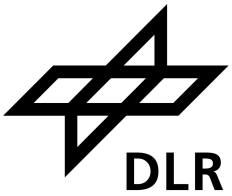

Master Thesis

Minimum Viable Systems Engineering (MVSE) for Rapid Space-System Development

Satya Sree Rupa Rallabandi

6256157

Fachbereich 4

MSc. Space Engineering II

Supervised by:

Dipl.-Ing. Dominik Quantius

Dr.-Ing. Caroline Lange

Bremen, April 2026

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Dipl.-Ing. Dominik Quantius and Dr..-Ing. Caroline Lange, for their invaluable guidance, constructive feedback, and continuous support throughout this thesis. Their expertise and insights, particularly during the literature review and framework development, have been instrumental in shaping this work. I am especially grateful for their patience in reviewing multiple drafts and for pushing me to achieve a higher standard of scientific rigour.

I would also like to extend my sincere thanks to all the systems engineers, technical leads, and programme managers who took the time to participate in the survey. Their candid responses and willingness to share their experiences provided the empirical foundation without which this research would not have been possible. The insights gathered from their daily practice directly informed the design of the MVSE framework.

I am equally grateful to the participants of the SHERLOCK workshop, whose engagement and thoughtful feedback during the hands-on sessions validated the practical applicability of the framework. Their constructive criticism and suggestions for improvement have been invaluable.

I thank the Institute of Space Systems at DLR Bremen and the University of Bremen for providing the resources, facilities, and academic environment that made this research feasible.

On a personal note, I would like to thank my parents for their unwavering support and encouragement throughout my studies. Their belief in me has been a constant source of motivation. I am very grateful to my closest friends Pruthvi Shankar Pulijala, Meghana Kolluru, Ankita TV, Prajwal Gudadoor, Ravindu K, Majella Riona Rajendran, and Niclas Wrede, who supported me, directly or indirectly, in completing the thesis.

Finally, I would like to acknowledge the broader NewSpace community. The willingness of startups, entrepreneurs, and engineers to challenge established practices and pursue ambitious goals is what makes this field so exciting. It is my hope that the MVSE framework contributes, in some small way, to their success.

Satya Sree Rupa Rallabandi
Bremen, April 2026

Abstract

NewSpace organisations increasingly rely on rapid hardware iteration, small engineering teams, and compressed development timelines. In these conditions, classical Systems Engineering frameworks, although rigorous, often become too document-heavy and phase-driven to effectively support fast-moving development. At the same time, modern space projects cannot compromise on verification credibility, traceability, or mission assurance.

This thesis explores how Systems Engineering can be adapted for these environments through the concept of Minimum Viable Systems Engineering (MVSE). MVSE aims to identify the essential set of Systems Engineering (SE) practices and decision mechanisms that preserve system integrity while reducing unnecessary overhead. The focus is on digital integration, iterative verification, proportional tailoring based on risk, and evidence-centred rather than document-centred workflows.

The central hypothesis is that teams with stronger digital continuity, clearer tailoring rules, and more iterative verification practices achieve higher Systems Engineering effectiveness, expressed through faster update cycles, improved consistency, and reduced non-value-added effort. By analysing current NewSpace practices and comparing them with classical SE frameworks, this thesis develops a lightweight, standard-compliant MVSE framework suited to fast-paced space development.

The outcome aims to support organisations seeking to maintain mission assurance while increasing engineering velocity and reducing systems engineering burden.

