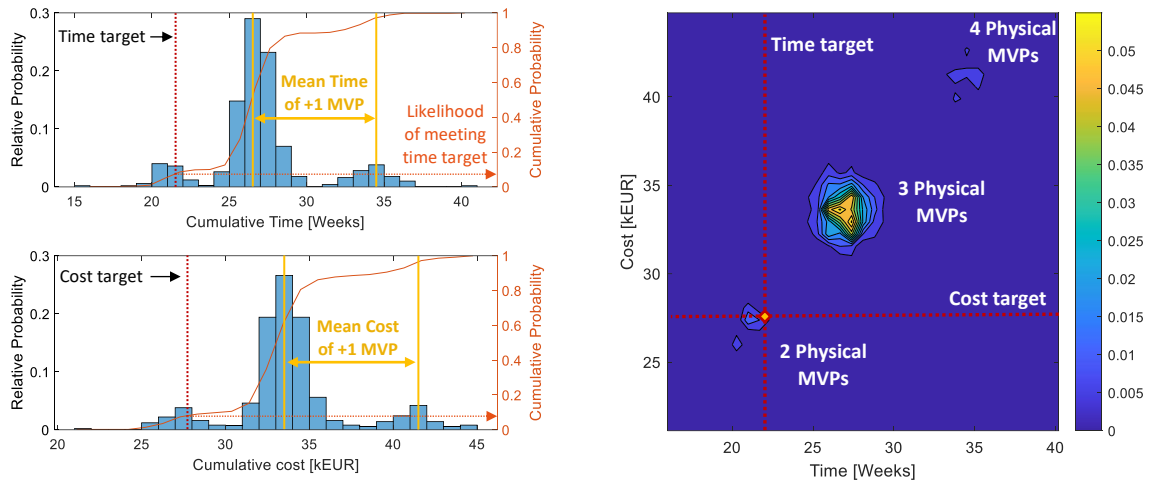


Figure 38. Illustrative case - Simulation output Time and Cost Probability Distribution Functions (PDFs) (on the left) Joint Time and Cost PDF contour plot of the simulated scenarios (on the right)

Such graphs are particularly useful to quickly compare process cost and time against budget and schedule targets. During the contract negotiation phase, it can be used to tailor the project schedule and budget and minimize programmatic risks. For instance, the delivery can be set in twenty-two weeks, with a budget of 28.5kEUR. From the graph, it can be easily realized that it is a tight schedule (likelihood of meeting the time target 8.6%) and a small budget (likelihood of meeting the cost target of 8.4%).

Another critical piece of information retrievable from the graph is the time and cost of iterations on physical MVPs. In Figure 38, the distributions exhibit three distinct areas corresponding to three different MVPs strategies. The first peak in the distributions corresponds to two physical MVPs, the second peak corresponds to three MPVs, and the third one to four. Each of the three macro-scenarios (i.e., MVP strategy) corresponds to different verification activities performed on a given physical MVP. Relying on these data, engineering teams can define a development strategy to reach the desired product maturity that meets time on time and budget constraints and assessing its feasibility.

3.4.3 Planning

Let us consider the notional case of delivering the system within twenty-two weeks. The objective is then to meet the schedule constrain, minimizing the cost. Providing the time target, CURSIVE identifies a baseline scenario minimizing the root mean square error between targets and available scenario data (Figure 39). The subsequent Sprint planning and MVPs structuring is then constrained to these targets.

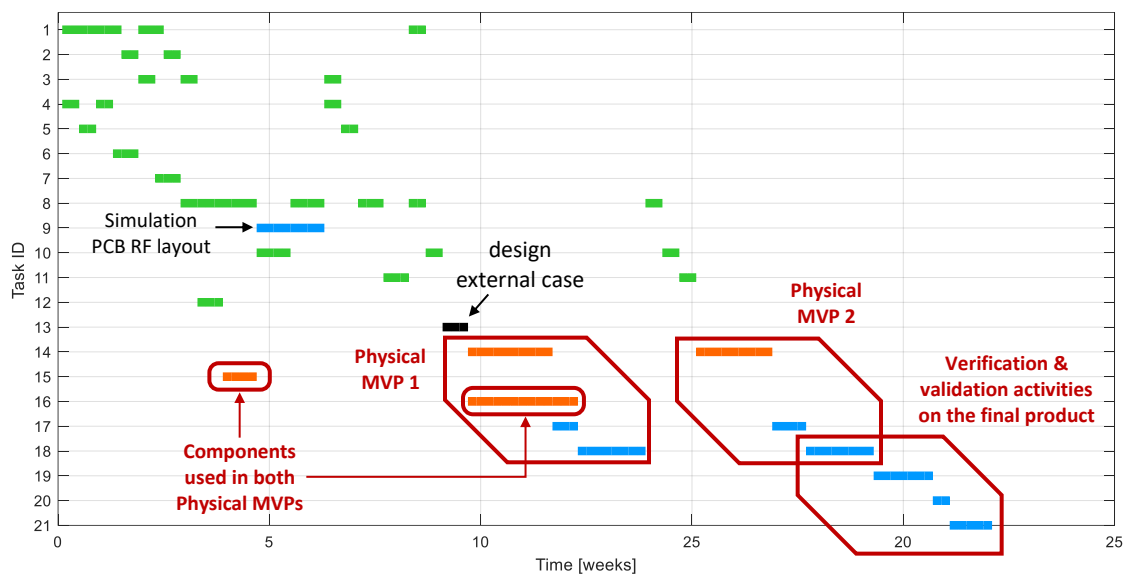


Figure 39. Illustrative case - Gantt chart of the baseline scenario meeting budget and time constraints design activities (marked in green and black), procurement or manufacturing (marked in orange), and AIT activities (marked in blue).

The heuristic approach deployed to identify the optimal number of MVPs suggests structuring the development process in seven Sprints. Combining this information with the product backlog architecture, the framework outlines the Sprints backlog clustering the user stories. Figure 40 shows the suggested Sprints planning.

The first Sprint lasts for two weeks, and it is devoted to design the baseband signal processing (FPGA), the digital-to-analog converter, the baseband filters, the phase-locked loop oscillator, the mixer, the X-Band Filter. The second Sprint lasts for two weeks, and it is dedicated

to designing the amplifier stages and the PCB layouts, refining the functional blocks defined in the first Sprint, and defining the In/Out interfaces. The third Sprint lasts for three weeks and a half. It is mainly dedicated to the simulation of PCB RF layout and refining the system's design based on the simulation results or potential issues encountered during the layouts' definition. In this sprint, the team also initiate the procurement of connectors. The fourth Sprint lasts for four weeks and mainly relates to finalizing the PBC layout (RF, digital, and power), designing the external case, and manufacturing the PCB.

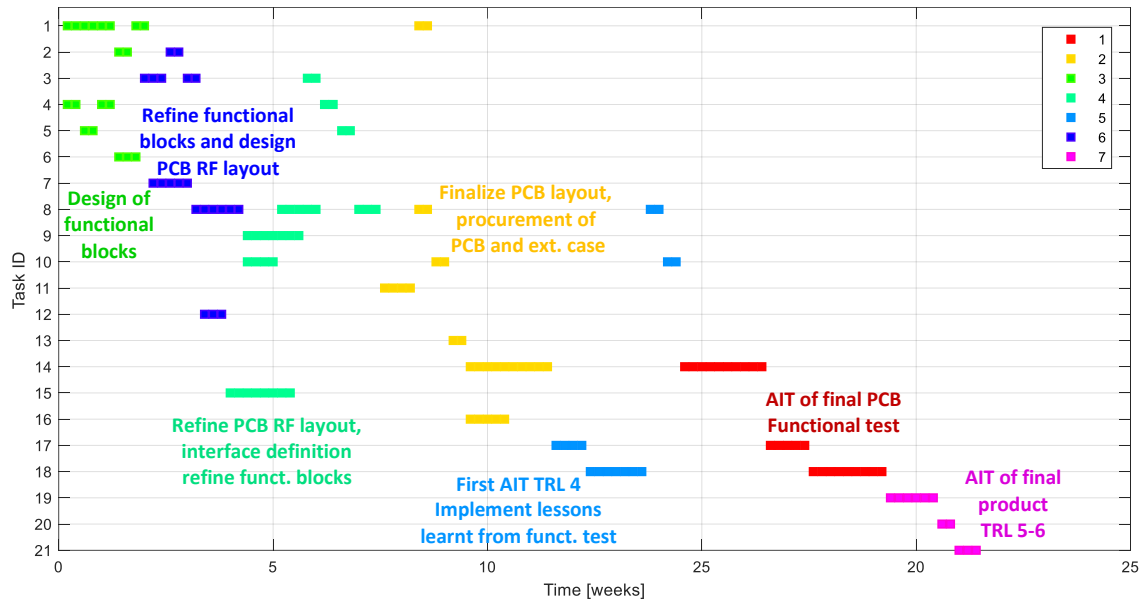


Figure 40. Illustrative case - Sprints Backlog

Sprint five lasts for three weeks and relates to PCB assembling and preliminary functional tests. Those preliminary functional tests provide feedback to the PCB layout definition that is then refined and consolidated within the same Sprint. The Sprint six lasts for four weeks, and it is devoted to final PCB version manufacturing assembling, and testing. The last Sprint lasts for two weeks, and it is entirely devoted to assembly integration and testing of the final product.

The outcome of the Sprint definition process offers information on the set of tasks to be performed within each Sprint (Figure 40), the Sprints durations, and the related cost Figure 41.

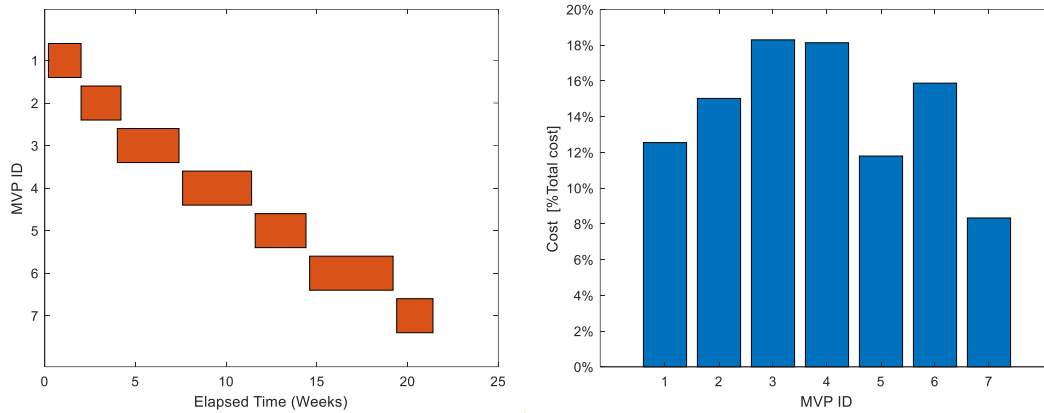


Figure 41. Illustrative case - Sprint sequence and costs

This information can be traced back to the MVP taxonomy provided in Table 5, ensuring a consistent product improvement over the Sprints sequence and a balance between Sprint length and TRL achieved. Table 6 summarizes the map of the MVP for the current case study related to the MVP taxonomy.

Table 6: Map of MVPs related to the MVP taxonomy

Sprint		MVP			Acceptance criteria: objective on MVP	
ID	Length	Fidelity	Artifacts	Repr. mode	V&V	Activities
1	2	Medium	Schematics	Digital	Verification	Analysis
2	3	High	PCB layout	Digital	Verification	Analysis
3	3	High	PCB layout	Digital	Verification	Simulation
4	3	High	Product Subset	Physical	Verification	Inspection
5	2	High	Product Subset	Physical	Verification	Functional Test
6	4	High	Product Subset	Physical	Verification	Functional Test
7	2	High	Product	Physical	Verification	Performance Test

The framework also provides additional insights about the process, unfolding the time and cost of each required to perform the different activities as well as the time and cost of iterations (showed as different staked color bars) for the given scenario (Figure 42). The additional information can help the development team reasoning on the cost-benefit of the selected implementation strategy and evaluate the product maturity improvement associated with the MVP delivery sequence.

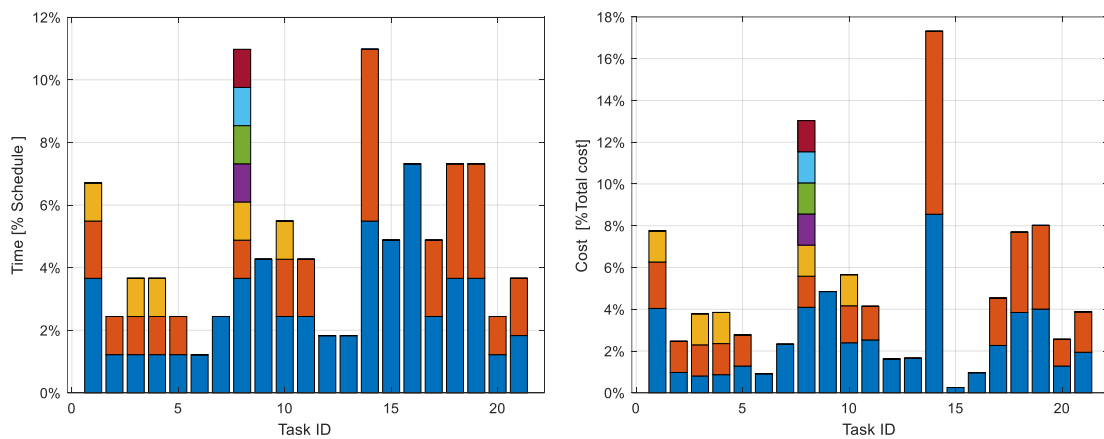


Figure 42. Illustrative case - Tasks Time and Cost Breakdown

As demonstrated in this simple case study, CURSIVE offers a set of methods and tools to support engineering teams in structuring and planning a development project. It helps project managers in defining the most viable development approach. It supports teams in clearly understanding the effect each task and each iteration cycle has on the cost and the schedule of the entire project; thus, it helps the team navigate programmatic and technical tradeoffs. All this information can support the fine-tuning of the process structure to increase process robustness, minimizing its sensitivity to potential design changes occurring during project execution.

3.4.4 Comparison with traditional project management approaches

This section considers the illustrative case presented above and applies traditional project management approaches, specifically the Program Evaluation and Review Technique (PERT) and the Critical Path Method (CPM). The comparison will provide a better understating of the limitation of traditional approaches, thus highlighting the benefits and the novelty of the framework proposed in this thesis.

As mentioned in the literature review, one of the main shortcomings of those traditional approaches is that they rarely model or analyze iterations. Furthermore, the PERT/CPM (Wiest & Levy, 1977) accounts only for the precedence constraints (Ben Issa & Tu, 2020).

In order to apply the PERT/CPM to the illustrative case study, the first step is to reshape the data in the required format. The tasks dependencies, i.e., tasks predecessors, are identified from the DSM in Figure 34. The expected time (t_e) required to complete an activity is evaluated according to eq (9), with a standard deviation evaluated according to eq (10) (Habibi et al., 2018; Wiest & Levy, 1977)

$$t_e = \frac{T_{LB} + 4T_{ML} + T_{UP}}{6} \quad (9)$$

$$\sigma_{t_e} = \frac{T_{UP} - T_{LB}}{6} \quad (10)$$

The expected time (T_e) required to complete a path in the project (i.e., a sequence of activities), according to classical PERT, is equal to the sum of all the t_e in the path, as reported in eq (11), with a standard deviation evaluated according to eq (12).

$$T_e = \sum_{i=1}^n t_{e_i} \quad (11)$$

$$\sigma_{T_e} = \sum_{i=1}^n \sigma_{t_{e_i}}^2 \quad (12)$$

The PERT data so evaluated are summarized in Table 7. Early start and finish, as well as late start and finish, have been estimated based on the predecessors derived from the DSM.

Table 7 PERT data (durations are presented in days)

Task ID	Predecessor	Early Start	Duration	Early Finish	Late Start	Slack	Late Finish
1		0,0	5,8	5,8	0,0	0,0	5,8
2	1	5,8	1,3	7,2	5,8	0,0	7,2
3	2	7,2	1,3	8,5	7,2	0,0	8,5
4	3	8,5	1,7	10,2	8,5	0,0	10,2
5	4	10,2	1,2	11,3	10,2	0,0	11,3
6	5	11,3	1,3	12,7	13,2	1,8	14,5
7	5	11,3	3,2	14,5	11,3	0,0	14,5
8	3 & 4 & 5 & 6 & 7	14,5	5,8	20,3	14,5	0,0	20,3
9	8	20,3	7,2	27,5	21,5	1,2	28,7
10	1 & 2 & 4 & 5 & 7 & 8	20,3	3,0	23,3	20,3	0,0	23,3
11	1 & 2 & 8 & 10	23,3	3,2	26,5	23,3	0,0	26,5
12	1 & 7	14,5	3,0	17,5	25,7	11,2	28,7
13	8 & 10 & 11 & 12	26,5	2,2	28,7	26,5	0,0	28,7
14	8 & 9 & 10 & 11 & 12 & 13	28,7	9,2	37,8	28,7	0,0	37,8
15	12	17,5	5,5	23,0	32,3	14,8	37,8
16	13	28,7	9,7	38,3	43,0	14,3	52,7
17	14 & 15	37,8	3,2	41,0	37,8	0,0	41,0
18	17	41,0	5,8	46,8	41,0	0,0	46,8
19	17 & 18	46,8	5,8	52,7	46,8	0,0	52,7
20	16 & 17 & 18 & 19	52,7	1,8	54,5	52,7	0,0	54,5
21	20	54,5	3,2	57,7	54,5	0,0	57,7

Starting from those data, the next step is to create an activity network and identify the critical path (CP), defined as the longest sequence of activities (in terms of time) that must be completed to conclude a project successfully, from start to finish. The activity network and identify the critical path are presented in Figure 43. In this case study the critical path is represented by the sequence of activities CP= [1, 2, 3, 4, 5, 7, 8,10, 11, 13, 14, 17, 18, 19, 20, 21]. The time associated with the critical path is 11.5 weeks with a standard deviation $\sigma_{CP} = 1.3$ weeks. It is easy to observe that the result is significantly lower compared to the estimates provided by the framework prosed

in this thesis. Mainly because the PERT/CPM does not account for any kind of iterations and typically does consider resource allocation and leveling. To overcome those issues, in practice, project managers tend to consider the worst scenario estimates or add a margin to the PERT estimates (around 30%) (Ben Issa & Tu, 2020). In such a situation, the estimated time associated with the critical path will be 18.6 and weeks 15 weeks, respectively. These values are closer to the estimation of the framework proposed here in the case of a single MVP.

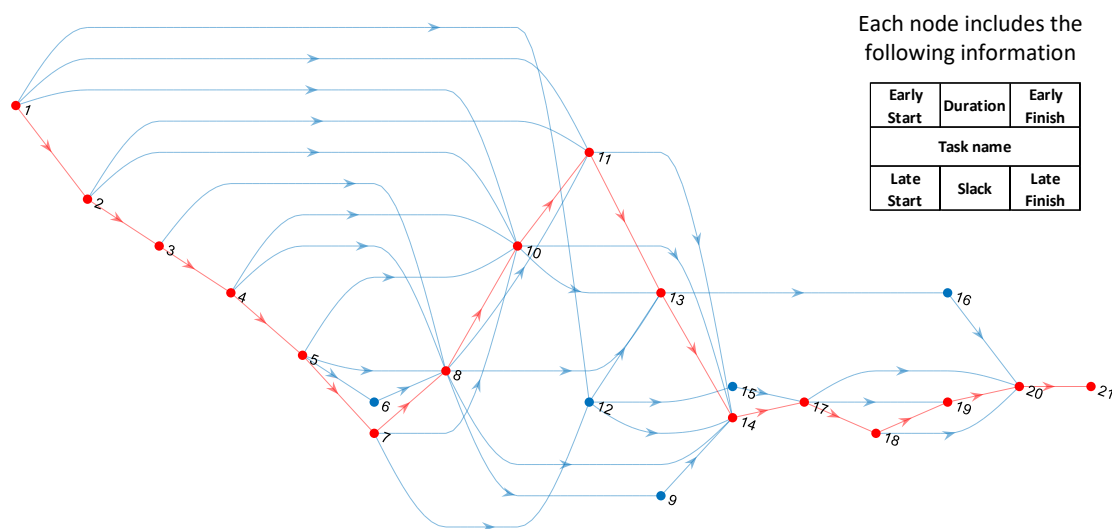


Figure 43 PERT (activities on nodes), CPM in red

Nevertheless, as just demonstrated, the PERT/CPM offers no means to evaluate the effect of multiple MVPs or potential iterations occurring due to new input coming from tasks performed during subsequent stages in the design and development of engineering systems.

To overcome the limitation of the approaches presented above, the project management community started considering Resource-Constrained Project Scheduling techniques (RCPS). Such methodologies account for precedence and resource constraints. Many approaches have been developed, including heuristic methods, meta-heuristic methods, exact methods, multi-criteria heuristic methods. The reader can refer to Ben Issa & Tu (2020) for an extensive literature review

on the topic. Those methodologies overcome the limitation of PERT/CPM; however, they still do not explicitly consider iterations, thus being agnostic on the number of MVPs.

The following time distribution is obtained by applying a heuristic method to the activity network presented in Figure 43 and resource constraints $RC = [2 \ 1 \ 1 \ 2]$ (i.e., electronic engineers, mechanical engineers, procurement specialists, and systems engineers, respectively).

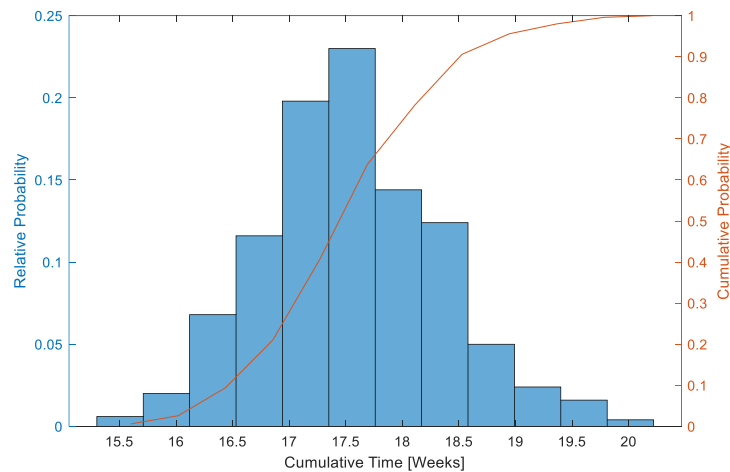


Figure 44. Time distribution for RCPS

Comparing the estimates provided by the RCPS with the one provided by the framework proposed in this thesis, it is observable that RCPS estimates are included in the results provided by CURSIVE. RCPS results correspond to the time distribution that considers one MVP (Figure 38). That is correct because, excluding iterations and multiple MVPs releases, the limiting case would be a project scheduling problem under precedence and resource constraints (that is correctly included in CURSIVE estimates).

These comparisons have proved that the methodology proposed in this work extends current theories and practices, including the ability to simulate and analyze iterations and multiple physical MVPs releases, essential features of Agile PD.

3.5 Validity of the decision support system

Within this thesis, considerable efforts have been dedicated to validating the decision support system (DSS) and specifically to the validation of the *process model*, the *data model*, the *graphical user interface*, as well as to the DSS *general* and *face validity*. However, the quality of support perceived by managers and engineer teams has not been assessed. Such an analysis would be somehow subjective, and results would be hardly generalizable. Furthermore, it is outside the scope of this work.

To evaluate the validity of the decision support system, it has been adopted the validation framework for DSS proposed by Finlay & Wilson (1997). The original validation framework has also been enriched integrating methods and criteria formulated by Adelman (1991); Boukhayma & Elmanouar (2016), Isaksson et al. (2020), Le Dain et al. (2013) as well as with the process model validation criteria proposed by Law, (2014), Smith & Morrow (1999). Figure 45 summarizes the validation framework we adopted.

3.5.1 *Process model validity*

The first type of validity tackled is the process model validity. In the original validation framework (Finlay & Wilson, 1997), it is called generically logic model validity, while in this work, it is specifically addressed as a process model. This type of validity consists of *analytical* and *theoretical* validity.

The *theoretical* validity relates to the adherence of the model construct to the theories underpinning the model itself. In other words, it aims to verify if the assumptions, the simplifications, and the process elements model are theoretically sound. CURSIVE satisfies the theoretical validity because it uses assumptions and modeling parameters based on the existing

literature. Specifically, as presented in section 3.2, the process is characterized as a network of activities exchanging information and deliverables (Smith & Morrow, 1999; Wynn & Clarkson, 2018), and the network topology is defined by a DSM (Eppinger & Browning, 2018). Task states are consistent with the Agile theory: *to do, in progress, done* (Rodríguez et al., 2018). Iteration models are aligned with the literature models presented in section 2.2.2. Both iteration and working policies are transparent and reported in *Algorithm 1*, page 68.

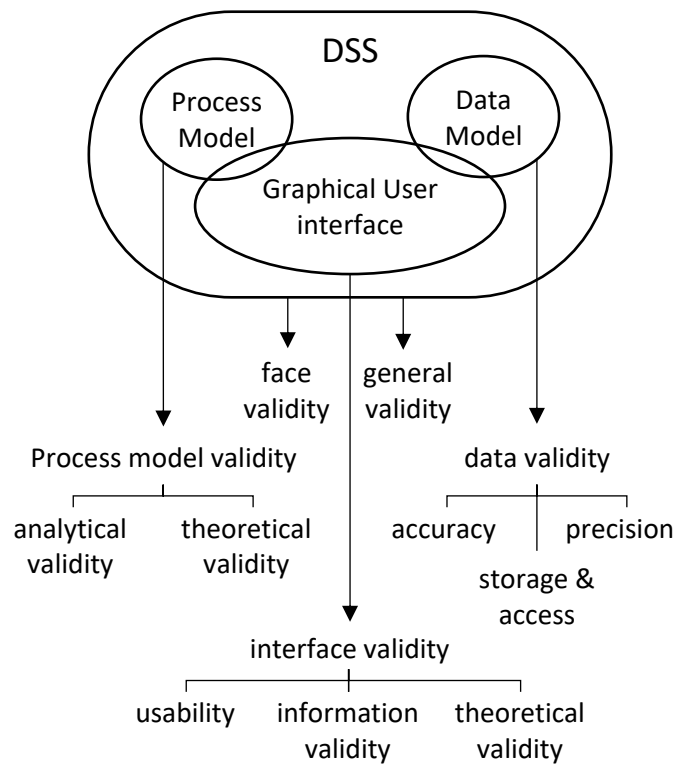


Figure 45. Framework for evaluating DSS validity

The *analytical* validity relates to the consistency between the model outputs and actual process performance. Typically, this validity is achieved by applying the framework to retrospective datasets gathered from the industry. In the literature, it is also defined this validity as the second level of validity (Smith & Morrow, 1999). The analytical validity of CURSIVE has been evaluated within the first case study, presented in chapter 4. Specifically, in section 4.4.2, the

forecast of our simulation model has been compared with the actual data, using as an input the process structure and project execution metadata provided by the manufacturer of the system subject of the case study. As reported in section 4.4.2, the model provided quite consistent outcomes compared with the actual data. By satisfying *analytical* and *theoretical* validity criteria, the process model can be considered validated for Finlay & Wilson (1997), Smith & Morrow (1999), Le Dain et al. (2013), and Isaksson et al. (2020).

3.5.2 Data model validity

Data model validity concerns both *accuracy* and *precision* of input/output data as well as data integrity aspects related to data storage and access.

In the context of DSS data model validity, *accuracy* refers to the set of parameters used in the model and not end-user defined, which can potentially systematically bias the output of the DSS (Finlay & Wilson, 1997). As concerns the simulation model, preset parameters have been avoided. The only parameter that can potentially bias the results is the learning rate adopted in the iteration model. The case studies presented in this thesis have been used a value equal to 15% (i.e., the repeated execution of a given task will take at every iteration 15% less time than the previous execution). A minimum boundary related to minimum process sensitivity has been set. This preset value can potentially systematically affect the forecast, overestimating or underestimating the time required for iterations. Since it is tightly correlated with the team executing the project, we recommend fine-tuning it according to team performance, relying on historical data derived from previous projects (if available). Concerning our case studies, that value represented a good approximation of the team learning rate.

As concern the Agile Viability Indexes formulation, the coefficients used in eq (3) and (5) have been derived from different industry surveys (Atzberger et al., 2020; Automotive Agile PEP,

2018; Digital.ai & VersionOne Inc., 2020; Schmidt et al., 2018b, 2019). The values of these coefficients are solidly grounded in the current state of the art and practice.

In the context of DSS data model validity, *precision* refers to the coherence of metrics and units of measurement between input-output data (Finlay & Wilson, 1997). In other words, it is necessary to ensure that metrics and units of measurement used in the simulation model and provided as output are coherent with the ones provided in the input data and vice versa. As concerns the cost metric, our DSS is units of measure agnostic, thus providing the output in the same unit of the input. As concerns the time metric, a check on the unit of measure of input data has been implemented, the time step of the simulation is set accordingly. Output data can be scaled according to end-user preference (output resolution is defined by input resolution).

The last aspect to be analyzed for validating the data model relates to data integrity, specifically data handling aspects such as storage and access. Input data are stored in a Microsoft® Excel® file (.xlsm extensions is used). Data access is managed by a MATLAB® script. The current version of the script is the seventh (*Input_v7.m*). The script reads and uploads DSM data, the task vector, the time and cost estimates vectors. It verifies the completeness of the data, reporting an error if any value is missing. Lastly, it also verifies the correct formulation of the time and cost estimates (in the triangular distribution, the three values a,b,c shall satisfy the inequality constraints $a \leq b \leq c$), reporting a warning in case of non-compliance. If all the checks are successfully passed, the system moves to the simulation. Simulation output data are stored in a MATLAB® data file (*simulation_output.mat*). Visualization script and planning scrip access and append their additional data to the simulation output file without the ability to alter the simulation data.

Based on the considerations mentioned above, the data model can be considered validated (Borenstein, 1998; Finlay & Wilson, 1997; Isaksson et al., 2020; Le Dain et al., 2013).

3.5.3 Graphical User Interface validity

Even if the current interface is to be considered as a minimum viable product of the DSS user interface, some efforts have been dedicated to ensuring its validity. Graphical user interface validity mainly concerns *usability* and *information validity*, and *theoretical validity* (Finlay & Wilson, 1997; Myers, 1995).

Usability is typically broken down into simplicity, consistency, and flexibility. Two points of view have been considered for evaluating usability: engineering teams' perspective and project managers' perspective. As an input interface, it has been used the process DSM tool (Figure 21) and a table for additional information (Figure 22). Based on the case studies, engineering teams generally seem more familiar with the DSM tool than project managers. However, it required less than ten minutes for project managers to master the DSM tool. As a simulation output interface, histograms and a contour plot (Figure 28) are provided. A traditional Gantt chart and a set of bar charts are used as the output of the planning phase (Figure 29). Based on the case studies, both engineering teams and project managers seem familiar with the tools adopted in the output interface and find them easy to use and interpret. As application software for the input interface, CURSIVE uses Microsoft® Excel®. It is widely adopted in many industries and holds a significant market share (Statista, 2021). As application software handling the output interface, CURSIVE uses MATLAB®; however, moving it into a web application might be considered in the future.

It is understood that two case studies do not provide enough data to reach statistical significance and generalize the claim on the interface usability; however, they are enough for a preliminary validation and to meet the objectives of this thesis (Smith & Morrow, 1999).

The *information validity* of the graphical user interface mainly relates to the consistent use of metrics and units of measurement between input-output data. This is primarily ensured by data

model validity. The user can define the units of measurement for time and cost in the input file. Then they will be consistently used over the whole model.

Theoretical validity of the graphical user interface mainly relates to the compliance of the selected data visualization strategy with best practice. As mentioned before, output data are presented by means of histograms, Gantt charts, and bar charts, being fully compliant with project management best practices.

Based on the considerations mentioned above, we can consider the graphical user interface validated according to Borenstein (1998), Finlay & Wilson (1997), Myers (1995).

3.5.4 DSS general validity

DSS General validity refers to the validity of DSS from a holistic perspective. It includes six main elements: robustness, internal validity, conceptual validity, experimental validity, operational validity, and reliability (Figure 46).

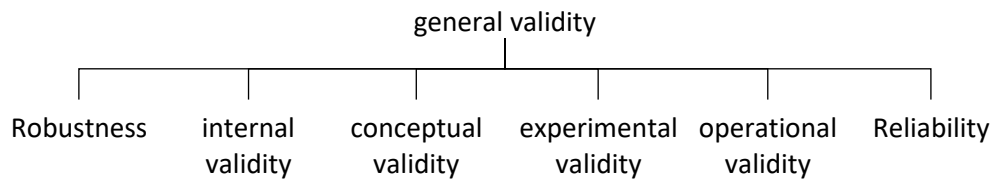


Figure 46. DSS general validity elements

Robustness, also called external validity (McCutcheon & Meredith, 1993; Salkind, 2012), transferability (Lincoln & G. Guba, 1958), analytical generalization or generalization validity (Yin, 2013), refers to the extent to which a method or a theory developed from one case is extendable to other situations with similar conditions (i.e., the range of applications the DSS can serve). The DSS developed in this thesis has been specifically designed and applied to physical and cyber-physical products. As presented in chapters 4 and 5, the framework can cover a wide range of applications.

CURSIVE has been applied to the development of a space system as well as the development of a consumer product. It has not been tested on “pure” software systems; thus, we cannot claim the applicability to such kind of products. Concerning the R&D composition and location, the DSS has been deployed and tested in cases of collocated (chapter 4) and sparse/dispersed R&D teams (chapter 5), providing in both situations valuable process insights (sections 4.5 and 5.5).

Internal validity refers to the extent to which causal relationships are certifiable in observing a phenomenon (Le Dain et al., 2013). According to (Yin, 2013) internal validity is mainly a concern for “explanatory case studies” (i.e., when a researcher is trying to explain how and why event x led to event y). According to Yin (2013), this logic is not applicable to exploratory or descriptive studies which do not concern this kind of causal situation (as in the case of our research). In this case, authors such as Smith & Morrow (1999), Le Dain et al. (2013), Yin (2013), and Isaksson et al. (2020) suggest adopting the *credibility* criterion, i.e., the extent to which the results appear to be adequate representations of the situation under study. In our framework is ensure by process mode validity (refer to section 3.5.1), including both theoretical validity (i.e., adherence of the model construct to the theories underpinning the model itself) and analytical validity (i.e., consistency between the model outputs and actual process performance ensured by verification on the case studies).

Conceptual validity refers to the suitability of the selected tools to describe/measure the phenomenon. Some authors relate to this kind of validity as construct validity (Finlay & Wilson, 1997; Isaksson et al., 2020; Le Dain et al., 2013; Yin, 2013). The literature acknowledges the difficulty of ensuring this validity the inevitable intrusion of the researcher’s biases. To overcome this issue, many authors recommend using multiple sources of evidence, constructing chains of evidence (by using model/tool reflecting reality), and have preliminary results reviewed by key informants (Isaksson et al., 2020; Le Dain et al., 2013; Seepersad et al., 2006; Yin, 2013). This

thesis followed these recommendations deploying the system for different case studies (Chapters 4 and 5). Data collection and analysis have been performed according to a standard format (as present in Table 9), and different interviewees in the different organizations participating in the case studies have reviewed the results. Having received positive feedback from case studies participants, the conceptual validity can be consider ensured. It is understood that two case studies do not provide enough data to reach statistical significance and generalize the claim on the conceptual validity; however, they are enough for a preliminary validation and to meet the objectives of this thesis (Smith & Morrow, 1999).

Operational validity relates to the extent to which the DSS can actually be used in an operational environment (Finlay & Wilson, 1997). Specifically, it concerns the ease of use, entry barriers (time required to learn how to use it), informativeness of the output, running time. This validity is mainly ensured by the usability, information validity, and theoretical validity of the graphical user interface (section 3.5.3). As concerns the running time, elapsed times are reported in Table 8. Simulations have been run on a workstation equipped with an Intel® Core™ i7-7700HQ CPU @ 2.80GHz and 64GB of installed ram. Based on the data reported in (section 3.5.3) and Table 8, the operational validity can be consider ensured.

Table 8. Running time

Elapsed time [s]		Activity
Case A (7000 sim.)	Case B (3000 sim.)	
1.7946	1.9825	Accessing the input file, reading and importing data, verifying data integrity
391.9040	614.5148	Simulation
10.2117	9.4135	Saving simulation data
0.0348	0.1123	Data visualization
3.8456	1.9317	Scenario analysis and planning

Experimental validity relates to the extent to which the research process is transparently and comprehensively exposed for external critical scrutiny (Finlay & Wilson, 1997; Le Dain et al., 2013). This thesis ensures experimental validity.

Reliability relates to the extent to which a study can be repeated with the same results given the same input conditions (Finlay & Wilson, 1997; Isaksson et al., 2020; Le Dain et al., 2013; Yin, 2013). This validity has been ensured by running the DSS several times using the same case studies input data and verifying the consistency of the results.

Based on the considerations mentioned above, the DSS can be considered generally valid according to Finlay & Wilson (1997), Isaksson et al. (2020), Le Dain et al. (2013), and Yin (2013).

3.5.5 *Face validity*

Face validity refers to the extent to which the models, the data, the assumptions, and the computational tractability seem reasonable to those who are familiar with the field of product development management (Smith & Morrow, 1999). It is a subjective metric, and it mainly refers to the relevance of the DSS for DSS test users.

CURSIVE satisfies *face validity* criteria according to Smith & Morrow (1999) since it addresses an important product development issue (as reported in chapter 1), has reasonable computational tractability (Algorithm 1 and Table 8), uses modeling parameters and assumptions (chapter 3, sections 3.1, 3.2 and 3.3) based on the current state of the art (survey in chapter 2) and state of the practice (chapter 2 section 2.5). Face validity has also been ensured by applying the DSS to the two case studies and collecting feedback from the interviewees.

Based on all the validity criteria mentioned above, the framework can be considered validated for Finlay & Wilson (1997), Isaksson et al. (2020), Le Dain et al. (2013), and Yin (2013).

3.6 Deployment in development projects

The design and development of products is a highly dynamic process, with a complex interplay of people and activities. Each project comes with a unique combination of people, skills, and tasks. Therefore, experiments in a controlled environment that sufficiently represent the problem would be extremely challenging to define. Nevertheless, simplified academic case studies would hardly reflect real project situations (Adelman, 1991).

Over the following chapters, the framework is applied to a set of real projects. The objective is to verify CURSIVE the capability in different industry contexts. The chapters go through the framework deployment steps over the projects and evaluate how the system supports engineering teams, thus meets research goals and research questions outlined in Sections 1.3 and 1.4.

Each project represents a different case study with different boundary conditions, team setting, organizational structure (Yin, 2013). The first project, presented in chapter 4, pertains to the development of a payload for a New Space Mission. It is a pilot study on a project already executed. This will allow us to benchmark the results provided by our model with the actual development metadata. The objective is to verify the results and validate the model forecasts.

The second project, presented in chapter 5, pertains to the development of a consumer product. This development project is still ongoing. In this case study, CURSIVE is used to support decision-makers in optimizing the combined development of the product platform and product accessories to minimize the time to market while keeping customers' hype.

Empirical data are collected and analyzed following a common approach and reported in the same format. The case studies format consists of the sections reported in the list on the following page, while data collection methods are reported in Table 9.

Case studies format:

1. General case study data
2. Organizational structure
3. Agile in the development process and motivation for Agile transition
4. CURSIVE deployment
 - a. Process structuring,
 - b. Simulation
 - c. Planning
5. Insights

Table 9. Data collection methods

Item of interest	Source of data	Notes
General case study	Structured interview ²	The interview's structure is reported in Figure 47 and Figure 48
Organizational structure	Documentation and archival records	They have been used to retrieve information on the organization of R&D divisions and typical supply chain management.
	Semi-structured interview ³	It has been used to get details on the development project structure
Agile in the development process and motivation for Agile transition	Semi-structured interview	
CURSIVE input data	Documentation and structured interview	Participants were asked to share process data and to fill the product backlog architecture

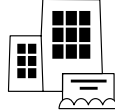
² Some sources refer to *structured interviews* also as *closed quantitative interviews* (Blessing & Chakrabarti, 2009)

³ Some sources refer to *semi-structured interview* also as *standardized open-ended interviews* (Blessing & Chakrabarti, 2009; Yin, 2013)

Company name



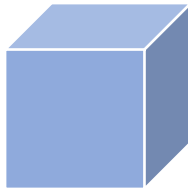
Location
country



Size
Startup, SME, Large



Employee
Total *n*
R&D *m*

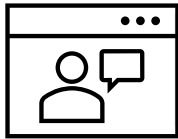


Product

Brief product/products description

Market

Addressed markets



Customers

Brief description of target customers, and go to market strategy

Product composition

Mechanics	%
Electronics	%
Embedded Software	%
Standalone Software	%
Discipline n	%
<hr/>	
Total	100%
Degree of physicality	[0, 1]

The product composition refers to the percentage of elements of the product belonging to a specific domain.

DoPh Degree of physicality

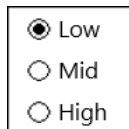
Hw Hardware

Sw_{Emb} Embedded Software

Sw Standalone Software

$$DoPh = \frac{\sum Hw + \frac{1}{2} Sw_{Emb}}{\sum Hw + \sum Sw}$$

Customer involvement



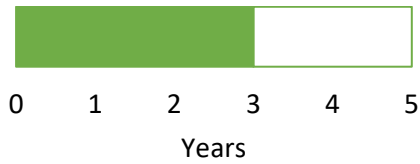
Customer only validating the product

Customer in mildly involved (monthly meetings)

Customer is heavily involved (weekly meetings)

Figure 47. General case study data collection format (1)

Experience with Agile



Sprint Length in weeks



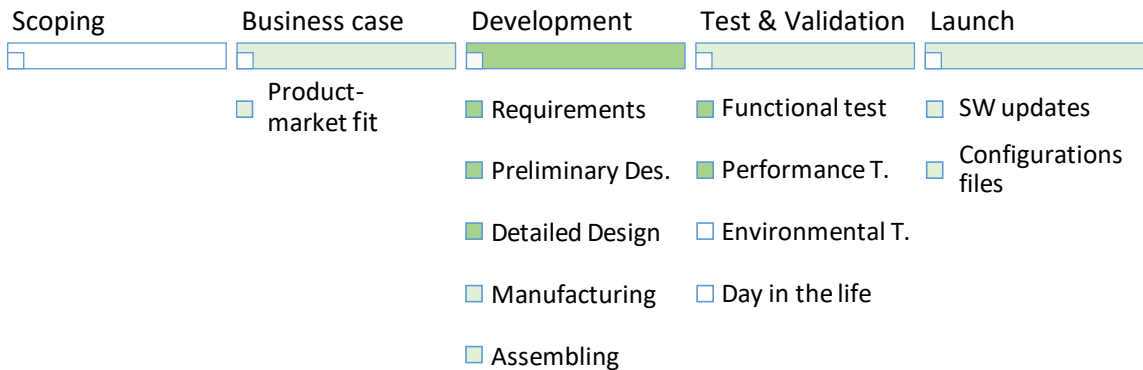
Impact on the organization



Notes on Scrum implementation if any

Scrum in the Development Process

- Agile fully implemented
- Agile used in some extent



Additional notes if any

Interviewees


	Name	<i>Interviewee name</i>
	Position	<i>position</i>
	Background	<i>degree, specialization</i>
	Role in Scrum	<i>Product Owner, Dev Team, etc.</i>

Figure 48. General case study data collection format (2)

This page intentionally left blank

Chapter 4

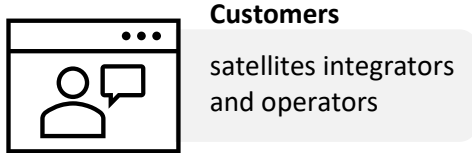
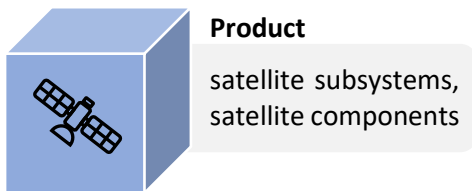
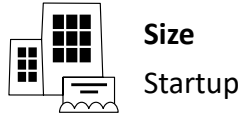
Cast study: New Space mission payload

In this chapter, the framework is applied to the development of an optical telecommunication payload for a New Space mission. The system under consideration is the same that has been the subject of the field research presented in Section 2.5. While during the field research, the analyses were limited to the design and development of the FlatSat model (ECSS, 2010, 2018), in this chapter, the development process of the entire system is covered, from the design phase to the assembling, integration, and testing (AIT) of the flight model (ECSS, 2010, 2018).

The context of the case study is a nanosatellite mission developed in a multiparty consortium (Camps et al., 2018). In the following section, the general case study data resulting from the structured interview are presented. Section 4.2 reports the organization and the project structure resulting from documentation and publicly available data analysis as well as the semi-structured interview with a project participant. Section 4.3 summarizes the motivation for Agile adoption and the fitting of Agile into development processes traditionally used by the organization, as described by the interviewee. Section 4.4 describes the application of the framework to the project data provided by the organization, detailing all the implementation steps. Lastly, process insights are derived, and the case study conclusion is drawn.

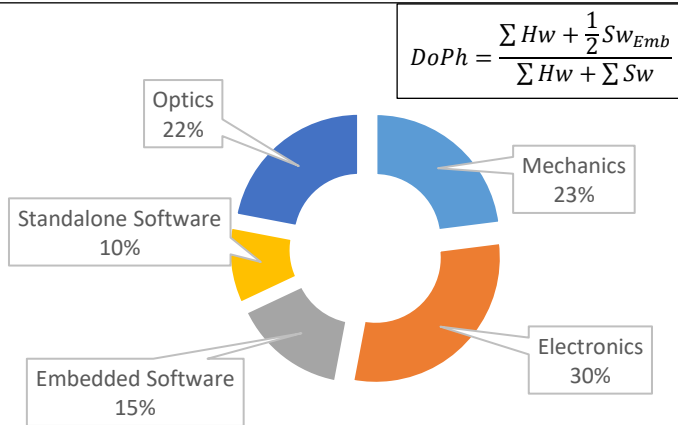
4.1 General case study data

Company A

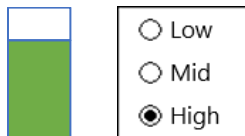


Product composition

Mechanics	23%
Electronics	30%
Embedded Software	15%
Standalone Software	10%
Optics	22%
Total	100%
Degree of physicality	0.83

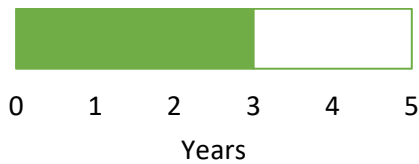


Customer involvement



- Customer only validating the product
- Customer in mildly involved (monthly meetings)
- Customer is heavily involved (weekly meetings)

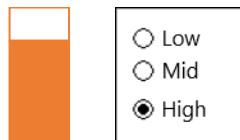
Experience with Agile



Sprint Length in weeks



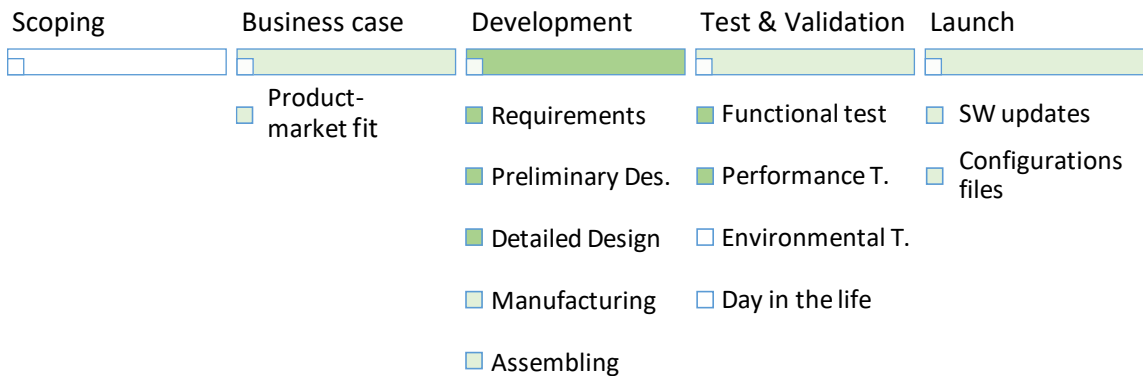
Impact on the organization



Scrum is implemented in most parts of the development. However, large obstacles exist due to the legacy paradigms of the market the company serves.

Scrum in the Development Process

- Agile fully implemented
- Agile used in some extent



Note: Some parts can be manufactured multiple times (at each iteration: electronics and some mechanical components). Some other parts can be manufactured only once (final version: expensive optics and mechanics)

Interviewees

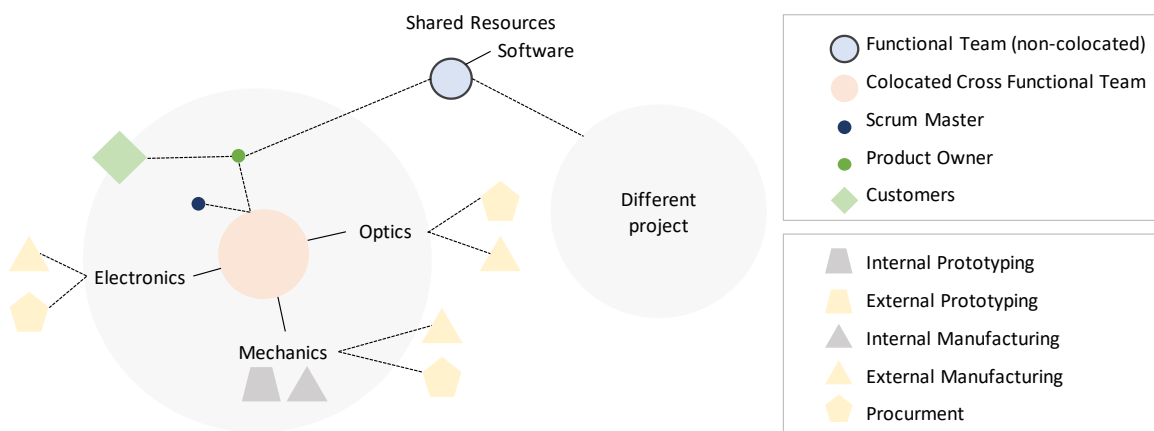
	Name	<i>Interviewee A1</i>
	Position	<i>Executive</i>
	Background	<i>Electronic engineer</i>
	Role in Scrum	<i>Mixed role, formally Product Owner, but also Dev Team</i>

4.2 Organizational structure

The R&D division of organization A has a cross-functional Scrum team consisting of mechanical, optoelectronics, and software engineers. Each team member is responsible for the design, procurement, assembly, integration, and test of the components of his/her area of knowledge. Software engineers are “shared resources” since they are employed in more than one project and relate to more than one team. Mechanical engineers are also responsible for the AIT activity of the full system (Figure 49).

The organization carries out the design of mechanics, optics, and electronics in-house. The manufacturing of all components is outsourced. Assembly, integration, and testing (functional and performance) are performed in-house. Environmental tests and day-in-the-life tests are entrusted to a third party for quality certification purposes. The software is fully developed in-house.

The main production facility is located in Estonia. Part of the supply chain is Estonian, but most suppliers are abroad, spread over Europe, the USA, and China. The customers are located in Europe and particularly in the Netherlands, Italy, and Portugal.



4.3 Agile in the development process, motivation for Agile adoption

Organization A uses a Stage-Gate model as the overall management tool for development activities and payments-deliveries negotiation with customers. The Stage-Gate process represents the status quo in the reference sector; therefore, it is used as the principal guiding framework even while attempting to implement Agile.

Agile-Scrum is fitted primarily in the stage called “Development” and in the stage called “Test and validation.” This is because to mark an MVP as done is required to prove its correct functionality and performance. Some efforts have also been made to push Scrum activities into the preceding “Business Case” stage (refer to section 4.1, page 107).

The team involved in the project includes three people covering all the disciplines required to develop the product: one optoelectronic engineer (responsible for optics and electronics), one mechanical engineer (responsible for structures, mechanisms, and AIT), one electronics/software engineer (responsible for electronics and software). Two out of the three members in the team had already used Agile, specifically the Scrum framework, in other projects. From the previous experience, they appreciated the team self-organization feature that enabled team members to micromanage their own work. They also appreciated the ability to enhance the product iteratively, allowing for early releases, thus a shorter time to market.

Therefore, given the small size of the team, the budget, and schedule constraints, the team decided to experiment with Agile, believing that, by adopting a traditional approach, they would not have delivered the product on time. The interviewee reported that the main drivers for Agile adoption were potential time and cost savings. The choice of experimenting with Agile was not based on rigorous quantitative analyses but instead relied on the team belief of purported time and cost benefits.

4.4 CURSIVE deployment

4.4.1 Process Structuring

As a first step, the team shall define the user stories for the project. Since the team was already familiar with Scrum and was using Jira (Atlassian, 2021) as a project tracking tool, the user stories the team defined on Jira at the beginning of the development were used. In this way, CURSIVE has been fed with the very same information the team had at the beginning of the project, avoiding biases caused by the new knowledge engineers have acquired while developing the product. Data were imported manually; in the future, an API to automate the importing process and enhance CURSIVE compatibility with existing project management tools might be developed.

As a second step, the development team has been asked to define the dependencies among the tasks using a DSM. Dependencies can be primarily infer based on the system architecture (mainly for design tasks and AIT tasks) and logical linkage between activities (e.g., a procurement task requires one or more design tasks as input, an AIT task requires one or more procurement/manufacturing tasks as input; AIT tasks provide feedback to design activities). This rationale minimizes the likelihood of potential biases due to the knowledge the team acquired during project implementation.

In the DSM, cells on the diagonal correspond to the tasks, the marks in off-diagonal cells indicate tasks interactions. The marks in the row denote activities inputs (sub-diagonal marks), while marks in the column indicate the feedbacks (super-diagonal marks). Feedback links can be deterministic, marked with the black filled circle symbol, i.e., probability of occurrence equal to 1, or probabilistic, defined with a probability $p \in [0,1]$.

Figure 50 shows the *product backlog architecture* the team produced. It provides a comprehensive overview of the *product backlog*, listing the eight-six *user stories* complemented

by the elicitation of their interactions. The backlog includes design activities (marked in green), rapid prototyping (yellow tasks), procurement or manufacturing (marked in orange), and AIT activities (marked in blue). Development activities involve three different fields of knowledge, namely mechanics, optics, and electronics.

Most of the activities exhibit deterministic dependencies. A few probabilistic interactions between AIT activities in the late stages of development and related design activities have been specified for some electronic components.

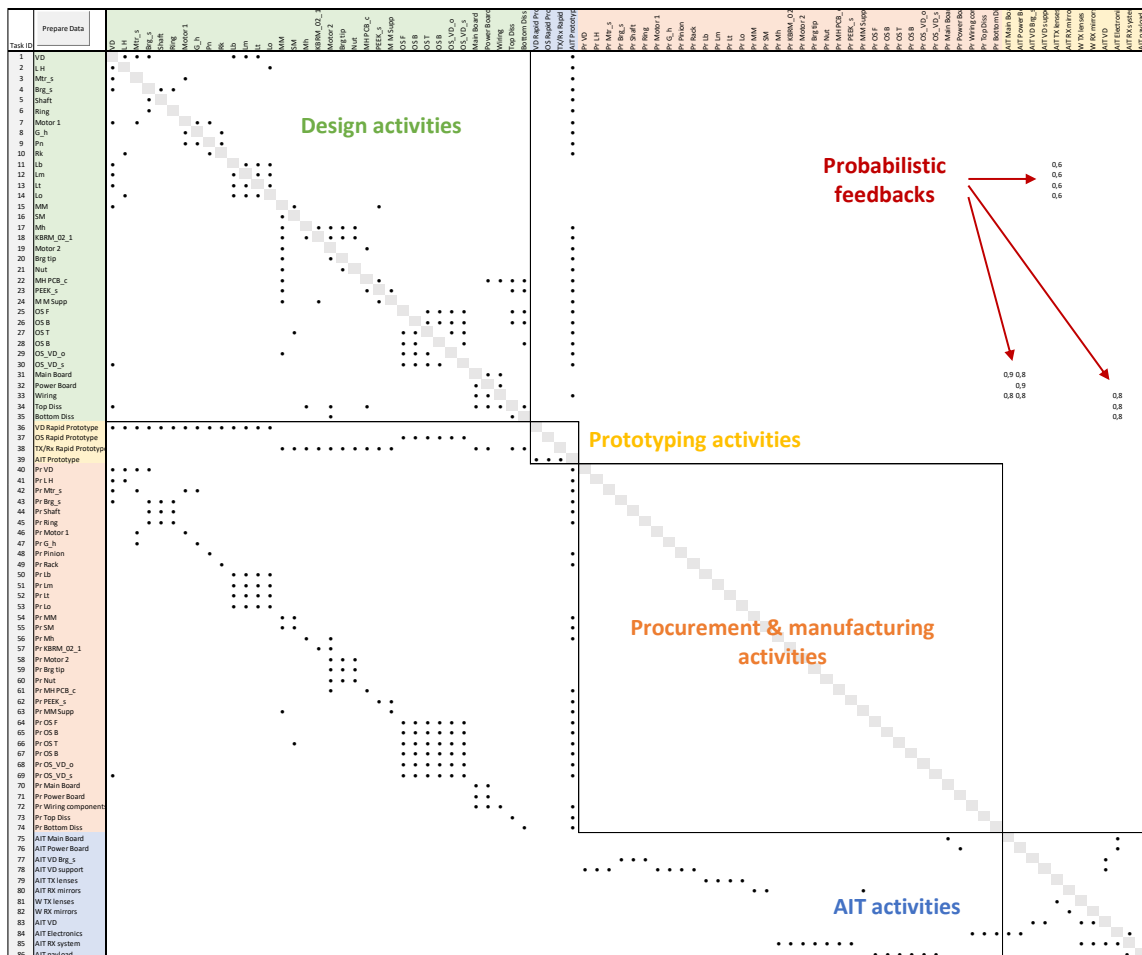


Figure 50. Input DSM for development of the optical telecommunication payload

Within the product backlog definition, project participants are also asked to provide additional data about the time, cost, and resources required to perform the activities (Figure 51). The information can be based on historical data, experts' opinions, design support tools output (Garzaniti et al., 2020), or their combination.

The process of defining the resources needed to execute the project does not involve pre-assigning tasks to people since it would violate the Agile principle of self-organizing teams. The team is asked only to indicate the discipline related to the activities. Those data become essential to refine the team composition and manage dependencies between internal resources (i.e., development team) and external resources (i.e., procurement and manufacturing partners).

Task ID	Prepare Data Task Name	WP ID	Expertise	# Exp needed	weeks			Currency		
					min	mode	max	min	mode	max
1	VD	1	M	1	0.6	1	2	960	1600	3200
2	LH	1	M	1	0.6	1	1.4	960	1600	2240
3	Mtr_s	1	M	1	0.4	0.6	0.8	640	960	1280
4	Brg_s	1	M	1	0.8	1	2	1280	1600	3200
5	Shaft	1	M	1	0.1	0.2	0.6	160	320	960
6	Ring	1	M	1	0.2	0.4	0.6	320	640	960
7	Motor 1	1	M	1	0.2	0.4	0.6	320	640	960
8	G_h	1	M	1	0.2	0.4	0.6	320	640	960
9	Pn	1	M	1	0.4	0.8	1	640	1280	1600
10	Rk	1	M	1	0.4	0.6	1	640	960	1600
11	Lb	1	O	1	0.2	0.4	0.6	320	640	960
12	Lm	1	O	1	0.2	0.4	0.6	320	640	960
13	Lt	1	O	1	0.2	0.4	0.6	320	640	960
14	Lo	1	O	1	0.2	0.4	0.6	320	640	960
15	MM	1	O	1	0.2	0.4	0.6	320	640	960
16	SM	1	O	1	0.2	0.4	0.6	320	640	960

31	Main Board	1	E	1	2	3.6	4	3200	5760	6400
32	Power Board	1	E	1	1	1.8	2	1600	2880	3200
33	Wiring	1	E	1	0.2	0.4	0.6	320	640	960

40	Pr VD	3	p	1	8	12	16	936	1170	1697
41	Pr LH	3	p	1	8	12	16	760	950	1378
42	Pr Mtr_s	3	p	1	8	12	16	696	870	1262
43	Pr Brg_s	3	p	1	8	12	16	800	1000	1450
44	Pr Shaft	3	p	1	3	4	8	2	2	5

Figure 51. Case study A - Complementary information provided within the product backlog architecture definition – work-packages traceability, expertise required (e.g., M: mechanical engineer, O: Optical engineer, E: electronic engineer, p: procurement department), cost and time estimates.

The definition of the product backlog architecture is a collective effort involving all players participating in the project. Each team member provides tasks, input/out dependencies, and time or cost estimates. The final product backlog architecture results from team consensus reached by adopting an iterative refinement process. We leveraged the value of the Agile Manifesto of promoting “Individuals and interactions over process and tools” (Beck et al., 2001) while providing a common language and structure to define the activities required to deliver the product. At the same time, we offer a holistic view of the project.

While shaping the product backlog architecture, the framework also provides the team with a handy chart to understand the viability of implementing Agile in the different portions of the project (Figure 52).

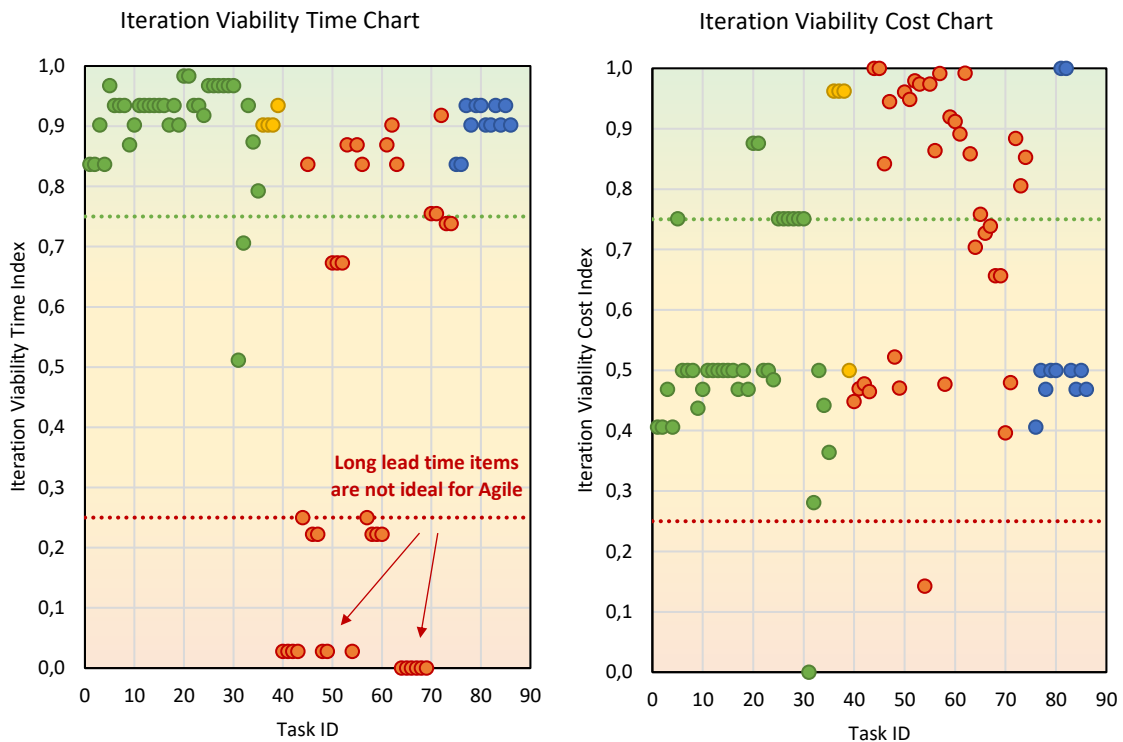


Figure 52. Case A – Agile Implementation viability

The same color code used in the product backlog architecture is adopted: design tasks are marked in green, procurement or manufacturing tasks are marked in red, prototyping tasks are in yellow, and AIT tasks are in blue. The tasks in the upper pane represent the activities for which Agile implementation is efficient both in terms of cost and time. In this case, fast iterations will lead to an equally rapid increase of product maturity without a significant impact on budget and schedule. Tasks in the lower pane represent the activities for which Agile implementation would be highly inefficient in terms of both cost and time. Iterating such tasks would lead to a schedule disruption or cost overrun. The tasks in the two yellow bands represent the activities for which Agile implementation would be inefficient either in terms of cost or time.

Based on this information, the team might decide to adopt different solutions.

1. To increase the level of granularity of task decomposition. This solution may improve the viability indexes of resulting tasks in terms of time (e.g., the activities resulting from activity 31 breakdown will be located higher than activity 31 itself). As a consequence, it also may improve the viability indexes of related procurement and manufacturing tasks.
2. To improve system modularity. This solution may improve the viability indexes of resulting tasks in terms of both time and costs. It also may improve the viability indexes of related procurement and manufacturing tasks.
3. To accept the current plan. Then to use a Hybrid-Agile process, implementing agile on the portion of the project where it will be efficient and Stage-gate where time and cost of iteration will be inefficient.

Such information already in the early phase represents a valuable aid for optimizing the whole process. Interviewee A1 collaborating on this case study, said: *“Having a better understanding of the quality of task decomposition as well as the system modularity would have been particularly useful during the project planning and execution. For instance, as shown in the chart, the high integration of the subsystem related to task 31 significantly impacted the cost. Unfortunately, we realized this at the end of the project. We plan to increase the modularity of that subsystem for the next version of the product”*.

The consolidated product backlog architecture resulting from this process, including the DSM and the complementary information, is then used as input for the simulation step of our framework.

4.4.2 Simulation

Seven thousand simulations (in batches, s , of 1000) have been run to stabilize mean and variance of time and cost distributions within precision, $\varepsilon = 10^{-4}$ (Figure 53) according to equation (6), (7).

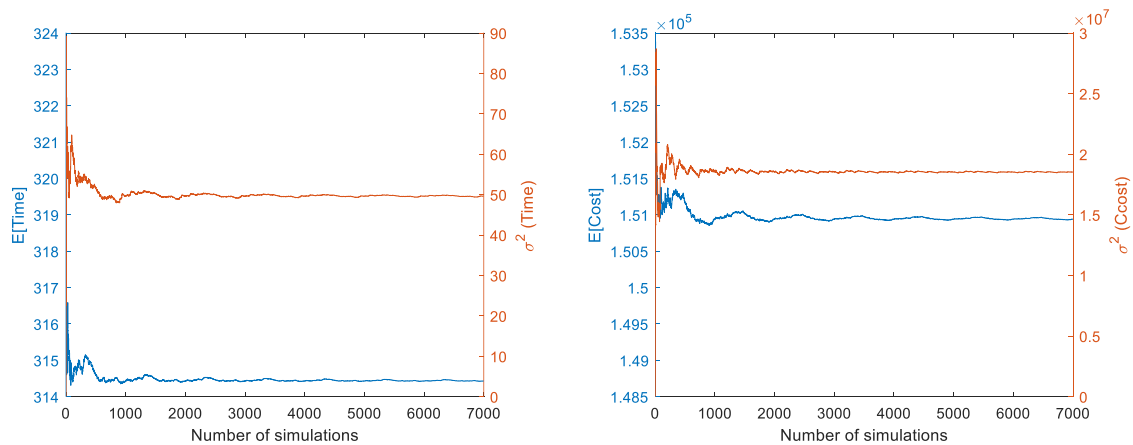


Figure 53. Mean and variance over the number of simulations for time (right) and cost (left) distributions

As a result of the simulation, we get distributions of time and cost. Figure 54 (left side) shows the probability density functions (PDF) of simulated cost and schedule outcomes. Figure 54 (right side) shows the joint PDF resulting from paired cost and schedule outcomes.

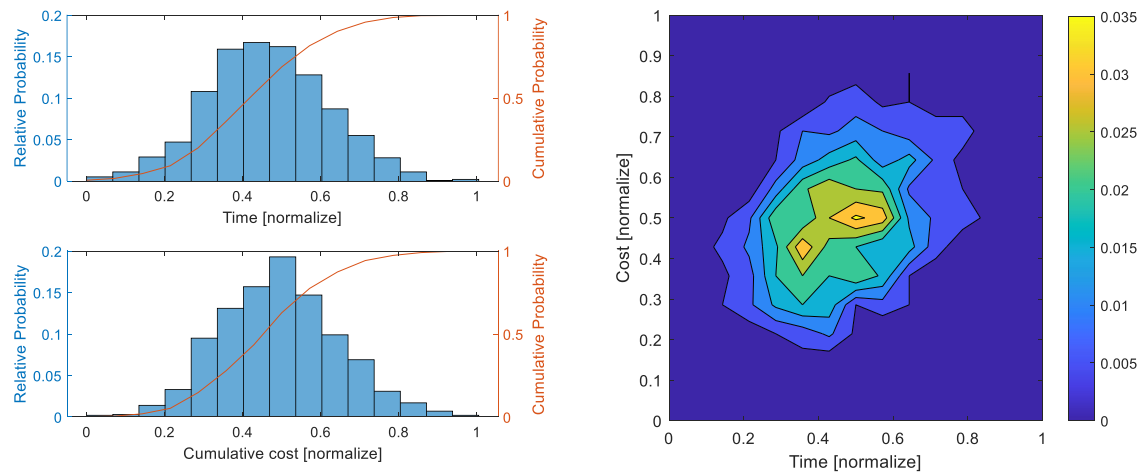


Figure 54. Case A - Simulation output Time and Cost Probability Distribution Functions (PDFs) normalized (on the left) Joint Time and Cost PDF contour plot of the simulated scenarios (on the right)

These graphs are particularly useful to quickly compare process cost and time against budget and schedule targets. (Figure 55). During the contract negotiation phase, the simulation model can be used to tailor project schedule and budget, mitigating programmatic risks and increasing the likelihood of successful product delivery within cost and time constraints. Alternatively, it can also be run while the process is ongoing to monitor process execution, as presented in section 3.3.1. Activity duration and cost estimates or tasks dependencies can be updated with actual values as soon as they become available, thus having more reliable results for the subsequent phases of the project. The simulation results evolve over the process execution to the extent the development team iterates and updates the product backlog architecture data.

Simulations become crucial when dealing with projects involving hundreds of tasks, different stakeholders, and complex interactions among activities and people. As the complexity

increases, simulations become the only way to assess the impact of a single design decision on the whole development process.

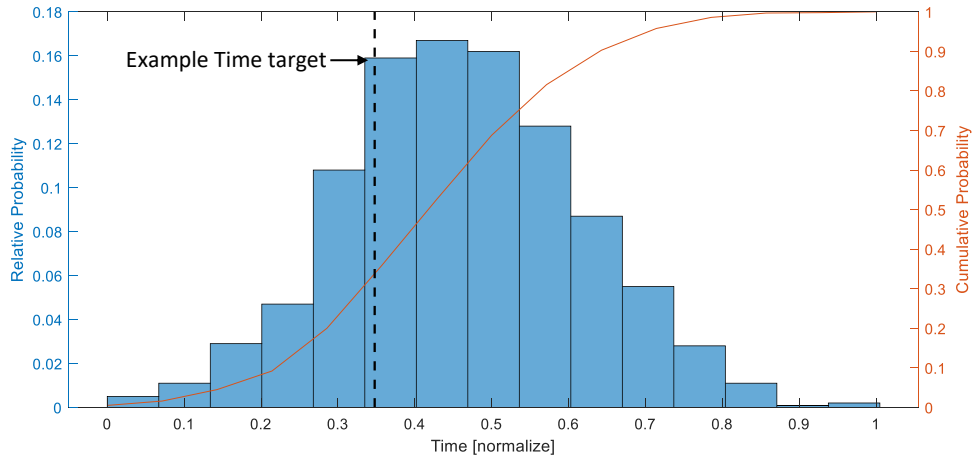


Figure 55. Schedule Target vs time distribution of simulated scenarios

The model proposed here also allows evaluating the sensitivities of budget and schedule to potential risk factors associated with activity costs, durations, and iterations. Thus, it enables decision-makers to refine the process architecture to enhance process performance. At the time of project implementation, the team did not optimize the process, which generally resulted in delays. Interestingly, given the input summarized in Figure 50 and Figure 51 of the non-optimized process, CURSIVE provided quite consistent outcomes compared with the actual data (Figure 56).

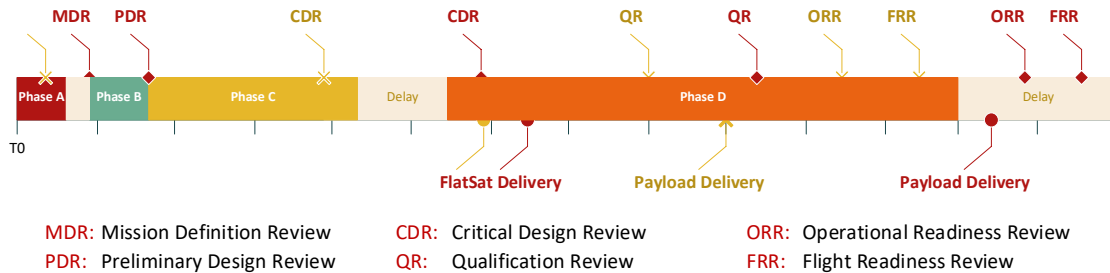


Figure 56. Case A- Overall Schedule. In yellow planned milestone, in red actual milestone.

4.4.3 Planning

During the contract negotiation phase, the consortium set the schedule and budget constraints for all the mission contributors. According to the agreement, the payload shall be delivered within a tight schedule and a small budget (0.20 T of time distributions and minimum cost in Figure 54).

Providing time and schedule targets, CURSIVE identifies a baseline scenario able to meet such targets minimizing the root mean square error between targets and available scenario data (Figure 57). Therefore, the Sprint planning and MVPs structuring is constrained to these targets.

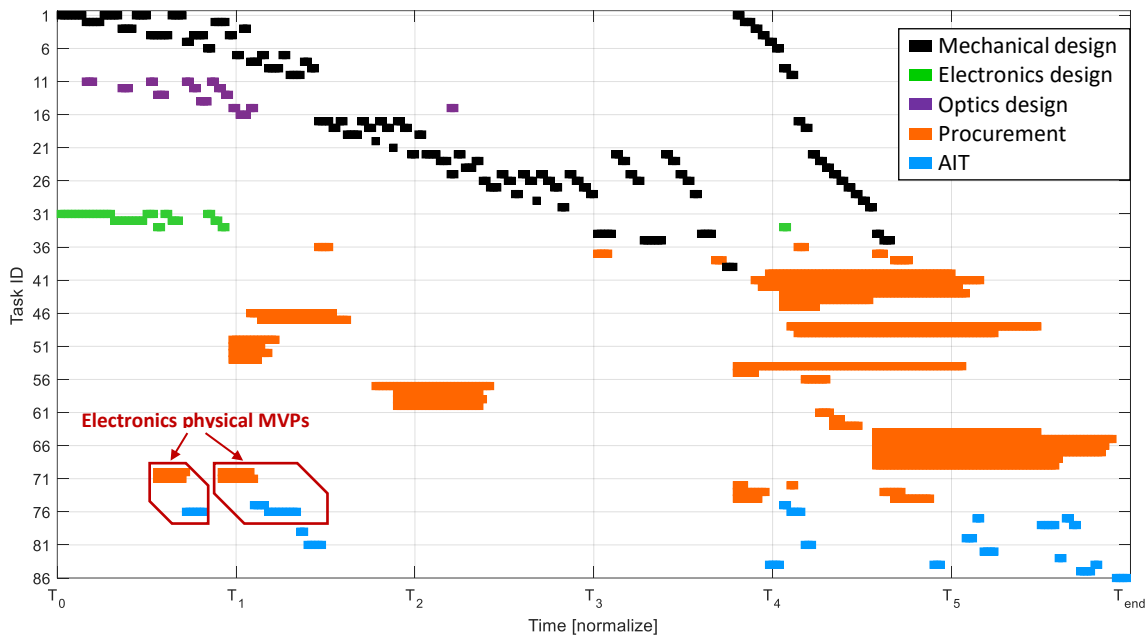


Figure 57. Gantt chart of the baseline scenario meeting budget and time constraints

The heuristic approach deployed to identify the optimal number of MVPs suggests structuring the development in seventeen Sprints. Combining this information with the product backlog architecture, the framework outlines the Sprints backlog clustering the different user stories. The result of the cluster analysis is summarized in Figure 58.

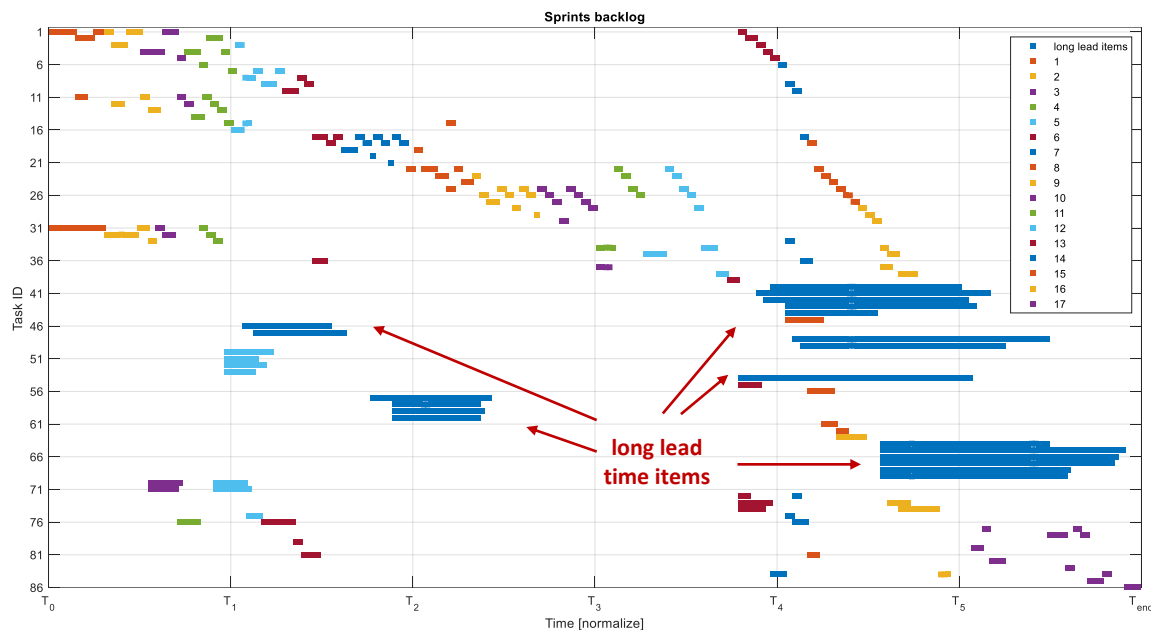


Figure 58. Case A - Sprint backlog

As we can observe in Figure 58, the system performed quite well in organizing the backlog for design, procurement of non-long lead time items, and AIT activities. As concern procurement and manufacturing of long lead time items (lengthy blue bars), which are cross-Sprint activities, they have not been included in the cluster analysis because they would have introduced noise, and the framework would not have been able to allocate them properly. In some cases, they would have been assigned based on the starting time, in some others according to the finishing time. The tasks to be excluded from the cluster analysis are systematically identified based on the Agile viability indexes presented in Figure 52. Excluding long-lead items from this analysis does not represent a critical issue for two reasons. First, the team can refer to the project Gantt chart as presented in Figure 58 or to the long lead items allocation chart (Figure 59, bottom) to initiate components procurement in the most appropriate Sprint according to the execution time. Second, by its nature, our framework does not aim to replace the people reasoning and decision-making activities, but it seeks to support and ease that.

The outcome of the Sprint definition process offers information on the set of tasks to be performed within each Sprint, as well as the Sprints' durations (Figure 59, left side) and the related cost (Figure 59, right side). Long lead items cost is allocated within the Sprint the procurement is initiated, while the delivery time is not included in the Sprints duration. Information about those items is summarized in a different graph indicating the Sprint where the procurement is initiated and the Sprint where the orders are delivered (Figure 59, bottom).

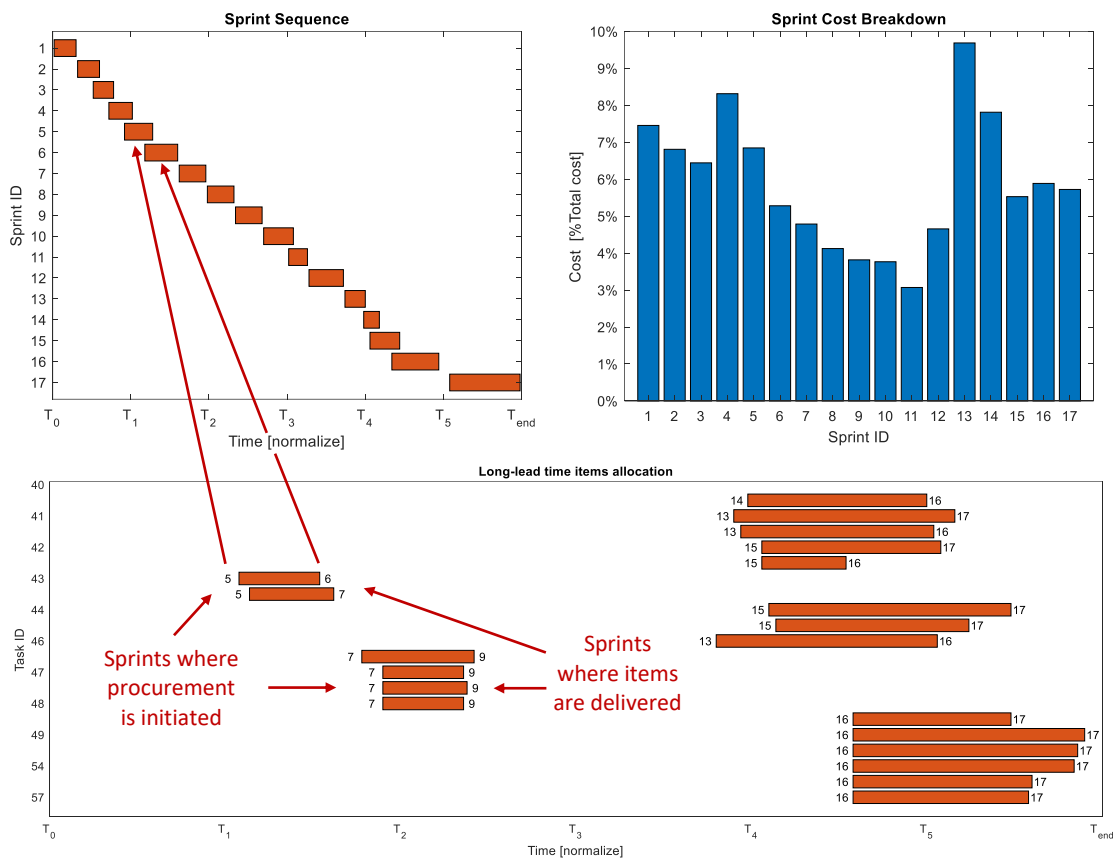


Figure 59. Sprint sequence and cost Breakdown

Most of the Sprints are in a finish-start sequence; however, some sprints have some degrees of concurrency. This is because the framework tries to optimize resource allocation, ensuring resource-leveling. So, if a Sprint is dedicated to developing an MVP that does not require a given discipline, CURSIVE suggests also starting another Sprint where the team can exploit that

120

discipline. Even if this might not be very aligned with the orthodoxy of Agile theory, a non-homogeneous environment requires additional attention to the problem of resource allocation and leveling. This strategy may help improve process performance.

4.5 Process Insights

The analysis of the process implementation strategy provides useful insights to evaluate the actual benefit of using the Agile/Hybrid-Agile PDP in terms of cost, time, and technical risk. Figure 60 and Figure 61 show the normalized cost and time required to perform the different activities as well as the time and cost of iterations (showed as different staked color bars) for the given scenario.

Figure 60 and Figure 61 that some user stories are time-critical while others are cost-critical. Therefore, deciding whether to iterate a given task based on the cost or time efficiency directly relates to the Δ risk-mitigation enabled by each iteration cycle.

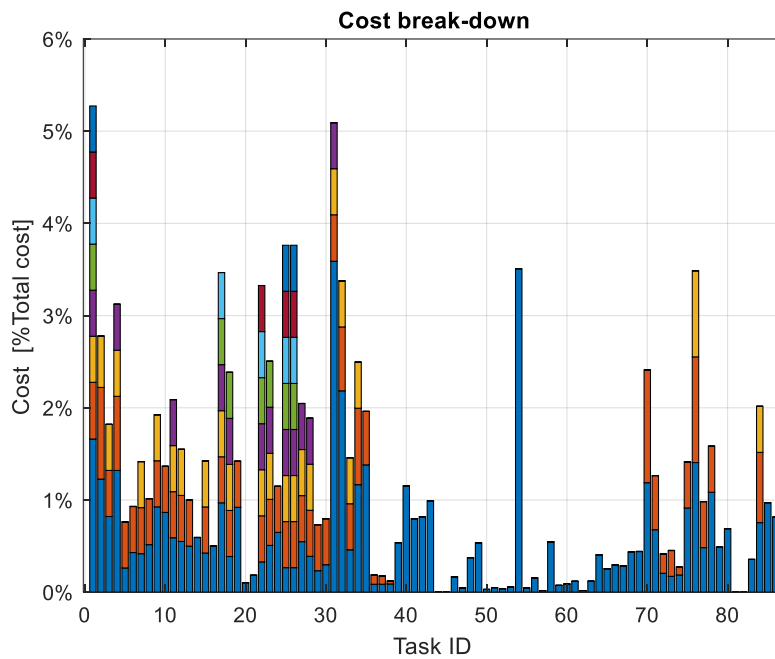


Figure 60. Case A - Tasks Cost Breakdown

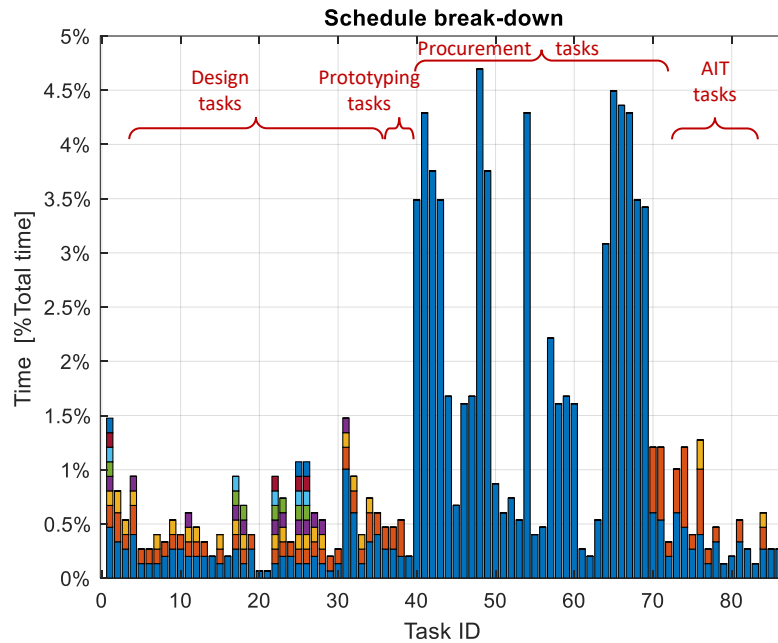


Figure 61. Case A - Tasks Time Breakdown

The team can use the information reported in, Figure 59, Figure 60, and Figure 61 to map the MVPs sequence to the taxonomy evaluating the product maturity improvement achieved over time, as shown in Table 10.

For instance, the team can decide to mitigate the technical risk deriving from a task, such as the 31st, at different stages of the lifecycle, accepting the programmatic risk posed by cost and time investment. If the task relates to a critical element of the system, they can fully exploit the Agile PDP capability of retiring technical risk by using DMTR iteration.

For other activities, the programmatic risk might jeopardize the whole project (e.g., procurement of long-lead items – task 64 to 69), and the team can decide to be conservative using DAR iterations (e.g., task 25 to 30 refers to the design of those long lead items) and adopting traditional systems engineering approaches.

Table 10. Map of MVPs related to the MVP taxonomy

Sprint	MVP			Acceptance criteria: objective on MVP	
ID	Fidelity	Artifacts	Repr. mode	V&V	Activities
1	Medium	Schematics/CAD	Digital	Verification	Analysis
2	Medium	PCB layout/CAD	Digital	Verification	Analysis
3	High	PCB Prot./CAD	Physical	Verification	Inspection
4	High	PCB Prot./CAD	Physical	Verification	Functional Test
5	High	Funct. Prototypes	Physical	Verification	Functional Test
6	High	Funct. Prototypes	Physical	Verification	Funct.Test/Inspection
7	High	Funct. Prototypes	Physical	Verification	Functional Test
8	High	CAD, BOM	Digital	Verification	Analysis
9	High	Product Subset	Physical	Verification	Funct.Test/Inspection
10	High	Funct. Prototypes	Physical	Verification	Funct.Test/Inspection
11	High	Funct. Prototypes	Physical	Verification	Funct.Test/Inspection
12	High	Funct. Prototypes	Physical	Verification	Funct.Test/Inspection
13	High	Product Subset	Physical	Verification	Functional Test
14	High	Product Subset	Physical	Verification	Funct. and Perf. Test
15	High	Product Subset	Physical	Verification	Funct. and Perf. Test
16	High	Product Subset	Physical	Verification	Funct. and Perf. Test
17	High	Product	Physical	V&V	Performance Test

Having a clear understanding of the effect each task and iteration cycle has on the cost and the schedule of the entire project helps the development team navigate programmatic and technical tradeoffs. The information offered by the proposed framework can also support teams in fine-tuning the process structure to increase process robustness, minimizing its sensitivity to potential design changes occurring during project execution.

4.6 Summary and interim conclusion

This chapter analyzed the development process of an optical telecommunication payload for a New Space mission applying our framework. The analysis covered the entire development of the system from the design phase to the assembling, integration, and testing (AIT) of the flight model (ECSS, 2010, 2018), including prototyping, procurement, and manufacturing activities.

Sections 4.1 and 4.2 introduced the case study providing background information on the project and the organization executing the project. Section 4.3 briefly discussed the motivation for adopting Agile. Section 4.4 dived into the PDP structuring and planning analysis using CURSIVE. The analysis of the PDP has been organized according to the three macroblocks of our framework, namely process *structuring*, *simulating* (or *simulation*), and *planning*.

In the *structuring* phase, a set of methods and tools for reasoning about the structure of the decision problem have been deployed. First, eighty-six tasks spanning over the whole development cycle have been identified. Then, those activities have been organized in a DSM, defining their interconnections (tasks dependencies) and outlining the order in which tasks shall be addressed. Tasks exhibited deterministic and probabilistic dependencies. The outcome of this exercise has been consolidated in the product backlog architecture. The product backlog architecture has been integrated with complementary information about the time, cost, and resources required to perform the activities.

Within the *structuring* phase, CURSIVE also assessed the viability of implementing Agile for the given project. The time viability index for the entire project, AV_{prj}^T , is equal to 0.7, while the cost viability index, AV_{prj}^C , is equal to 0.64. Those values imply that the project can benefit from Agile adoption; however, it is not recommended to execute all the portions of the project using

Agile. The detailed analysis of the Agile Viability charts (Figure 52) identified the items for which the use of Agile is not beneficial (i.e., long lead time items). Therefore, the framework suggested executing the project adopting a Hybrid-Agile approach. The information resulting from the structural reasoning exercise has been then used as input for the *simulation* phase.

During the *simulation* phase, CURSIVE has analyzed the solutions satisfying the problem constraints and evaluates the overall process performance for different variables combinations and values. As an outcome of the simulation, we got time and cost distributions representing the feasible scenarios and the dataset, including all the process details for each considered scenario.

The *planning* phase has been explored in-depth the scenario that meets the time and cost target offering an actionable plan. A heuristic approach has been deployed to define the Sprints backlog as well as the Sprints sequence. While defining the Sprints backlogs, CURSIVE has excluded the activities that are not recommended to be executed using Agile, as indicated by the viability indexes. Those activities have been then related to the different Sprints depending on procurement initiation and delivery time. The results of the planning activities have been summarized in a set of charts (Figure 57, Figure 58, and Figure 59), offering a comprehensive picture, from multiple perspectives, of the project execution plan.

Together with the structuring reasoning exercise, the Sprint backlog definition represented a fundamental activity in the deployment of Agile for hardware systems. The first provided the rationale for Agile implementation. The latter provided the rationale for the organization of Sprints (including goals, lengths, MVPs, and transition), as well as means of coordination between activities executed adopting different implementation strategies (iterative vs. not iterative).

The combined outcome of the three phases of our framework provided valuable insights on the process performance (Agile or Hybrid-Agile) in terms of cost, time, and technical risk.

Figure 60 and Figure 61 reported the normalized cost and time required to perform the different activities, as well as the time and cost of iterations. Table 10 mapped the Sprint sequence to the MVP taxonomy, evaluating the product maturity improvement achieved over time. These data brought the focus on critical activities, such as tasks 1 and 31 (cost-critical), or 48 and 64 (time-critical). This information can be used to evaluate potential strategies for process improvement, such as increase tasks granularity (e.g., task 1), increase the system modularity (e.g., task 31), or consider different suppliers (e.g., task 64 to 69).

As shown in the case study, CURSIVE does not aim to make independent decisions, replacing development team judgment. Instead, it is a predictive model aiming to provide the team with quantitative means to underpin their decision-making process. The following conclusions can be drawn from this study:

- If a Hybrid-Agile approach is adopted, means of coordination are needed not only with external partners (e.g., consortium participants) but also within the organization between Agile and non-Agile activities (Figure 59). This coordination of tasks dependencies becomes essential to ensure coherent development and avoid schedule disruption.
- Using Agile does not necessarily mean shorter development time or lower cost. Actually, due to the cost of iterations (Figure 60), Agile might exhibit a higher cost than traditional development processes. However, Agile lowers the technical debt (Allman, 2012), reducing the probability and the impact of reworks in later phases of the development.
- Vendors' selection, and generally supply chain management, represents a critical activity not only for quality reasons but also in terms of lead times. For instance, tasks from 64 to 69 could have been assigned to a different supplier able to deliver in less than a tenth of the time (without cost increases), allowing iterations also in that portion of the project.

- System modularity is a key feature for the effective implementation of Agile. It represents a crucial driver for both the cost and time of iterations; high modularity allows for more and faster iterations. In this case study, the system exhibited a mild degree of modularity: system modularity index (SMI), equal to 0.223 (Figure 62). The SMI index is based on the exponential decay approximating the actual decay structure of sorted singular values of the product DSM. Please refer to eq. (6) in (Holttä-Otto & de Weck, 2007) for further details.

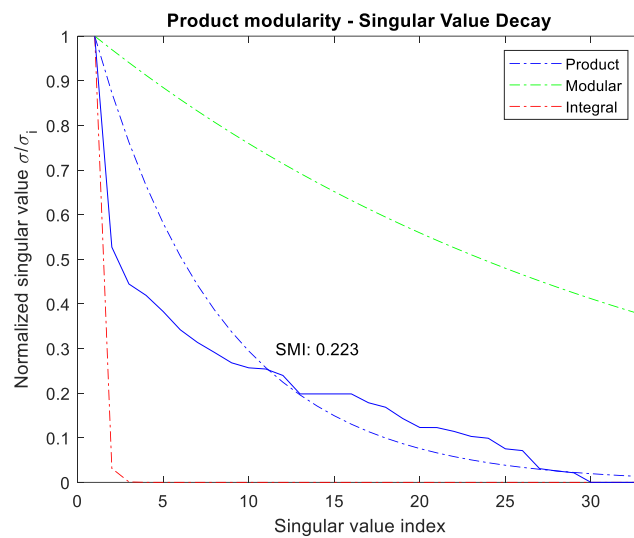


Figure 62. Degree of modularity

Given the degree of modularity, the Hybrid-Agile approach was a reasonable compromise. However, some subsystems, such as the one related to task 31 or 32, could have been further decomposed, enhancing the cost-effectiveness of Agile implementation.

- In the context of Agile for hardware, the product maturity does not increase homogeneously. Different subsystems achieve different maturity in different Sprints (as reported in Table 10), eventually converging over the project execution. Mapping and managing the MVP sequence over time allows for the coherent integration of all the subsystems and successful product delivery.

This page intentionally left blank

Chapter 5

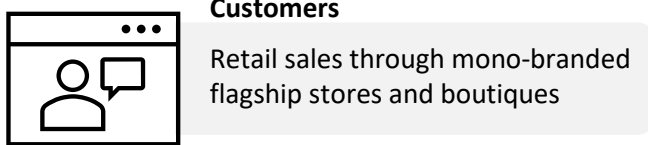
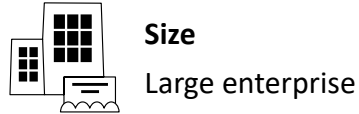
Cast study: A consumer product

In this chapter, the framework is applied to the development of a consumer product, specifically a household appliance. The system under consideration consists of a product platform and a set of accessories. The company aims to release the product on the market within a tight schedule, and it is willing to invest considerable resources to achieve this goal. The management provided the project details. The framework then used this information to assess the feasibility of meeting the time target and propose potential process improvements to either shorten the time-to-market or increase the likelihood of meeting the scheduled release.

Section 5.1 presents the general case study data resulting from the structured interview. Sections 5.2 and 5.3 report the organization and the project structure resulting from documentation and publicly available data analysis, as well as the semi-structured interview with a project participant. Section 5.3 summarizes the motivation for Agile adoption and the fitting of Agile into development processes traditionally used by the organization, as described by the interviewees. Section 5.4 describes the application of the framework to the project data provided by the organization, detailing all the implementation steps, and eventually, it derives process insights and draws the case study conclusion.

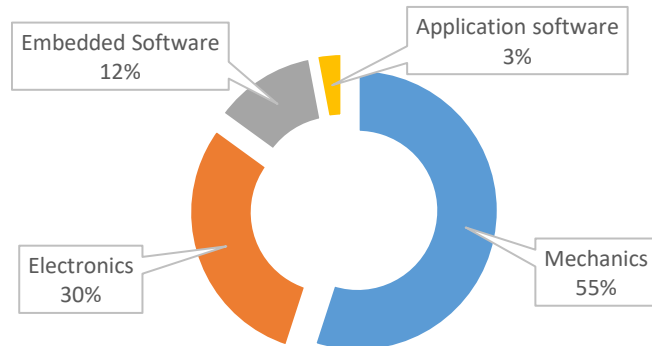
5.1 General case study data

Company B

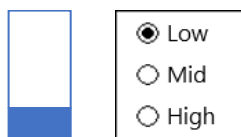


Product composition

Mechanics	55%
Electronics	30%
Embedded Software	12%
Application software	3%
<hr/>	
Total	100%
Degree of physicality	0.91

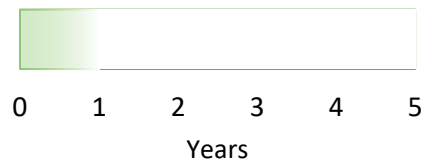


Customer involvement



- Customer only validating the product
- Customer in mildly involved (monthly meetings)
- Customer is heavily involved (weekly meetings)

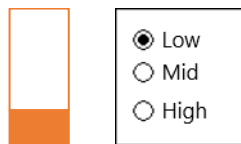
Experience with Agile



Sprint Length in weeks



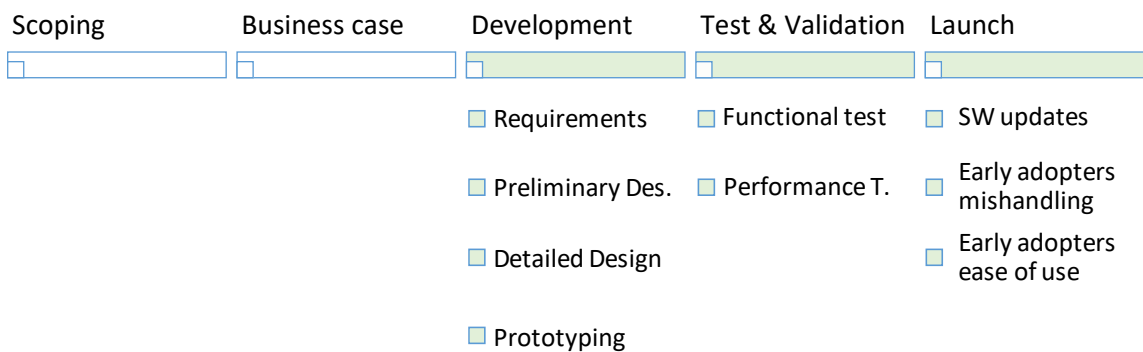
Impact on the organization



We are still evaluating a potential Agile transition. So far we do not really use Agile.

Scrum in the Development Process

- Agile fully implemented
- Agile used in some extent



Note: the company leverage the internal manufacturing facility for rapid prototyping of some components, however the product of the final system is outsourced

Interviewees

Name Interviewee B1
Position Head of product development dpt
Background Economy
Role in Scrum No formal role

Name Interviewee B2
Position deputy head of product development
Background Unknown
Role in Scrum No formal role

5.2 Organizational structure

The R&D division of organization B has the headquarters in Moscow, Russian Federation, but has a network of development partners spread over Germany, Russia, France, Italy, Switzerland, China, Turkey, South Korea, and Australia. Each partner is responsible for the development of a different element of the final product.

The organization mostly outsources the development of all the components constituting the product. The R&D division primarily coordinates the different design teams, ensuring that outcomes of R&D activities are consistent with the product vision. Particular efforts are also dedicated to quality assurance and after-sales service. The project presented here is handled by a project manager located in the Moscow headquarter, coordinating company development partners.

The company does not have its own production site. Products are manufactured in China, Poland, Hungary, Germany, Japan, Turkey, and France. The organization mainly trades in Russia, the commonwealth of independent states, and Poland.

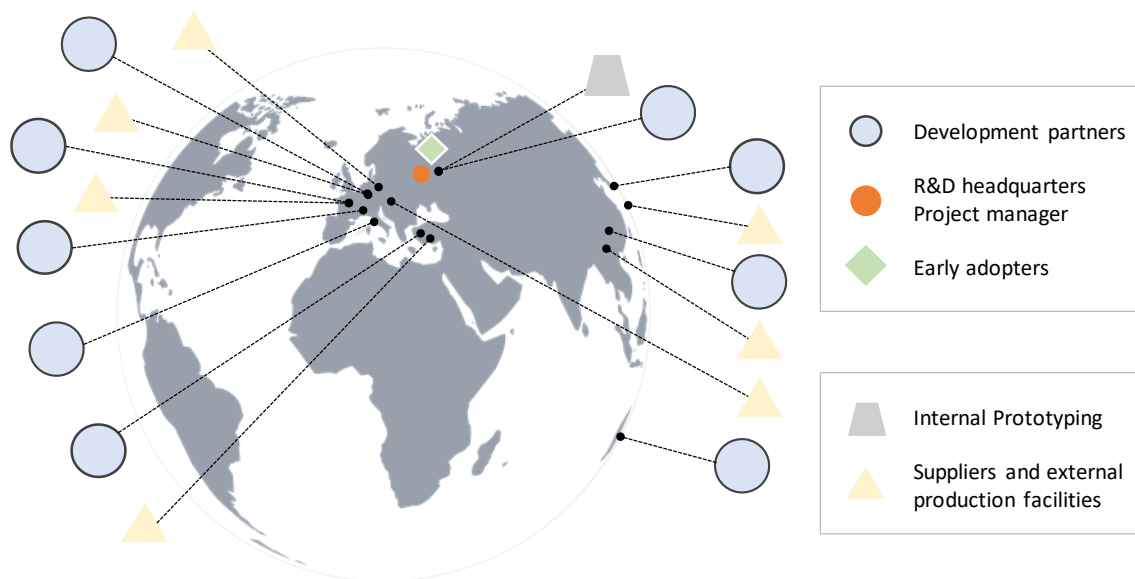


Figure 63. Case B – Organization of the development teams

5.3 Agile in the development process, motivation for Agile adoption

Organization B is a large company that employs more than a thousand people worldwide, with development partners spread over twelve different countries. As such, the company uses a Stage-Gate model as the overall management tool for development activities.

The company does not formally use Agile. However, some of the Agile principles are implemented during the stages called “Development”, “Test and validation”, and “Launch” (refer to Section 5.1 page 131).

Organization B is not directly interested in the Agile process. However, it is always interested in any approach that can potentially streamline the process and accelerate the development schedule. Therefore, if Agile can bring some benefits in any stage of development, they are willing to experiment with that. In this case study, the main driver for a potential Agile adoption relates to time savings.

5.4 CURSIVE deployment

5.4.1 Process Structuring

At the beginning of the project, there was a kick-off meeting with the management of the product development department. Company B provided an overview of the product vision and a preliminary concept. After this first introductory meeting, I asked the project manager and development team to define the set of tasks they envision for completing the project.

The team listed eighty-four tasks related to ten macro areas: Electronics, Mechanics, Software, Tooling, Manufacturing, Assembly line, Testing, Packaging, Certification, Patenting. Then, these tasks were collected in a DSM, and the dependencies’ structure was outlined. In the

DSM, cells on the diagonal correspond to the tasks, the marks in off-diagonal cells indicate tasks interactions. The marks in the row denote activities inputs (sub-diagonal marks), while marks in the column indicate the feedbacks (super-diagonal marks). In this case study, the team provided only deterministic feedback links, marked with the black filled circle symbol, i.e., probability of occurrence equal to 1. While defining the process architecture, the team and I realized that some tasks related to the definition of interfaces between subsystems were missing. Those tasks were added and, within two iterations, the final product backlog architecture was consolidated.

Figure 64 shows the resulting product backlog architecture the team produced. The backlog includes design activities (marked in green), rapid prototyping (yellow tasks), procurement or manufacturing (marked in orange), AIT activities (marked in blue), IP protection (purple tasks), certifications activities (marked in grey), packaging (light grey tasks) and assembly line (light blue).

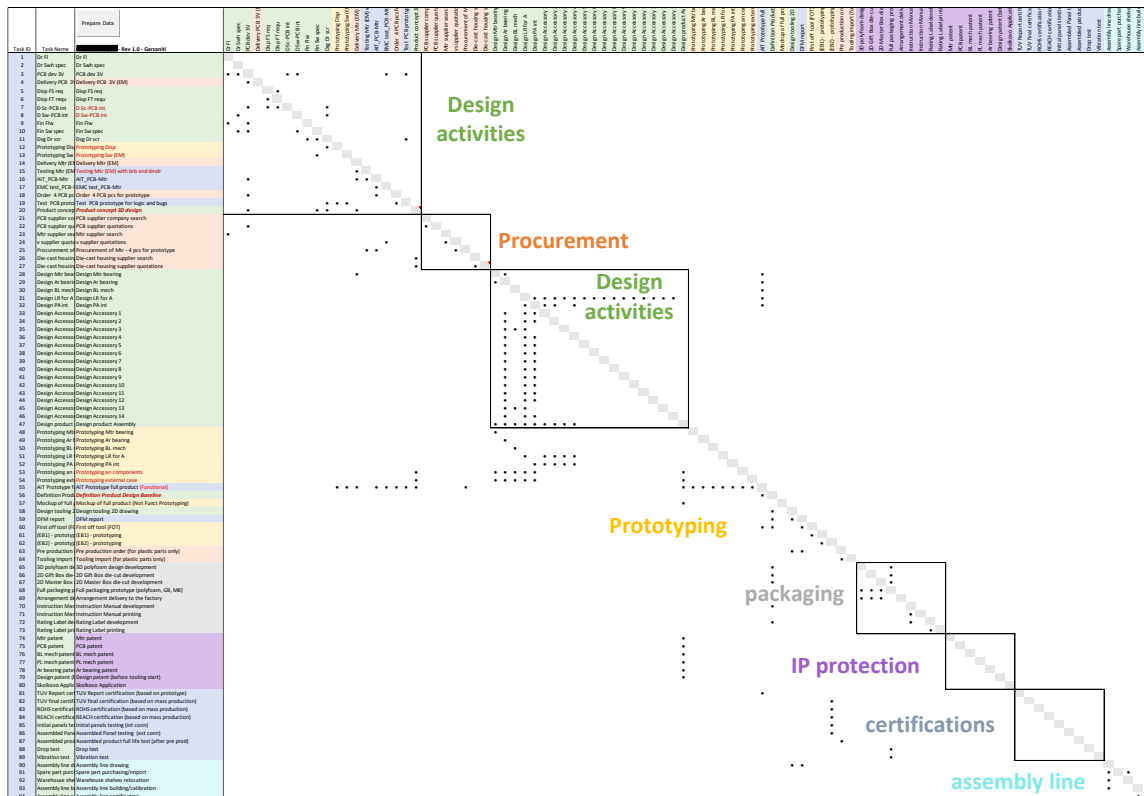


Figure 64. Input DSM for development of the household appliance

After consolidating the tasks list and the process structure, project participants were asked to provide additional data about time, cost, and resources required to perform the activities (Figure 65). The information is based on historical data, experts' opinion, or their combination. The main concern of the management was related to the time-to-market; therefore, they decided to provide only time data and focus all the analysis on the schedule.

Task ID	Prepare Data	Task Name	WP ID	Expertise	# Exp needed	Exp code	time (man-day)		
							min	mode	max
1	Dr Fl		1	E	1	1	1	2	20
2	Dr Swh spec		1	E	1	1	4	4	15
3	PCB dev 3V		1	E	1	1	29	58	65
4	Delivery PCB 3V (EM)		1	p	1	4	3	14	40
5	Disp FS req		1	E	1	1	3	17	20
10	Fin Sw spec		1	E	1	1	25	49	60
11	Dsg Dr scr		1	E	1	1	49	98	120
12	Prototyping Disp		1	p	1	4	60	101	120
13	Prototyping Sw (EM)		1	p	1	4	50	101	120
14	Delivery Mtr (EM)		1	p	1	4	75	124	150
15	Testing Mtr (EM) with brb and dmdr		1	AIT	1	5	3	16	60
21	PCB supplier company search		1	S	1	6	3	13	15
22	PCB supplier quotations		1	S	1	6	5	6	60
23	Mtr supplier search		1	S	1	6	0	2	40
24	v supplier quotations		1	S	1	6	0	1	15
25	Procurement of Mtr - 4 pcs for prototype		1	p	1	4	15	54	60
26	Die-cast housing supplier search		1	S	1	6	5	50	60
27	Die-cast housing supplier quotations		1	S	1	6	3	5	15
28	Design Mtr bearing		1	E	1	1	5	19	40
29	Design Ar bearing		1	E	1	1	5	43	50
65	3D polyfoam design development		3	E	1	1	5	15	30
66	2D Gift Box die-cut development		3	E	1	1	5	7	30
67	2D Master Box die-cut development		3	E	1	1	5	18	20
75	PCB patent		4	PT	1	8	15	53	60
81	TUV Report certification (based on prototype)		4	C	0	7	147	174	210
82	TUV final certification (based on mass production)		4	C	0	7	15	54	60
90	Assembly line drawing		8	E	5	1	10	11	30
94	Assembly line certification		12	p	9	4	10	40	45

Figure 65. Case study B - Complementary information provided within the product backlog architecture definition – work-packages traceability, expertise required, and time estimates.

The definition of the product backlog architecture was a collective effort involving the management team responsible for the project. Each team member provided tasks, input/out dependencies, and time estimates. The final product backlog architecture resulted from team consensus reached by adopting an iterative refinement process.

We leveraged the value of the Agile Manifesto of promoting collaboration between project participants and development partners, while providing a common language and structure to define the activities required to deliver the product. The resulting product backlog architecture offered the team a holistic view of the project.

While shaping the product backlog architecture, the framework provided the team with a chart to understand the viability of implementing Agile in the different portions of the project (Figure 66). The same color code used in the product backlog architecture is adopted: design tasks are marked in green, procurement or manufacturing tasks are marked in red, prototyping tasks are in yellow, AIT tasks in blue, IP protection in purple, certification in grey, packaging in light grey and industrialization tasks are marked light blue. The tasks in the upper pane represent the activities for which agile implementation is efficient in terms of time. In this case, fast iterations will lead to an equally rapid increase of product maturity without significantly impacting the schedule. Tasks in the lower pane represent the activities for which Agile implementation would be highly inefficient in terms of time. Iterating such tasks would lead to a schedule disruption.

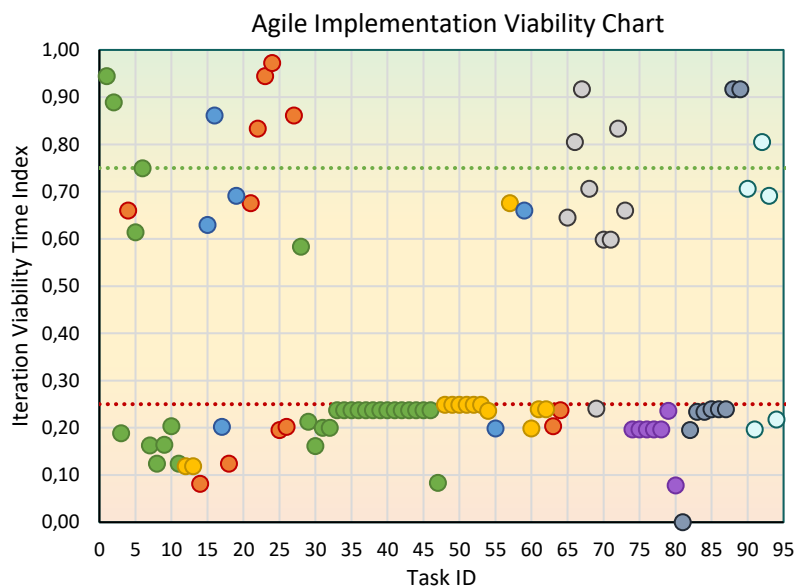


Figure 66. Case B – Agile Implementation viability

It is quickly noticeable from the chart that 68% of tasks lie in the non-viable area and less than 15% in the viable one. The time viability index for the entire project, AV_{prj}^T , is equal to 0.38. Therefore, based on this structuring reasoning exercise, it can be concluded that Agile is not a viable solution for the project under consideration. Nevertheless, some small iterations might be considered for a few tasks.

From the product perspective (Figure 67 shows the product DSM), we can also notice that the product exhibits a bus-modularity structure (Holttä-Otto & de Weck, 2007), i.e., one component connects to many other components, but none of the other components connects to each other. This product feature opened the question of optimizing the combined development of the product platform and product accessories to minimize the time to market while keeping customers' hype.

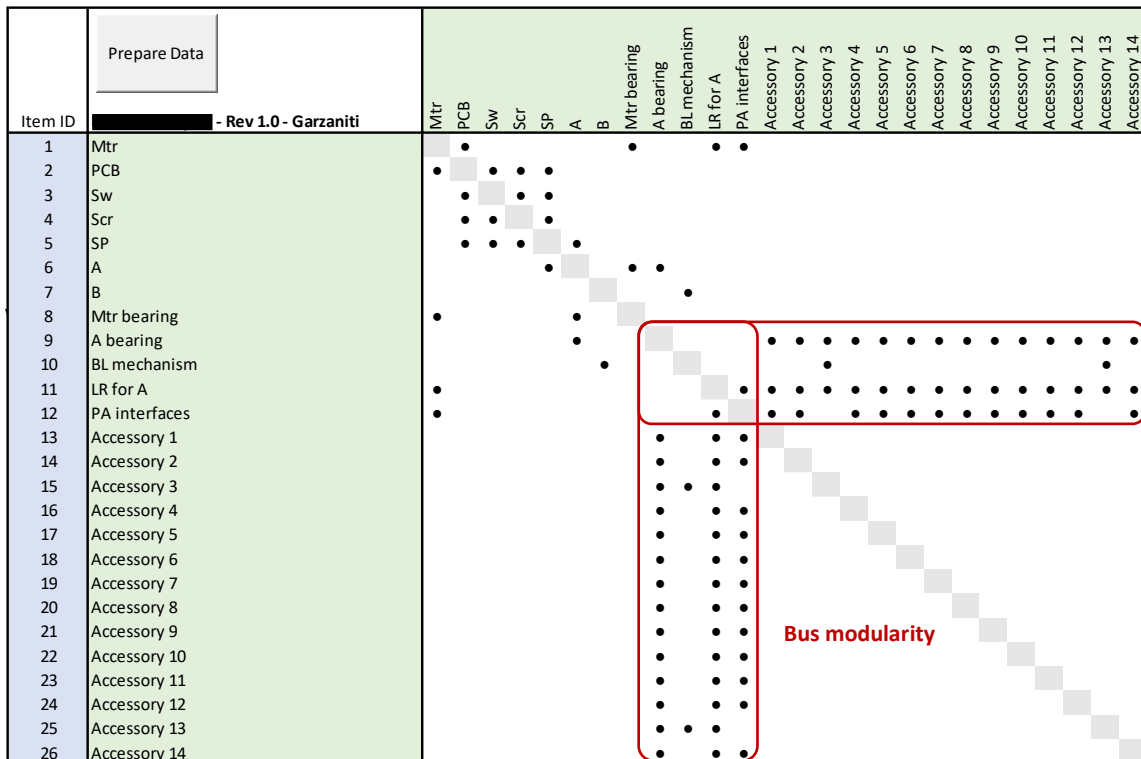


Figure 67. Product DSM of the household appliance

Specifically, the question was to understand the benefit of defining a set of primary accessories to be ready for the product launch, postponing the development of the secondary accessories to be released in the later months, or develop and release all in one. The simulation and planning CURSIVE capabilities are used to evaluate this tradeoff.

The consolidated product backlog architecture resulting from the structuring process (Figure 64), including the DSM and the complementary information (Figure 65), is used as input for the simulation step of the framework.

5.4.2 Simulation

Three thousand (3000) simulations (in batches, s , of 500) have been run to stabilize both mean and variance of time distribution within precision, $\varepsilon = 10^{-4}$ (Figure 68) according to equation (6), (7).

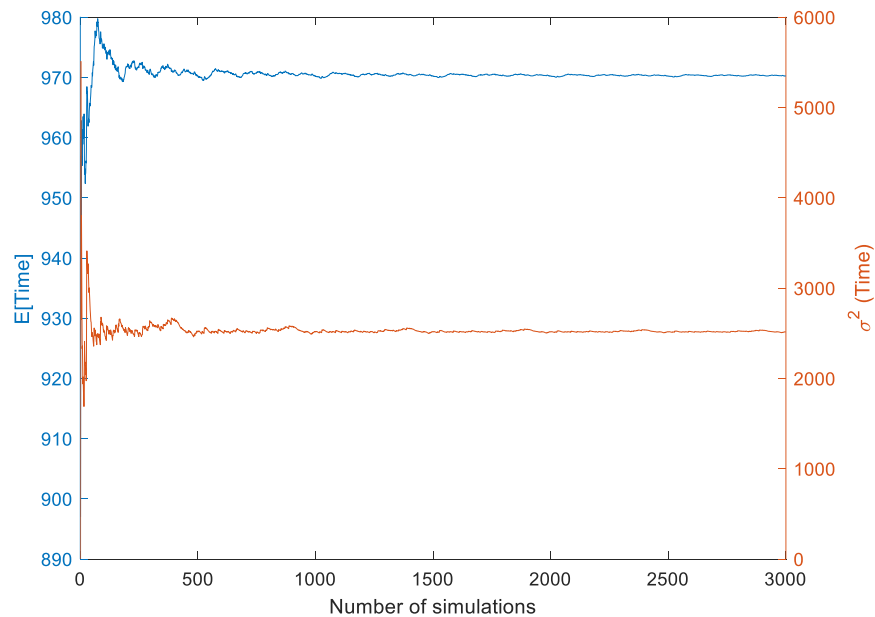


Figure 68. Mean and variance over the number of simulations for time distribution

As a result, the simulation provided a time distribution. Figure 69 shows the probability density functions (PDF) of simulated schedule outcomes.

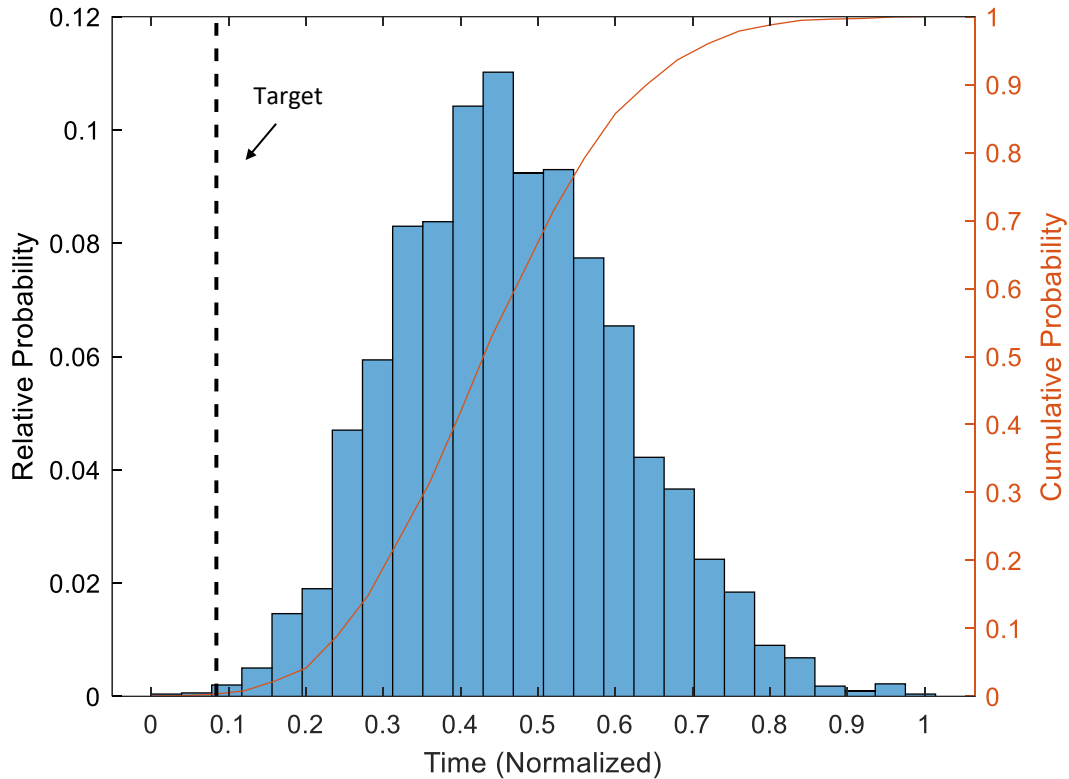


Figure 69. Case B - Simulation output Time Probability Distribution Functions (PDFs) normalized

The graph was helpful in quickly comparing process time against the scheduled target. We could immediately notice that the programmatic risks associated with the time objective were extremely high. There was less than a 10% likelihood to complete the project within the target timeframe, while the most likely scenario was about 24% longer than the schedule objective. This opened the question of whether the issue was the process structure or the sensitivity of the implementation strategy to some specific activities' uncertainty. This concern is further addressed by analyzing different implementation scenarios. Then the results are used to benchmark process performance identifying the source of programmatic risks.

5.4.3 Planning

In the planning stage, CURSIVE was provided with different time targets, trying to get more details on the different project implementation strategies. Two scenarios have been analyzed: the nominal scenario (tight schedule, target 0.07 T of time distributions in Figure 69) and the most likely scenario. Figure 70 shows the Gantt chart of the nominal scenario meeting time constraints.

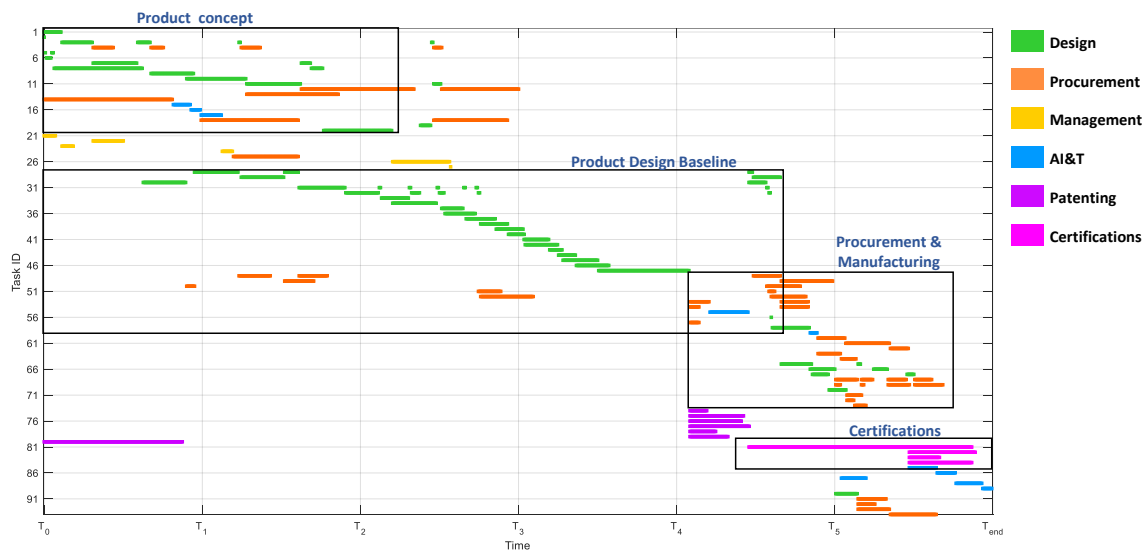


Figure 70. Case B- Gantt chart of the baseline scenario meeting time constraints, in green design tasks, in orange procurement and manufacturing, in yellow project management tasks, in blue AIT, in purple IP management, and in light purple certification tasks

Given the structure of the process, summarized in Section 5.4.1, Agile did not represent a viable solution for the considered project. Therefore, the algorithm for identifying the number of MVP and defining the Sprint backlog has not been deployed. Nevertheless, the algorithm has been used to define some activities clusters and set milestones to monitor the project execution.

Then the information gathered within the CURSIVE deployment has been collected and used to derive insights, thus provide recommendations.

5.5 Process insights

The comparison of the Gantt chart of two process implementation scenarios, the nominal scenario (Figure 71 - left) and the most likely scenario (Figure 71 - right), suggested that the development time issue relates to the process structure rather than the sensitivity to task durations.

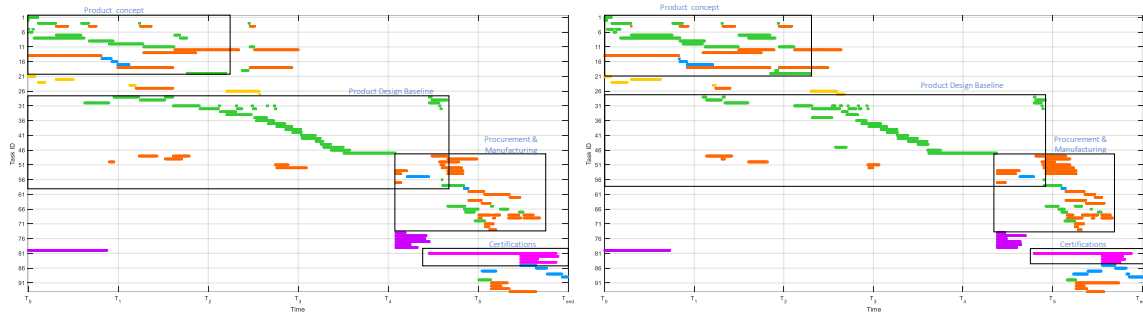


Figure 71. Case B - Comparison of different scenarios

The analysis of the normalized time required to perform the different activities as well as the iterations time (showed as different staked color bars) for a given scenario (Figure 72) confirmed this hypothesis. The development process is mostly sequential, with no iterations and no task overlaps.

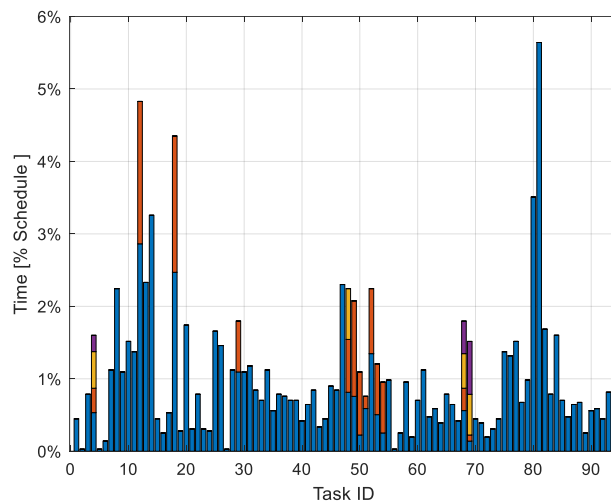


Figure 72. Case B - Tasks Time Breakdown

Further analysis of the PDP also showed that 65% of the time is spent on the Product baseline definition and 20% of the time on the development of the secondary accessories.

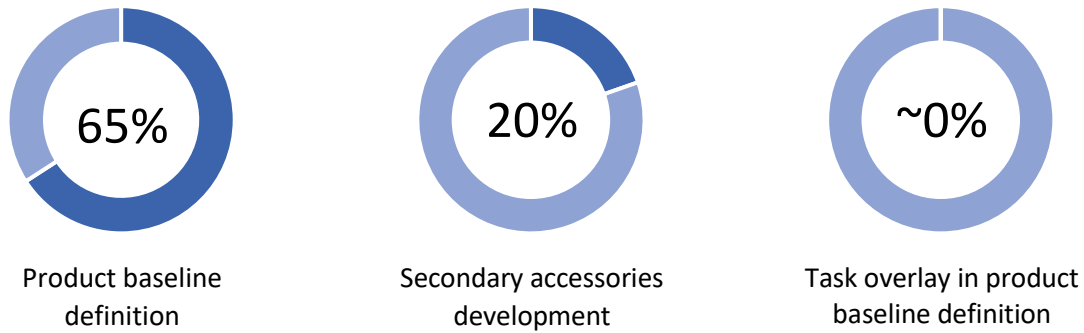


Figure 73. Summary of Process insights

Therefore, the following suggestions were made: 1) To defer the development of the secondary accessories until the completion of the “first product” account for requirements commonalities. 2) To overlap development tasks and prototyping activities to reduce lead time and early mitigate technical risks. This combined strategy would increase the likelihood of releasing the product within the target timeframe (Figure 74).

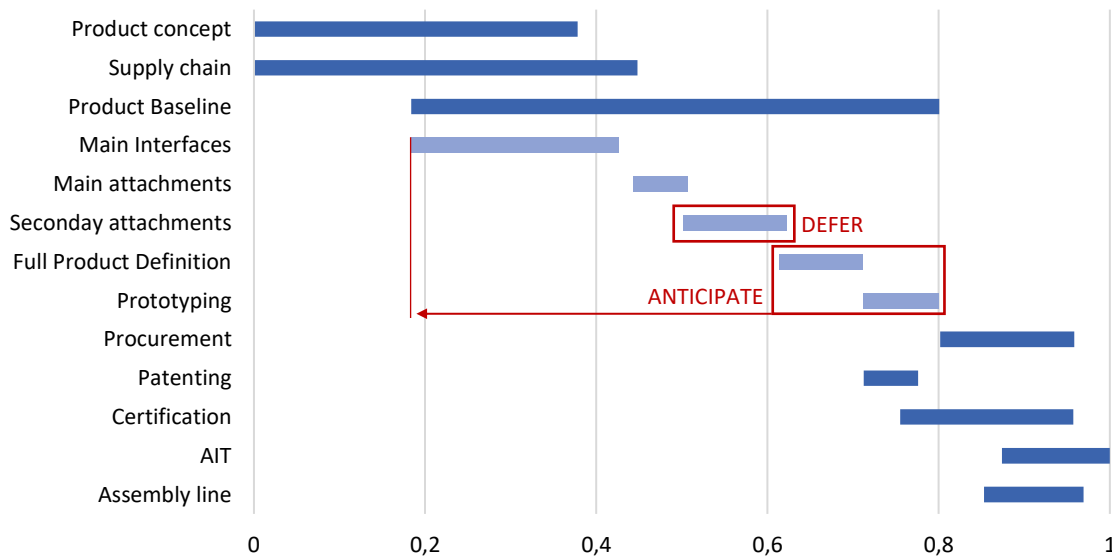


Figure 74. Suggested optimization

The process structure was amended according to the recommendations resulting from the analysis. Then the new process was simulated. Figure 75 shows the time probability density of the optimized process. It is immediately observable that there is a much higher likelihood to complete the project in time in this new configuration. Furthermore, if we assume the same risk posture of the non-optimized scenario, the time to market can be reduced by about 14% compared to the planned target.

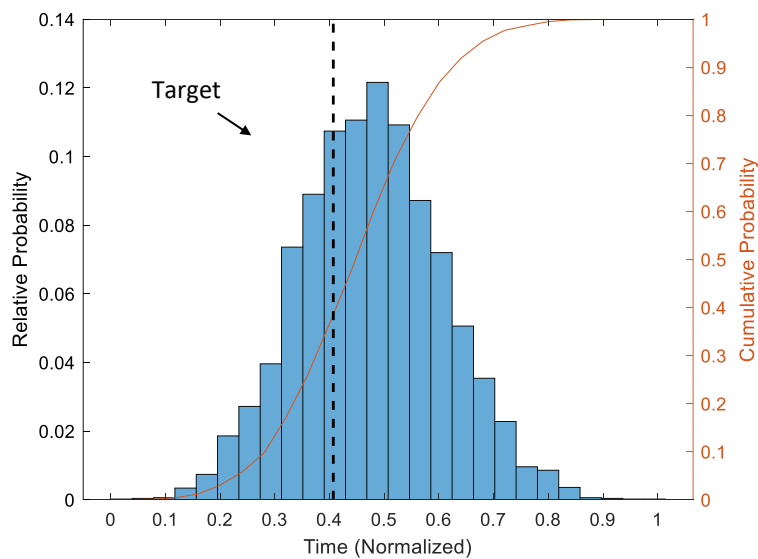


Figure 75. Case B – Simulation output of optimized process structure

Additional optimization can also be made at the resource-allocation level. Improving the coordination with the development partners and temporarily doubling the personnel working on the project for the required time frame will lower the time-to-market by 40% while having hypothetically no impact on the project's total cost. Increasing the full-time equivalent (FTE) for a short period might not affect the budget much while bringing significant benefits to the schedule. However, this scenario may affect other projects the organization is running. Not having a comprehensive picture of the R&D portfolio and knowing all the project interdependencies, this solution was discarded.

Based on these analyses, the process implementation strategy presented in Figure 76 has been recommended: 1) Starting the development of the product platform and primary accessories at T_0 , deferring the development of secondary accessories at T_5 . 2) Anticipating the prototyping and AIT activities between T_1 and T_3 . On the one hand, this solution optimized the speed of delivery, increasing the likelihood to release the product in time; on the other hand, it allows for earlier mitigation of technical risks reducing the risk for reworks.

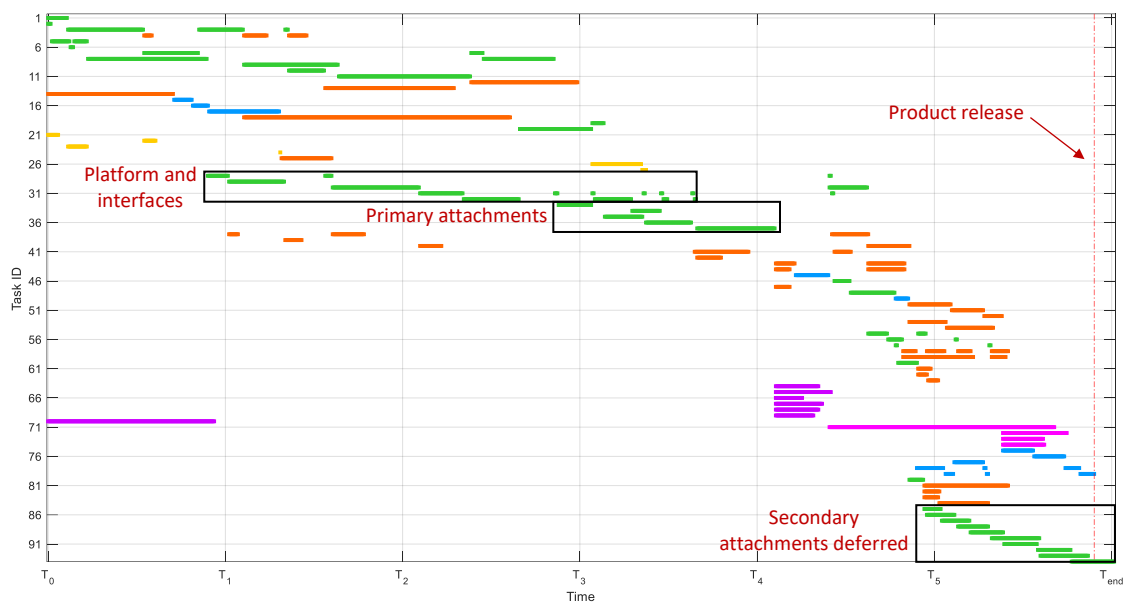


Figure 76. Recommended strategy

As we can observe in Figure 76, the complete set of secondary attachments will be ready immediately after the product release; however, these attachments might require additional tests before being ready for commercialization. Those tests are not included in the current Gantt chart and might require further investigations. Deferring the development of secondary attachments also has some implications in terms of design optimization. While the combined development of the product platform and the full set of accessories would ensure a system entirely optimized, deferring a subset of accessories might lead to a suboptimal solution. The system would be optimized for

product platform and primary set of attachments, while secondary ones are not taken into consideration in platform design and optimization. Potential issues deriving from this situation are mitigated by a careful interface definition and specifically by developing a standard interface (tasks 31 and 32 in the product backlog architecture).

5.6 Summary and interim conclusion

This chapter analyzed the development process of a consumer product and specifically a house appliance applying our framework. The analysis covered all the aspects of the system development, including design, prototyping, manufacturing, assembling, integration, testing, packaging, assembly line design and deployment, certification, and IP protection

After introducing the case study providing background information on both the project and the organization executing the project, and briefly discuss the motivation for adopting Agile, we have dived into the PDP structuring and planning analysis using CURSIVE. The analysis of the PDP has been organized according to the three macroblocks of our framework, namely *process structuring*, *simulating* (or *simulation*), and *planning*.

During the *structuring* phase, we deployed a set of methods and tools for reasoning about the structure of the decision problem. First, we have identified ninety-four tasks spanning over the whole development cycle. Then, those activities have been organized in a DSM, defining their interconnections (tasks dependencies) and outlining the order in which tasks shall be addressed. Tasks exhibited deterministic dependencies. The outcome of this exercise has been consolidated in the product backlog architecture. The product backlog architecture has been integrated with complementary information about the time and resources required to perform the activities.

Within the *structuring* phase, CURSIVE also assessed the viability of implementing Agile for the given project. The time viability index for the entire project, AV_{prj}^T , is equal to 0.38. This value implies that the project would not benefit from Agile adoption; on the contrary potential iterations would have a huge impact on the schedule, eventually jeopardizing the project. Therefore, it is not recommended to execute the project using Agile, even if some small iterations might be considered for few tasks. The detailed analysis of the Agile Viability charts (Figure 66) identified the few items for which iterations are feasible (about 14 tasks). Overall, the framework suggested executing the project adopting a traditional Stage-gate approach, with a light Agile-hybridization on electronics design tasks and packaging design tasks. The information resulting from the structural reasoning exercise has been then used as input for the *simulation* phase.

During the *simulation* phase, CURSIVE has analyzed the solutions satisfying the problem constraints and evaluates the overall process performance for different variables combinations and values. As an outcome of the simulation, we got a time distribution representing the feasible scenarios and the dataset, including all the process details for each considered scenario.

In the *planning* phase, we have explored in-depth the scenario that meets the time target offering an actionable plan. The results of the planning activities have been summarized in a set of charts (Figure 70, Figure 72, and Figure 73), offering a comprehensive picture, from multiple perspectives, of the baseline execution plan.

Together with the structuring reasoning exercise, the analysis of the project execution plans represented a fundamental activity in defining the degree of agility (intended as iterations viability) to deploy within the development of a hardware system. The first provided the rationale for Agile implementation. The latter provided means of coordination between activities executed adopting different implementation strategies (iterative vs. not iterative).

The combined outcome of the three phases of our framework provided valuable insights on the process performance in terms of time and technical risk. Figure 72 reported the normalized time required to perform the different activities, as well as the time of iterations. These data brought the focus on critical activities, such as task 81 (time-critical), and highlighted that the PDP is mostly sequential with no iterations and no tasks overlap. Further analyses also showed that 65% of the time is spent on the Product baseline definition and 20% of the time on the development of the secondary accessories. All this information can be used to evaluate potential strategies for process improvements (Figure 74). First, it has been suggested to defer the development of secondary attachments leveraging subsystems' commonalities and standard interfaces. Then, it has been suggested to overlap design and prototyping activities to reduce lead time and early mitigate technical risks. This combined strategy would increase the likelihood of releasing the product within the target timeframe, as presented in Figure 75.

It has also been investigated the possibility of additional optimizations at the resource-allocation level. Improving the coordination with the development partners and temporarily increasing the personnel working on the project for the required time frame will lower the time-to-market while having hypothetically no impact on the project's total cost (increasing FTEs for a short period might not affect the budget much while bringing significant benefits to the schedule). However, not having a comprehensive picture of the full R&D portfolio of the company and without knowing all projects' interdependencies, we did not recommend pursuing this solution.

As shown in the case study, CURSIVE is a versatile framework that provides project managers and development teams with quantitative means to underpin their decision-making process. It can be used to assess the feasibility of meeting a time target for a given process structure as well as propose and investigate potential process improvements to either shorten the time-to-market or increase the likelihood of meeting the scheduled release.

Based on the analysis conducted within this study, the following conclusions can be drawn.

- Team composition, location, and synchronization play a key role in the feasibility of Agile/hybrid-Agile approach implementation. One of the factors that made Agile not viable for this case study relates to the team theme. As presented in section 5.2, organization B has the headquarters in Moscow, Russian Federation, but has a network of development partners spread over nine different countries. Each partner is responsible for developing a subset of the final product. Furthermore, the company does not have its own production site but leverages a network of manufacturers spread over seven countries. This organization setting with *highly dispersed functional teams* is not ideal for Agile adoption.
 - Dispersed functional teams (instead of traditionally recommended co-located cross-functional teams) might lack a holistic perspective on the product, requiring a detailed set of system requirements in clear contrast with Agile theory.
 - Dispersed functional teams make difficult the development of incremental MVPs requiring additional logistic support for product shipment between the different development sites. Only functional MVPs related to the discipline assigned to a given development site are allowed, preventing the full exploitation of Agile potential in retiring technical risk through fast iterations at multiple stages of product development.
 - Dispersed teams might exhibit synchronization issues related to the different development speed different teams might have. Therefore, different product subsets might be ready at a different moment requiring additional coordination efforts for efficient project execution.

- Functional teams might exhibit issues related to the development culture. A complex project entails a large variety of disciplines involved, and each team might not share the same approach to the development. As a result, they might prioritize different system features at different stages of the product development leading to suboptimal solutions.
- This case study highlighted that System modularity is not only an essential factor for the effective implementation of Agile/hybrid-Agile, but it generally represents a crucial driver in the process structuring. From the Agile perspective, modularity drives iterations cost and time (high modularity allows for more and faster iterations). From the process structuring perspective, modularity allows for different prioritization strategies of development activities. For instance, in this case study, the system exhibited a bus modularity structure: system modularity index (SMI), equal to 0.157 (Figure 77).

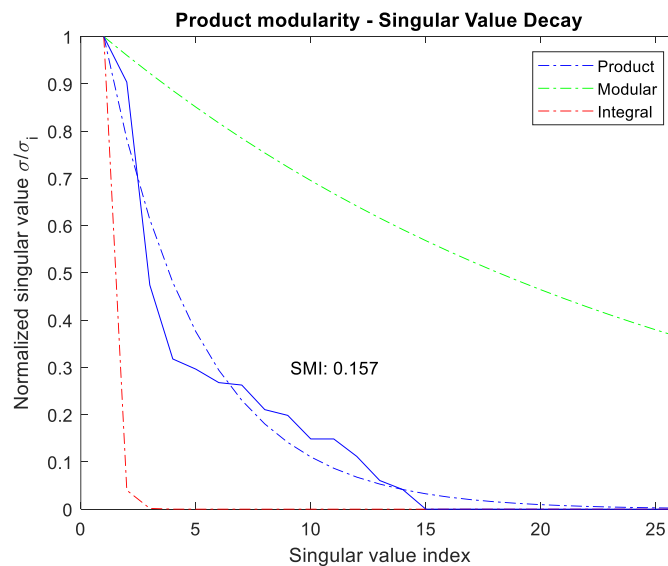


Figure 77. Degree of modularity

The SMI index is based on the exponential decay approximating the actual decay structure of sorted singular values of the product DSM (Figure 67). For additional details on the SMI

formulation., please refer to eq. (6) in (Holtta-Otto & de Weck, 2007). The bus modularity, combined with a careful interface definition and standardization, enabled the deferral of secondary attachments development, increasing the likelihood to release the product according to the time target (Figure 75 and Figure 76).

- Outsourcing the development of all the components constituting the product hampers the possibility of developing the entire product through iterations and incremental MVPs. Nevertheless, it opens the question of the number and the type of MVPs/prototypes to produce. The R&D division responsible for the project coordination and product integrations might still need a set of MVPs/prototypes alongside digital models (or a digital twin, if any) for verification and validation purposes.
- Product certification might represent a significant impediment to iterative development. As we can observe in Figure 70 (light-purple tasks), certification accounts for a considerable amount of time in the project schedule. Every new iteration on the product would require an additional set of certification activities, ultimately jeopardizing the project schedule.

“Simplicity is the ultimate sophistication”.
(Leonardi da Vinci)

Chapter 6

Conclusion

The thesis presented a decision support system for Agile development of complex hardware systems. This research work contributes to the current debate in the field on the viability of implementing Agile in the context of hardware systems development. The proposed framework aims to support engineering teams and project managers in structuring and planning development projects. This last chapter summarizes the main findings of the research. It highlights the thesis contributions, states the limitations, and sets the basis for future research.

6.1 Thesis summary

The thesis has first identified and discussed the gaps in the current Agile theory contextualized in the development of physical products, as well as the challenges and opportunities arising from the implementation of Agile in hardware projects. Then it has proposed a methodology and a tool to fill those gaps and exploit the opportunity opened by Agile adoption.

Chapter 1 introduced the problem and provided the rationale for the research opportunity. It has been acknowledged that the need to develop products under uncertain, volatile, complex, and ambiguous (VUCA) conditions drove and is still driving companies in all industries to focus on

streamlining the development processes and eventually looking for new development methodologies. In this context, Agile methods have drawn considerable attention. Nevertheless, despite the large variety of frameworks and practices available, organizations still struggle to identify and implement a development process structure that best fits their needs, thus getting the maximum benefit. Therefore, the need for a methodology to support the structuring of Agile or Hybrid-Agile product development for hardware systems underpinning the decision-making process by quantitative analyses and statistical evidence has been identified.

Chapter 2 critically reviewed the bodies of knowledge that frame the research. It has provided the foundations of Agile theory and specifically Agile- Scrum, and it offered an overview of the literature on product development and discussed current project management techniques. Chapter 2 has also reported the results of the field research conducted to validate the gaps identified in the literature and better inform the research questions.

Chapter 3 proposes the decision support system Agile development of hardware systems (CURSIVE). It includes an analytical approach to managing development activities within a hardware project and consists of three macroblocks, namely *structuring*, *simulating*, and *planning* (Figure 19). A cross-blocks layer, the *representing or viewing* layer, serves as a graphical user interface. The decision support system has been implemented in an integrated tool. Chapter 3 also addressed CURSIVE validity, specifically the validity of the process model, the data model, the graphical user interface, as well as the general and face validity. Given the lack in the literature of a formal methodology for validating decision support systems (few authors addressed this concern), this thesis also formulated a validation framework as a comprehensive synthesis of the works in related fields. This framework has been then used to guide the validation process.

In chapter 4 and chapter 5, the proposed framework is applied to two case studies for verification and validation purposes.

The first case study, presented in chapter 4, entailed the development of an optical telecommunication payload for a nanosatellite mission. The case study covered the entire development process of the system, from the design phase to the assembling, integration, and testing (AIT) of the flight model. The initial sections of the case study reported the organization and the project structure, summarizing the motivation for Agile adoption and the fitting of Agile into development processes traditionally used by the organization. Then, the framework is applied to the project data, detailing all the implementation steps. Eventually, process insights are derived, and the case study conclusion is drawn. In this first study, it was also possible to benchmark CURSIVE estimates against actual development data, verifying the framework capabilities and validating the accuracy of the forecasts.

In the second case study, presented in chapter 5, the framework is applied to the development of a consumer product. The company aimed to release the product on the market within a tight schedule and was willing to invest considerable resources to achieve this goal. The challenge here was to support the management in optimizing the development to meet the time target, defining the best strategy for developing the product platform and the product accessories. By applying the framework to this second case study, it was possible to test its flexibility and ability to adapt to the different project contexts.

This last chapter draws conclusions, summarizes thesis contributions, informs on the limitation of our work, and sets the bases for future research.

6.2 Thesis Contributions

The thesis contributions are unfolded starting from the answers to the research questions defined in Section 1.4. Within the answer to the research questions, the contributions are highlighted.

RQ 1

How to understand when and how to use Agile methods within the development process physical systems based on the specific project context and system features?

This thesis has proposed a methodology that assesses the viability of implementing the Agile Scrum approach accounting for the project context and the systems feature. Such methodology includes the Agile viability indexes (Section 3.1.5), two quantitative metrics that, relying on the tasks' decomposition and on the time and cost estimates, assess to which extent is viable to implement Agile in a given project. Based on the current state of the practice, thresholds in terms of cost and time have been set to recommend the use of Agile, Hybrid-Agile, or traditional development processes. The methodology also offers high-level suggestions on the task granularity and system modularity level, even if it does not explicitly address the two concepts, supporting the design team in the decision-making process.

Such methodology and related metrics attempt to solve the long-standing debate in the product development community on the effectiveness and the potential benefits of using Agile. The community is currently split into Agile supporters and opponents. The former believe that Agile is the solution to be adopted in every project; the latter believe that Agile should never be used. From the case studies analysis, it can be noticed that the solution is a middle way. Some tasks, project portions, or entire projects may benefit from Agile adoption allowing for rapid technical risk retirement and reducing the technical dept. In some other situations, the iterative nature of Agile

may jeopardize the project, and therefore traditional approaches shall be preferred. By deploying the proposed framework, project managers and design teams can reason on the question supported by quantitative analysis. Such an approach represents a significant shift in perspective of the current literature, moving from the analysis of challenges and opportunities entailed by Agile development to the analysis of contextual variables to evaluate the viability of Agile implementation.

It has also been developed a predictive model to simulate the process implementation (Section 3.2), thus assessing the impact of a given process structure on the process performance, namely time and cost. The simulation results provide additional insight into the feasibility of a given implementation strategy to meet potential schedule and budget constraints. Such a model is able to simulate complex iteration paths and independently manage the number and the position of the iterations within the product development. It represents significant advance respect to the literature models that can simulate only simple iterations patterns modeled by manual input.

The results of such predict model allow navigating programmatic and technical tradeoffs. The performance benchmark of alternative project implementations offers valuable information for planning and executing engineering projects meeting the stakeholders' needs while accounting for organization capabilities and resource availability.

RQ 2

How to support project managers and engineering teams in structuring and executing Agile or Hybrid-Agile methods within product development projects?

A framework to support the project structuring (section 3.1) and planning (3.3) has been developed. First, it has been introduced the concept of *product backlog architecture*. The benefit of defining a backlog architecture over a traditional product backlog is twofold. First, project

participants are not only informed on the activities to be performed (i.e., user stories) but also are aware of their relationships. This will enable decision-makers to reshuffle or redefine the high-level user stories to mitigate programmatic risks and meet project milestones. Second, Agile teams move from a qualitative definition of the story points to a quantitative definition of FOMs (i.e., time and cost), complemented by information on the uncertainty associated with their formulation. This provides a common understanding of the effort related to the task, retiring the risk linked to the human factor.

In the product backlog architecture, tasks covering the entire product development cycle are included, also accounting for external dependencies such as procurement and manufacturing. This perspective complements the traditional Agile theory that lacking this aspect has caused several problems in organizations implementing Agile for physical products.

Since Agile for hardware significantly differs from the software version because of the non-homogenous development environments requiring expertise from different disciplines, the expertise information has been included in the *product backlog architecture* alongside time and cost estimates. Naturally, following the Agile principles, tasks are not pre-assigned to people, but the disciplines related to the tasks are marked, including internal resources (i.e., development team) and external resources (i.e., procurement and manufacturing). Those data will be used during the project execution simulation, ensuring correct resource allocation and leveling.

The proposed model also supports engineering teams in the sprint planning activities and Minimum Viable Product Definition. Combining simulation results with the product backlog architecture, the framework identifies an optimal number of MVPs and suggests a Sprints planning strategy (section 3.3) through a heuristic approach. The model then summarizes all cost, time, and

Sprint backlog information facilitating development team activities. It should be emphasized that the model is not intended to replace team planning activities but rather aims to provide quantitative analyses and a common starting point for classic Scrum activities.

The thesis work also proposes a multi-tier architecture to coordinate Agile with the traditional Stage-gate process in the context of multi-party consortia (Section 3.1.4). Such architecture ensures the mapping of the Product Backlog item to the work packages and enables traceability of consortium requirements to the Agile organization user stories.

Lastly, a taxonomy of Minimum Viable Products (Section 3.3.2) has been developed. MVPs are classified based on the level of fidelity, the type of artifacts, the representation mode, the verification and validation activities enable by the MV. By mapping verification and validation activities on the MVP artifacts, such a taxonomy allows development teams to trade off engineering efforts required to produce the MVP versus the risk retired by the Sprint outcome. A notional map of a tradeoff between Sprint length, V&V activities, and MVP artifacts is then also presented.

6.3 Limitations and Future Work

The work proposed here has a few limitations related to the validation. Within this thesis, considerable efforts have been dedicated to ensuring the validity of the decision support system and specifically to the validity of the process model, the data model, the graphical user interface, as well as to the CURSIVE general and face validity (as presented in chapter 3, section 3.5). However, even though the framework has also been verified on a set of case studies (chapters 4 and 5), a full validation would require an experimental setup to compare the implementations of the same project following the recommendations provided by our model, Stage-gate project management experts,

and Agile coach or Scrum master experts, plus control groups. This effort would require a relatively large sample size of complex engineering projects within different industrial settings to reach statistical significance. Possible future research works may address this point.

Concerning the range of applications the framework can serve, as discussed in chapter 3, section 3.5.4, it has been specifically designed and applied to physical and cyber-physical products. As presented in chapters 4 and 5, the framework can cover a wide range of applications. It has been applied to the development of a space system as well as the development of a consumer product. It has not been tested on “pure” software systems; thus, we cannot claim the applicability to this kind of product. Concerning the R&D composition and location, the DSS has been deployed and tested in cases of collocated (chapter 4) and sparse/dispersed R&D teams (chapter 5), providing in both situations valuable process insights (sections 4.5 and 5.5). Possible future research works may test the framework capability for a larger set of applications. In particular, it would be interesting to evaluate CURSIVE performance in the context of product-service systems.

Another direction of future work can be integrating our model with a broader data-driven systems engineering framework, e.g., Valispace (Valispace GmbH, 2019), to improve its analytical capabilities and increase the accuracy of estimates.

References

- Adelman, L. (1991). Experiments, Quasi-Experiments, and Case Studies: A Review of Empirical Methods for Evaluating Decision Support Systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21(2), 293–301. <https://doi.org/10.1109/21.87078>
- Age-of-Product.com. (2018). *Agile Transition: Scrum Master Duties, Serving a Single Team (Survey Results)*. <https://age-of-product.com/scrum-master-duties/>
- Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile Software Development Methodologies: Survey of Surveys. *Journal of Computer and Communications*, 05(05), 74–97. <https://doi.org/10.4236/jcc.2017.55007>
- Allman, E. (2012). Managing technical debt. *Communications of the ACM*, 55(5), 50–55. <https://doi.org/10.1145/2160718.2160733>
- Alqudah, M., & Razali, R. (2016). A Review of Scaling Agile Methods in Large Software Development. *International Journal on Advanced Science, Engineering and Information Technology*, 6(6), 828–837. <https://doi.org/10.18517/IJASEIT.6.6.1374>
- Anderson, E., Lim, S. Y., & Joglekar, N. (2017). Are More Frequent Releases Always Better? Dynamics of Pivoting, Scaling, and the Minimum Viable Product. *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*. <https://doi.org/10.24251/hicss.2017.705>
- Archibald, R. D. (2003). *Managing High-Technology Programs and Projects*. 396.
- Atlassian. (2021). *Jira | Project Tracking Software*. <https://www.atlassian.com/software/jira>
- Atzberger, A., Gerling, C., Schrof, J., Schmidt, T. S., Weiss, S., & Paetzold, K. (2019). Evolution of the Hype around Agile Hardware Development. *Proceedings - 2019 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2019*. <https://doi.org/10.1109/ICE.2019.8792637>

References

- Atzberger, A., Nicklas, S., Schrof, J., Weiss, S., & Paetzold, K. (2020). *Agile Entwicklung Physischer Produkte - Eine Studie zum aktuellen Stand der industriellen Praxis*. https://doi.org/10.18726/2020_5
- Atzberger, A., & Paetzold, K. (2019). Current challenges of agile hardware development: What are still the pain points nowadays? *Proceedings of the International Conference on Engineering Design, ICED, 2019-Augus*, 2209–2218. <https://doi.org/10.1017/dsi.2019.227>
- Automotive Agile PEP. (2018). *Automotive Agile PEP-Survey Report: Perspectives of Agile Product Development Processes in the Automotive Ecosystem*. www.agile-auto-pep.com
- Balint, T. S., & Freeman, A. (2017). *Designing the design at JPL'S innovation foundary*. <https://doi.org/10.1016/j.actaastro.2017.04.026>
- Batra, D. (2018). Agile values or plan-driven aspects: Which factor contributes more toward the success of data warehousing, business intelligence, and analytics project development? *Journal of Systems and Software*, 146, 249–262. <https://doi.org/10.1016/j.jss.2018.09.081>
- 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., & Thomas, D. (2001). *Manifesto for Agile Software Development*. <http://agilemanifesto.org/>
- Begel, A., & Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, 255–264. <https://doi.org/10.1109/ESEM.2007.12>
- Ben Issa, S., & Tu, Y. (2020). A survey in the resource-constrained project and multi-project scheduling problems. *Journal of Project Management*, 5, 117–138. <https://doi.org/10.5267/j.jpmm.2019.11.001>
- Berget, I., Mevik, B.-H., & Naes, T. (2008). New modifications and applications of fuzzy C-means methodology. *Computational Statistics & Data Analysis*, 52, 2403–2418. <https://doi.org/10.1016/j.csda.2007.10.020>

- Bergweiler, G., Hansen, J. O., & Dörfer, M. (2019). Agile Development with Physical Prototypes for a Better Project Planning. *ATZ Worldwide*, 121(7–8), 44–47. <https://doi.org/10.1007/s38311-019-0075-6>
- Bianchi, M., Marzi, G., & Guerini, M. (2018). Agile, Stage-Gate and their combination: Exploring how they relate to performance in software development. *Journal of Business Research*. <https://doi.org/10.1016/J.JBUSRES.2018.05.003>
- Blessing, L. T. M., & Chakrabarti, A. (2009). DRM, a design research methodology. In *DRM, a Design Research Methodology*. <https://doi.org/10.1007/978-1-84882-587-1>
- Boehm, B. (2004). *Balancing Agility and Discipline: A Guide for the Perplexed* (pp. 1–1). https://doi.org/10.1007/978-3-540-24675-6_1
- Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 22(5), 30–39. <https://doi.org/10.1109/MS.2005.129>
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61–72. <https://doi.org/10.1109/2.59>
- Borenstein, D. (1998). Towards a practical method to validate decision support systems. *Decision Support Systems*, 23(3), 227–239. [https://doi.org/10.1016/S0167-9236\(98\)00046-3](https://doi.org/10.1016/S0167-9236(98)00046-3)
- Bott, M., & Mesmer, B. (2020). An Analysis of Theories Supporting Agile Scrum and the Use of Scrum in Systems Engineering. *Engineering Management Journal*, 32(2), 76–85. <https://doi.org/10.1080/10429247.2019.1659701>
- Boukhayma, K., & Elmanouar, A. (2016). Evaluating decision support systems. *International Conference on Intelligent Systems Design and Applications, ISDA, 2016-June*, 404–408. <https://doi.org/10.1109/ISDA.2015.7489263>
- Box, G. E. P. (1979). Robustness in the Strategy of Scientific Model Building. In *Robustness in Statistics* (pp. 201–236). Elsevier. <https://doi.org/10.1016/b978-0-12-438150-6.50018-2>

References

- Brhel, M., Meth, H., Maedche, A., & Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. *Information and Software Technology*, 61, 163–181. <https://doi.org/10.1016/J.INFSOF.2015.01.004>
- Briatore, S., & Golkar, A. (2021). Estimating Task Efforts in Hardware Development Projects in a Scrum Context. *IEEE Systems Journal*. <https://doi.org/10.1109/JSYST.2021.3049737>
- Browning, T. R., & Eppinger, S. D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*, 49(4), 428–442. <https://doi.org/10.1109/TEM.2002.806709>
- Buijs, J. (2003). Modelling Product Innovation Processes, from Linear Logic to Circular Chaos Early Delft Period. In *CREATIVITY AND INNOVATION MANAGEMENT* (Vol. 12). <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8691.00271>
- Burger, N., Demartini, M., Tonelli, F., Bodendorf, F., & Testa, C. (2017). Investigating flexibility as a performance dimension of a Manufacturing Value Modeling Methodology (MVMM): a framework for identifying flexibility types in manufacturing systems. *Procedia CIRP*, 63, 33–38. <https://doi.org/10.1016/j.procir.2017.03.343>
- Butta, R., Kamaraju, M., & Sumalatha, V. (2021). Heuristic methods for data clustering. In *Artificial Intelligence in Data Mining* (pp. 65–86). Elsevier. <https://doi.org/10.1016/b978-0-12-820601-0.00003-3>
- Camps, A., Golkar, A., Gutierrez, A., Ruiz De Azua, J. A., Munoz-Martin, J. F., Fernandez, L., Diez, C., Aguilera, A., Briatore, S., Akhtyamov, R., & Garzaniti, N. (2018). FSSCAT, the 2017 copernicus masters’ “ESA sentinel small satellite challenge” winner: A federated polar and soil moisture tandem mission based on 6U Cubesats. *International Geoscience and Remote Sensing Symposium (IGARSS), 2018-July*, 8285–8287. <https://doi.org/10.1109/IGARSS.2018.8518405>
- Chander, S., & Vijaya, P. (2021). Unsupervised learning methods for data clustering. *Artificial Intelligence*

- in *Data Mining*, 41–64. <https://doi.org/10.1016/b978-0-12-820601-0.00002-1>
- Chiriac, N., Hölttä-Otto, K., Lysy, D., & Suk Suh, E. (2011). Level of Modularity and Different Levels of System Granularity. *Journal of Mechanical Design*, 133(10). <https://doi.org/10.1115/1.4005069>
- Cho, S. H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management*, 52(3), 316–328. <https://doi.org/10.1109/TEM.2005.850722>
- Chuang, S. W., Luor, T., & Lu, H. P. (2014). Assessment of institutions, scholars, and contributions on agile software development (2001-2012). *Journal of Systems and Software*, 93, 84–101. <https://doi.org/10.1016/j.jss.2014.03.006>
- Cooper, R. G. (1990). Stage-gate systems: A new tool for managing new products. *Business Horizons*, 33(3), 44–54. [https://doi.org/10.1016/0007-6813\(90\)90040-I](https://doi.org/10.1016/0007-6813(90)90040-I)
- Cooper, R. G. (2016). Agile–Stage-Gate Hybrids. *Research-Technology Management*, 59(1), 21–29. <https://doi.org/10.1080/08956308.2016.1117317>
- Cooper, R. G., & Sommer, A. F. (2016). The Agile–Stage-Gate Hybrid Model: A Promising New Approach and a New Research Opportunity. *Journal of Product Innovation Management*, 33(5), 513–526. <https://doi.org/10.1111/jpim.12314>
- Cooper, R. G., & Sommer, A. F. (2018). Agile–Stage-Gate for Manufacturers. *Research-Technology Management*, 61(2), 17–26. <https://doi.org/10.1080/08956308.2018.1421380>
- Costa, R., & Sobek, D. K. (2003). Iteration in engineering design: Inherent and unavoidable or product of choices made? *Proceedings of the ASME Design Engineering Technical Conference*, 3, 669–674. <https://doi.org/10.1115/detc2003/dtm-48662>
- Darpe, S., Beckman, S., Ferlin, T., Havenhill, M., Parrot, E., & Harcula, K. (2020). Method for tracking and communicating aggregate risk through the use of model-based systems engineering (MBSE) tools.

References

- Journal of Space Safety Engineering*, 7(1), 11–17. <https://doi.org/10.1016/J.JSSE.2020.01.001>
- Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 224–227. <https://doi.org/10.1109/TPAMI.1979.4766909>
- De Amorim, R. C., & Hennig, C. (2015). Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324, 126–145. <https://doi.org/10.1016/j.ins.2015.06.039>
- Deiningner, M., Daly, S. R., Lee, J. C., Seifert, C. M., & Sienko, K. H. (2019). Prototyping for context: exploring stakeholder feedback based on prototype type, stakeholder group and question type. *Research in Engineering Design*, 30, 453–471. <https://doi.org/10.1007/s00163-019-00317-5>
- Diebold, P., & Dahlem, M. (2014). Agile practices in practice - A mapping study. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/2601248.2601254>
- Digital.ai, & VersionOne Inc. (2020). 14th annual State of Agile Report. In *stateofagile.com*. <https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494>
- Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119, 87–108. <https://doi.org/10.1016/j.jss.2016.06.013>
- Dorst, K., & Cross, N. (2001). Creativity in the design process: Co-evolution of problem-solution. *Design Studies*, 22(5), 425–437. [https://doi.org/10.1016/S0142-694X\(01\)00009-6](https://doi.org/10.1016/S0142-694X(01)00009-6)
- Douglass, B. P. (2016a). Agile Systems Engineering. In *Agile Systems Engineering*. <https://doi.org/10.1016/B978-0-12-802120-0.00008-4>
- Douglass, B. P. (2016b). Chapter 2 – What Are Agile Methods and Why Should I Care? In *Agile Systems Engineering* (pp. 41–84). <https://doi.org/10.1016/B978-0-12-802120-0.00002-3>

-
- Dove, R. (2002). *Response Ability: The Language, Structure, and Culture of the Agile Enterprise* (Vol. 1).
<http://books.google.com/books?id=M3wnwrol2YgC&pgis=1>
- Dove, R., & LaBarge, R. (2014a). 8.4.2 Fundamentals of Agile Systems Engineering - Part 2. *INCOSE International Symposium*, 24(1), 876–892. <https://doi.org/10.1002/I.2334-5837.2014.TB03187.X>
- Dove, R., & LaBarge, R. (2014b). 8.4.1 Fundamentals of Agile Systems Engineering – Part 1. *INCOSE International Symposium*, 24(1), 859–875. <https://doi.org/10.1002/I.2334-5837.2014.TB03186.X>
- Dove, R., Schindel, W., & Hartney, R. W. (2017). Case study: Agile hardware/firmware/software product line engineering at Rockwell Collins. *11th Annual IEEE International Systems Conference, SysCon 2017 - Proceedings*. <https://doi.org/10.1109/SYSCON.2017.7934807>
- Eckert, C., Albers, A., Bursac, N., Chen, H. X., Clarkson, P. J., Gericke, K., Gladysz, B., Maier, J. F., Rachenkova, G., Shapiro, D., & Wynn, D. (2015). Integrated product and process models: Towards an integrated framework and review. *Proceedings of the International Conference on Engineering Design, ICED, DS 80-02*, 389–398.
- ECSS. (2010). *ECSS-E-HB-10-02A - Verification guidelines*. <https://ecss.nl/hbstms/ecss-e-10-02a-verification-guidelines/>
- ECSS. (2018). *ECSS-E-ST-10-02C Rev.1 – Verification*. <https://ecss.nl/standard/ecss-e-st-10-02c-rev-1-verification-1-february-2018/>
- Edwards, T. V. (2017). Application of agile methodology to electromechanical design: A case study. *2017 International Annual Conference of the American Society for Engineering Management, ASEM 2017*.
- EnduroSat. (2021). *X-Band Transmitter CubeSat Communication Module*.
<https://www.endurosat.com/cubesat-store/cubesat-communication-modules/x-band-transmitter/>
- Eppinger, S. D., & Browning, T. R. (2018). Process Architecture DSM Models. In *Design Structure Matrix Methods and Applications*. <https://doi.org/10.7551/mitpress/8896.003.0008>

References

- Estefan, J. A. (2008). Survey of model-based systems engineering (MBSE) methodologies. *IncoSE MBSE Focus Group*, 25, 1–70. http://www.omg-sysml.org/MBSE_Methodology_Survey_RevB.pdf
- Falcon, S. (2016). The k-Fibonacci difference sequences. *Chaos, Solitons and Fractals*, 87, 153–157. <https://doi.org/10.1016/j.chaos.2016.03.038>
- Fazzi Bortolini, R., Nogueira Cortimiglia, M., de Moura Ferreira Danilevich, A., & Ghezzi, A. (2018). Management Decision Lean Startup: a comprehensive historical review. *Management Decision*. <https://doi.org/10.1108/MD-05-2017-0477>
- Feldmuller, D. (2018). Usage of agile practices in Mechatronics System Design Potentials, Challenges and Actual Surveys. *Proceedings of the 2018 19th International Conference on Research and Education in Mechatronics, REM 2018*, 30–35. <https://doi.org/10.1109/REM.2018.8421803>
- Fernandes, R., Gouveia, J. B., & Pinho, C. (2012). Product mix strategy and manufacturing flexibility. *Journal of Manufacturing Systems*, 31, 301–311. <https://doi.org/10.1016/j.jmsy.2012.02.001>
- Finlay, P. N., & Wilson, J. M. (1997). Validity of decision support systems: Towards a validation methodology. *Systems Research and Behavioral Science*, 14(3), 169–182. [https://doi.org/10.1002/\(SICI\)1099-1743\(199705/06\)14:3<169::AID-SRES112>3.0.CO;2-G](https://doi.org/10.1002/(SICI)1099-1743(199705/06)14:3<169::AID-SRES112>3.0.CO;2-G)
- Fricke, E., & Schulz, A. P. (2005). Design for Changeability (DfC): Principles To Enable Changes in Systems Throughout Their Entire Lifecycle. *Systems Engineering*, 8(4). <https://doi.org/10.1002/sys.20039>
- Garzaniti, N., Briatore, S., Fortin, C., & Golkar, A. (2019a). Effectiveness of the Scrum Methodology for Agile Development of Space Hardware. *IEEE Aerospace Conference Proceedings, 2019-March*, 1–8. <https://doi.org/10.1109/AERO.2019.8741892>
- Garzaniti, N., Fortin, C., & Golkar, A. (2019b). Toward a Hybrid Agile Product Development Process. *IFIP Advances in Information and Communication Technology*, 565 IFIP, 191–200. https://doi.org/10.1007/978-3-030-42250-9_18

- Garzaniti, N., & Golkar, A. (2020). Performance Assessment of Agile Hardware Co-development Process. *ISSE 2020 - 6th IEEE International Symposium on Systems Engineering, Proceedings*.
<https://doi.org/10.1109/ISSE49799.2020.9272209>
- Garzaniti, N., Golkar, A., & Maggiore, P. (2020). Additive Manufacturing Evaluation Tool for Design Studies. *IEEE Systems Journal*, 14(3), 4382–4393. <https://doi.org/10.1109/JSYST.2019.2939906>
- Garzaniti, N., Tekic, Z., Kukolj, D., & Golkar, A. (2021). Review of technology trends in new space missions using a patent analytics approach. *Progress in Aerospace Sciences*, 125, 100727.
<https://doi.org/10.1016/j.paerosci.2021.100727>
- Gerwin, D., & Barrowman, N. J. (2002). An Evaluation of Research on Integrated Product Development. *Management Science*, 48(7), 938–953. <https://doi.org/10.1287/mnsc.48.7.938.2818>
- Ghezzi, A., & Cavallo, A. (2018). Agile Business Model Innovation in Digital Entrepreneurship: Lean Startup Approaches. *Journal of Business Research*. <https://doi.org/10.1016/j.jbusres.2018.06.013>
- Ghosh, S., & Seering, W. (2014). Set-Based Thinking in the Engineering Design Community and Beyond. *ASME International Design Engineering Technical Conferences*, 1–13.
<https://doi.org/10.1115/DETC2014-35597>
- Git. (2021). *Git*. <https://git-scm.com/>
- Golkar, A., Briatore, S., & Garzaniti, N. (2019). Lessons learnt in the deployment of scrum in space hardware development projects. *Proceedings of the International Astronautical Congress, IAC, 2019-Octob*.
https://doi.org/IAC-19_D1_5_6_x51187
- Gregory, P., Barroca, L., Taylor, K., Salah, D., & Sharp, H. (2015). *Agile Challenges in Practice: A Thematic Analysis* (pp. 64–80). https://doi.org/10.1007/978-3-319-18612-2_6
- Habibi, F., Taghipour Birgani, O., Koppelaar, H., & Radenović, S. (2018). Using fuzzy logic to improve the project time and cost estimation based on Project Evaluation and Review Technique (PERT). *Journal*

References

- of Project Management*, 3, 183–196. <https://doi.org/10.5267/j.jpm.2018.4.002>
- Hajdu, M., & Bokor, O. (2014). The Effects of Different Activity Distributions on Project Duration in PERT Networks. *Procedia - Social and Behavioral Sciences*, 119, 766–775. <https://doi.org/10.1016/j.sbspro.2014.03.086>
- Holtta-Otto, K., & de Weck, O. (2007). Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints. *Concurrent Engineering*, 15(2), 113–126. <https://doi.org/10.1177/1063293X07078931>
- Huang, P. M., Darrin, A. G., & Knuth, A. A. (2012). Agile hardware and software system engineering for innovation. *IEEE Aerospace Conference Proceedings*. <https://doi.org/10.1109/AERO.2012.6187425>
- Iansiti, M. (1995). Shooting the Rapids: Managing Product Development in Turbulent Environments. *California Management Review*, 38(1), 37–58. <https://doi.org/10.2307/41165820>
- INCOSE. (2015). *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* (4th ed.). John Wiley & Sons, Inc.
- Isaksson, O., Eckert, C., Panarotto, M., & Malmqvist, J. (2020). You need to focus to validate. *Proceedings of the Design Society: DESIGN Conference*, 1, 31–40. <https://doi.org/10.1017/DSD.2020.116>
- ISO/IEC JTC 1/SC 7. (2015). *ISO/IEC/IEEE 15288:2015 - Systems and software engineering - System life cycle processes*. <https://www.iso.org/standard/63711.html>
- John of Salisbury. (1159). *Metalogicon, Book III, Chapter 4*.
- Kamal Tipu, S., & Zia, S. (2012). An Effort Estimation Model for Agile Software Development. *Advances in Computer Science and Its Applications (ACSA)*, 314(1), 2166–2924. www.worldsciencepublisher.org
- Karlström, D., & Runeson, P. (2005). Combining Agile methods with stage-gate project management. *IEEE Software*, 22(3), 43–49. <https://doi.org/10.1109/MS.2005.59>

- Karlström, D., & Runeson, P. (2006). Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering*, 11(2), 203–225. <https://doi.org/10.1007/s10664-006-6402-8>
- Khaleeq uz Zaman, U., Rivette, M., Siadat, A., & Meysam Mousavi, S. (2017). Integrated product-process design: Material and manufacturing process selection for additive manufacturing using multi-criteria decision making. *Robotics and Computer Integrated Manufacturing*, 51, 169–180. <https://doi.org/10.1016/j.rcim.2017.12.005>
- KPMG. (2019). *Agile Transformation Survey on Agility: From Agile experiments to operating model transformation: How do you compare to others?* <https://assets.kpmg/content/dam/kpmg/be/pdf/2019/11/agile-transformation.pdf>
- Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. In *Computer* (Vol. 36, Issue 6, pp. 47–56). <https://doi.org/10.1109/MC.2003.1204375>
- Law, A. M. (2014). *Simulation Modeling and Analysis* (5th ed.). McGraw-Hill Education.
- Lawson, M., & Karandikar, H. M. (1994). A Survey of Concurrent Engineering. *Concurrent Engineering: Research and Applications*, 2(1), 1–6. <https://doi.org/10.1177/1063293X9400200101>
- Le Dain, M. A., Blanco, E., & Summers, J. D. (2013). Assessing design research quality: Investigating verification and validation criteria. *Proceedings of the International Conference on Engineering Design, ICED, 2 DS75-02*, 183–192.
- Lenarduzzi, V., & Taibi, D. (2016). MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product. *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, 112–119. <https://doi.org/10.1109/SEAA.2016.56>
- Lincoln, Y. S., & Guba, E. (1958). *Naturalistic Inquiry* (1st ed.). SAGE Publications, Inc.
- Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2016). The use and effectiveness

References

- of user stories in practice. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9619, 205–222.
https://doi.org/10.1007/978-3-319-30282-9_14
- Ma, G., Jia, J., Zhu, T., & Jiang, S. (2019). A critical design structure method for project schedule development under rework risks. *Sustainability (Switzerland)*, 11(24).
<https://doi.org/10.3390/SU11247229>
- Magdaleno, A. M., Werner, C. M. L., & Araujo, R. M. De. (2012). Reconciling software development models: A quasi-systematic review. *Journal of Systems and Software*, 85(2), 351–369.
<https://doi.org/10.1016/J.JSS.2011.08.028>
- Mahmoud-Jouini, S. Ben, Silberzahn, P., & Paris, T. (2017). Resolving the Commitment-Flexibility Dilemma in New Technology Ventures. *International Journal of Innovation Management*, 21(6), 1750047.
<https://doi.org/10.1142/S1363919617500475>
- Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, 85(9), 2086–2095. <https://doi.org/10.1016/j.jss.2012.04.005>
- Maier, A. M., & Störrle, H. (2011). What are the characteristics of engineering design processes? *ICED 11 - 18th International Conference on Engineering Design - Impacting Society Through Engineering Design, 1*, 188–198.
- Maier, J. F., Eckert, C. M., & Clarkson, P. J. (2017). Model granularity in engineering design – concepts and framework. *Design Science*, 3. <https://doi.org/10.1017/DSJ.2016.16>
- Maier, J. F., Eckert, C. M., & Clarkson, P. J. (2015). Different levels of product model granularity in design process simulation. *Proceedings of the International Conference on Engineering Design, ICED*, 3(DS 80-03).
- McCutcheon, D. M., & Meredith, J. R. (1993). Conducting case study research in operations management. *Journal of Operations Management*, 11(3), 239–256. [https://doi.org/10.1016/0272-6963\(93\)90002-7](https://doi.org/10.1016/0272-6963(93)90002-7)

- McGreal, D. (2020). *What is Sprint Planning?* Scrum.Org. <https://www.scrum.org/resources/what-is-sprint-planning>
- Meißner, M., Jacobs, G., Jagla, P., & Sprehe, J. (2021). Model based systems engineering as enabler for rapid engineering change management. *Procedia CIRP*, 100, 61–66. <https://doi.org/10.1016/J.PROCIR.2021.05.010>
- Myers, B. A. (1995). State of the Art in User Interface Software Tools. *Readings in Human–Computer Interaction*, 323–343. <https://doi.org/10.1016/B978-0-08-051574-8.50035-2>
- NASA. (2007). *Systems Engineering Handbook NASA/SP-2007-6105, Revision 1*.
- Nowack, M., Endrikat, J., & Guenther, E. (2011). Review of Delphi-based scenario studies: Quality and design considerations. *Technological Forecasting and Social Change*, 78(9), 1603–1615. <https://doi.org/10.1016/j.techfore.2011.03.006>
- O’Neal, C. (1993). Concurrent Engineering with Early Supplier Involvement: A CrossFunctional Challenge. *International Journal of Purchasing and Materials Management*, 29(1), 2–9. <https://doi.org/10.1111/j.1745-493X.1993.tb00001.x>
- Ovesen, N., & Dowlen, C. (2012). The challenges of becoming agile - Experiences from new product development in industry and design education. *Proceedings of the 14th International Conference on Engineering and Product Design Education: Design Education for Future Wellbeing, EPDE 2012*, 9–14.
- Ozkan, N., Gok, M. S., & Kose, B. O. (2020). Towards a Better Understanding of Agile Mindset by Using Principles of Agile Methods. *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020*, 721–730. <https://doi.org/10.15439/2020F46>
- Papadakis, E., & Tsironis, L. (2018). Hybrid methods and practices associated with agile methods, method tailoring and delivery of projects in a non-software context. *Procedia Computer Science*, 138, 739–746. <https://doi.org/10.1016/j.procs.2018.10.097>

References

- Pimentel, B. A., & de Carvalho, A. C. P. L. F. (2020). A Meta-learning approach for recommending the number of clusters for clustering algorithms. *Knowledge-Based Systems, 195*, 105682. <https://doi.org/10.1016/j.knosys.2020.105682>
- Punkka, T. (2012). Agile Hardware and Co-Design. *Embedded Systems Conference*, 1–8. <http://www.agilemodeling.com/essays/communication.htm>
- Ramos, A. L., Ferreira, J. V., & Barceló, J. (2013). Lithe: An agile methodology for human-centric model-based systems engineering. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 43*(3), 504–521. <https://doi.org/10.1109/TSMCA.2012.2207888>
- Rancic Moogk, D. (2012). Minimum Viable Product and the Importance of Experimentation in Technology Startups. *Technology Innovation Management Review, 2*(3), 23–26. <https://doi.org/10.22215/timreview/535>
- Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How Agile Practices Impact Customer Responsiveness and Development Success: A Field Study. *Project Management Journal, 48*(2), 99–121. <https://doi.org/10.1177/875697281704800208>
- Robey, D., Hellman, K., Monlouis, I., Nations, K., & Johnston, W. J. (2018). Between flexibility and discipline in new product development: expertise as a boundary condition. *Marketing Intelligence & Planning, MIP-02-2015-0042*. <https://doi.org/10.1108/MIP-02-2015-0042>
- Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatara, L. E., Seppänen, P., & Kuvaja, P. (2018). Advances in Using Agile and Lean Processes for Software Development. *Advances in Computers*. <https://doi.org/10.1016/BS.ADCOM.2018.03.014>
- Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2008). Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering, 11*(3), 246–262. <https://doi.org/10.1002/SYS.20098>
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.

-
- Journal of Computational and Applied Mathematics*, 20(C), 53–65. [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7)
- Rowe, G., & Wright, G. (1999). The Delphi technique as a forecasting tool: Issues and analysis. *International Journal of Forecasting*, 15(4), 353–375. [https://doi.org/10.1016/S0169-2070\(99\)00018-7](https://doi.org/10.1016/S0169-2070(99)00018-7)
- Royce, W. W. (1970). Managing the development of large software systems. *Electronics*, 26(August), 1–9. [https://doi.org/10.1016/0378-4754\(91\)90107-E](https://doi.org/10.1016/0378-4754(91)90107-E)
- Saat Network GmbH. (2008). *Quick Poll Results: Sprint Duration*. <https://saat-network.ch/2008/05/quick-poll-results-sprint-duration/>
- Saat Network GmbH. (2011). *What is the optimal Story Size?* <https://saat-network.ch/2011/03/what-is-the-optimal-story-size/>
- Salkind, N. (2012). Encyclopedia of Research Design. *Encyclopedia of Research Design*. <https://doi.org/10.4135/9781412961288>
- Schmidt, T. S., Atzberger, A., Gerling, C., Schrof, J., Weiss, S., & Paetzold, K. (2019). *Agile Development of Physical Products - An Empirical Study about Potentials, Transition and Applicability*. January, 96. <https://athene-forschung.unibw.de/doc/128068/128068.pdf>
- Schmidt, T. S., Chahin, A., Kößler, J., & Paetzold, K. (2017). Agile Development and the Constraints of Physicality: A Network Theory-based Cause-and-Effect Analysis. *21st International Conference on Engineering Design*, 199–208.
- Schmidt, T. S., Wallisch, A., Böhmer, A. I., Paetzold, K., & Lindemann, U. (2018a). Media richness theory in agile development choosing appropriate kinds of prototypes to obtain reliable feedback. *2017 International Conference on Engineering, Technology and Innovation: Engineering, Technology and Innovation Management Beyond 2020: New Challenges, New Approaches, ICE/ITMC 2017 - Proceedings, 2018-Janua*, 521–530. <https://doi.org/10.1109/ICE.2017.8279930>

References

- Schmidt, T. S., Weiss, S., & Paetzold, K. (2018b). *Agile Development of Physical Products: An Empirical Study about Motivations, Potentials and Applicability*.
- Schmidt, T. S., Weiss, S., & Paetzold, K. (2018c). Expected vs. Real effects of agile development of physical products: Apportioning the hype. *Proceedings of International Design Conference, DESIGN*, 5, 2121–2132. <https://doi.org/10.21278/idc.2018.0198>
- Schön, E.-M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards & Interfaces*, 49, 79–91. <https://doi.org/10.1016/j.csi.2016.08.011>
- Schuh, G., Dölle, C., Kantelberg, J., & Menges, A. (2018a). Identification of Agile Mechanisms of Action As Basis for Agile Product Development. *Procedia CIRP*, 70, 19–24. <https://doi.org/10.1016/j.procir.2018.02.007>
- Schuh, G., Dölle, C., & Schloesser, S. (2018b). Agile Prototyping for technical systems Towards an adaption of the Minimum Viable Product principle. *Proceedings of NordDesign: Design in the Era of Digitalization, NordDesign 2018*.
- Schuh, G., Riesener, M., & Breunig, S. (2017). Design for Changeability: Incorporating Change Propagation Analysis in Modular Product Platform Design. *Procedia CIRP*, 61, 63–68. <https://doi.org/10.1016/J.PROCIR.2016.11.238>
- Schwaber, K., & Beedle, M. (2001). Agile Software Development with Scrum. In *Cdswebcernch* (Vol. 18). <https://doi.org/10.1109/2.947100>
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.
- Scrum.org. (2019). *What is a Sprint Backlog?* Scrum Org. <https://www.scrum.org/resources/what-is-a-sprint-backlog>

-
- Sedano, T., Ralph, P., & Peraire, C. (2019). The Product Backlog. *Proceedings - International Conference on Software Engineering, 2019-May*, 200–211. <https://doi.org/10.1109/ICSE.2019.00036>
- Seebacher, G., & Winkler, H. (2014). Evaluating flexibility in discrete manufacturing based on performance and efficiency. *International Journal of Production Economics*, 153, 340–351. <https://doi.org/10.1016/J.IJPE.2014.03.018>
- Seepersad, C. C., Pedersen, K., Emblemståg, J., Bailey, R., Allen, J. K., & Mistree, F. (2006). The Validation Square: How Does One Verify and Validate a Design Method? *Decision Making in Engineering Design*, 303–313. <https://doi.org/10.1115/1.802469.CH25>
- Serrador, P., & Pinto, J. K. (2015). Does Agile work? — A quantitative analysis of agile project success. *International Journal of Project Management*, 33(5), 1040–1051. <https://doi.org/10.1016/J.IJPROMAN.2015.01.006>
- Sethi, R., & Iqbal, Z. (2008). Stage-Gate Controls, Learning Failure, and Adverse Effect on Novel New Products. *Journal of Marketing*, 72(1), 118–134. <https://doi.org/10.1509/jmkg.72.1.118>
- Shannon, W. D. (2007). 11 Cluster Analysis. In *Handbook of Statistics* (Vol. 27, pp. 342–366). Elsevier. [https://doi.org/10.1016/S0169-7161\(07\)27011-7](https://doi.org/10.1016/S0169-7161(07)27011-7)
- Smith, R. P., & Eppinger, S. D. (1997a). Identifying Controlling Features of Engineering Design Iteration. *Management Science*, 43(3), 276–293. <https://doi.org/10.1287/mnsc.43.3.276>
- Smith, R. P., & Eppinger, S. D. (1997b). A predictive model of sequential iteration in engineering design. *Management Science*, 43(8), 1104–1120. <https://doi.org/10.1287/mnsc.43.8.1104>
- Smith, R. P., & Morrow, J. A. (1999). Product development process modeling. *Design Studies*, 20(3), 237–261. [https://doi.org/10.1016/s0142-694x\(98\)00018-0](https://doi.org/10.1016/s0142-694x(98)00018-0)
- Sommer, A. F., Dukovska-Popovska, I., & Steger-Jensen, K. (2014). Barriers towards integrated product development - Challenges from a holistic project management perspective. *International Journal of*

References

- Project Management*, 32(6), 970–982. <https://doi.org/10.1016/j.ijproman.2013.10.013>
- Sommer, A. F., Hedegaard, C., Dukovska-Popovska, I., & Steger-Jensen, K. (2015). Improved Product Development Performance through Agile/Stage-Gate Hybrids: The Next-Generation Stage-Gate Process? *Research-Technology Management*, 58(1), 34–45. <https://doi.org/10.5437/08956308X5801236>
- Sosa, M. E., Eppinger, S. D., & Rowles, C. M. (2003). Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions. *Journal of Mechanical Design*, 125(2), 240–252. <https://doi.org/10.1115/1.1564074>
- Statista. (2021). *Office productivity software global market share 2021*. <https://www.statista.com/statistics/983299/worldwide-market-share-of-office-productivity-software/>
- Stevens, R. (2015). Concurrent engineering methods and models for satellite concept design. *2015 IEEE Aerospace Conference*, 1–15. <https://doi.org/10.1109/AERO.2015.7119270>
- Sugar, C. A., Gareth, & James, M., & James, G. M. (2003). Finding the Number of Clusters in a Dataset. *Journal of the American Statistical Association*, 98, 750–763. <https://doi.org/10.1198/016214503000000666>
- Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business.
- Takeuchi, H., & Nonaka, I. (1986). The New New Product Development Game. *Harvard Business Review*, 64(1), 285–305. <https://hbr.org/1986/01/the-new-new-product-development-game>
- Tang, B. (1993). Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424), 1392–1397. <https://doi.org/10.1080/01621459.1993.10476423>
- Theodoridis, S. (2020). Monte Carlo Methods. In *Machine Learning* (pp. 731–769). Elsevier. <https://doi.org/10.1016/b978-0-12-818803-3.00026-x>
- Thomke, S. H. (1997). The role of flexibility in the development of new products: An empirical study.

-
- Research Policy*, 26(1), 105–119. [https://doi.org/10.1016/S0048-7333\(96\)00918-3](https://doi.org/10.1016/S0048-7333(96)00918-3)
- Toggl. (2021). *Free Time Tracking Software*. <https://toggl.com/>
- Ullah, K., NiaziMahmood, & AhmadRashid. (2011). Barriers in the selection of offshore software development outsourcing vendors: An exploratory study using a systematic literature review. *Information and Software Technology*, 53(7), 693–706. <https://doi.org/10.1016/J.INFSOF.2010.08.003>
- Unger, D. W., & Eppinger, S. D. (2009). Comparing product development processes and managing risk. *International Journal of Product Development*, 8(4), 382–402. <https://doi.org/10.1504/IJPD.2009.025253>
- Ünlü, R., & Xanthopoulos, P. (2019). Estimating the number of clusters in a dataset via consensus clustering. *Expert Systems with Applications*, 125, 33–39. <https://doi.org/10.1016/j.eswa.2019.01.074>
- Upton, D. M. (1995). Flexibility as process mobility: The management of plant capabilities for quick response manufacturing. In *Journal of Operations Management* (Vol. 12).
- Valispace GmbH. (2019). *Valispace*. <https://www.valispace.com/>
- Valle, S., & Vazquez-Bustelo, D. (2009). Concurrent engineering performance: Incremental versus radical innovation. *International Journal of Production Economics*, 119(1), 136–148. <https://doi.org/10.1016/j.ijpe.2009.02.002>
- Wei, X., Prybutok, V., & Sauser, B. (2021). Review of supply chain management within project management. *Project Leadership and Society*, 2, 100013. <https://doi.org/10.1016/J.PLAS.2021.100013>
- Wiest, J. D., & Levy, F. K. (1977). *A management guide to PERT/CPM : with GERT/PDM/DCPM and other networks*. https://books.google.pt/books?id=GAFUAAAAMAAJ&pgis=1&redir_esc=y
- Wynn, D. C., & Clarkson, P. J. (2018). Process models in design and development. *Research in Engineering Design*, 29(2), 161–202. <https://doi.org/10.1007/s00163-017-0262-7>
- Wynn, D. C., & Eckert, C. M. (2017). Perspectives on iteration in design and development. *Research in*

References

- Engineering Design*, 28(2), 153–184. <https://doi.org/10.1007/s00163-016-0226-3>
- Wynn, D. C., Eckert, C. M., & Clarkson, P. J. (2007). Modelling Iteration in Engineering Design. *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07.2007*, 693-694 (exec. Summ.), full paper no. DS42_P_561.
- Yassine, A., Braha, D., & Engineering, C. C. (2003). Complex Concurrent Engineering and the Design Structure Matrix Method. *Concurrent Engineering*, 11(3), 165–176. <https://doi.org/10.1177/106329303034503>
- Yin, R. K. (2013). *Case Study Research: Design and Methods* (5th ed.). SAGE Publications Inc.
- Yingkui Gu, Juanjuan Liu, & Weidong Wu. (2006). Integrated Product and Process Development Mode Based on Models Coupling. *2006 6th World Congress on Intelligent Control and Automation*, 6744–6747. <https://doi.org/10.1109/WCICA.2006.1714389>
- Zhang, H. C., & Daguang Zhang. (1995). Concurrent Engineering: An Overview from Manufacturing Engineering Perspectives. *Concurrent Engineering*, 3(September 1995), 221–236. <https://doi.org/10.1177/1063293X9500300308>
- Zhang, X., Nojima, Y., Ishibuchi, H., Hu, W., & Wang, S. (2020). Prediction by Fuzzy Clustering and KNN on Validation Data With Parallel Ensemble of Interpretable TSK Fuzzy Classifiers. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–15. <https://doi.org/10.1109/TSMC.2020.2999813>
- Zink, L., Hostetter, R., Bohmer, A. I., Lindemann, U., & Knoll, A. (2017). The use of prototypes within agile product development explorative case study of a Makeathon. *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 68–77. <https://doi.org/10.1109/ICE.2017.8279871>