Contents

1 Introduction 1
1.1 Background and NewSpace Context 1
1.2 Problem Statement 2
1.3 Research Aim and Objectives 2
1.4 Research Questions 3
1.5 Expected Contributions 3
1.6 Scope and Limitations 3
1.6.1 Scope 4
1.6.2 Exclusions 4
1.6.3 Limitations 4
1.6.4 Intended Use 5
1.7 Thesis Outline 6

2 Literature Review. 7
2.1 Established Systems-Engineering Standards 7
2.1.1 INCOSE Systems-Engineering Handbook (v5.0) 7
2.1.1.1 Historical Development and Purpose 7
2.1.1.2 Core Framework Structure 7
2.1.1.3 Key Strengths of the INCOSE Approach 8
2.1.1.4 Limitations for NewSpace Context 9
2.1.2 NASA NPR 7123.1C 10
2.1.2.1 Context and Authority 10
2.1.2.2 Core Requirements 10
2.1.2.3 Key Strengths of the NASA SE Handbook 10
2.1.2.4 Limitations in the NewSpace Context 11
2.1.3 ISO/IEC 15288 12
2.1.4 ECSS-E-ST-10C Rev.1 12
2.1.4.1 Key Characteristics 13
2.1.4.2 Limitations for NewSpace Startups 13
2.1.4.3 Note on Tailoring 15
2.1.5 Common Patterns Across Traditional Standards 15
2.2 Agile and Lean Approaches in Systems Engineering 16
2.2.1 Agile Software Development Principles 16
2.2.1.1 Origins and Evolution 16

2.2.1.2 Core Agile Practices and Methods 16
2.2.1.3 Demonstrated Benefits of Agile 17
2.2.1.4 Limitations of Pure Agile for Hardware Systems 17
2.2.2 Lean Principles and Systems Engineering 18
2.2.2.1 Origins of Lean Thinking 18
2.2.2.2 Seven Principles of Lean 19
2.2.2.3 Lean Thinking Applied to Systems Engineering 19
2.2.3 Agile and Lean together 19
2.2.4 Agile and Lean Insights for MVSE 20
2.3 Model-Based Systems Engineering (MBSE) 21
2.3.1 Core Concepts and Benefits 21
2.3.1.1 Paradigm Shift from Document-Centric to Model-Centric SE 21
2.3.1.2 Systems Modeling Language (SysML) 23
2.3.1.3 MBSE Process and Workflow 24
2.3.1.4 When MBSE Is Most Valuable 25
2.4 NewSpace Practices and Constraints 25
2.4.1 The NewSpace Paradigm Shift 25
2.4.1.1 Definition and Historical Context 25
2.4.1.2 Key Technical Characteristics 26
2.4.1.3 Business Model Innovation 27
2.4.2 Key Challenges in NewSpace 29
2.4.2.1 Multidisciplinary integration complexity 29
2.4.2.2 COTS driven engineering risks 30
2.4.2.3 Testing Infrastructure Limitations 30
2.4.2.4 Underdeveloped Marginal Philosophy 31
2.4.2.5 Rapid Team Scaling and Turnover 32
2.4.2.6 Investor-driven Milestone Compression 33
2.4.3 Comparative Analysis of Traditional SE Frameworks 34
2.4.4 Summary of Identified Pain Points & Thesis Focus 36

3 Research Methodology 37
3.1 Research Philosophy and Design 37
3.2 Methodological Approach 37
3.3 Survey Design 38
3.3.1 Survey Elements from the research questions 38
3.3.2 Survey Structure and Rationale 38
3.3.3 Pilot Testing with AI-Generated Mock Data 39
3.3.3.1 Rationale and Methodology 39

3.3.3.2 Key Insights from the Mock Analysis 40
3.3.3.3 Limitations and Use of Mock Data 41
3.3.3.4 Value Added to the Research Process 41
3.3.4 Survey Instrument 42
3.4 Participant Recruitment and Sampling 42
3.4.1 Sampling Strategy 42
3.4.2 Recruitment Channels 43
3.5 Response and Anonymity 43
3.6 Ethical Procedures and Consent 44
3.6.1 Data Management and Anonymisation (GDPR Compliance) 44
3.7 Data Analysis Methodology 44
3.7.1 Quantitative Data Analysis 44
3.7.1.1 Data Cleaning and Preprocessing 44
3.7.1.2 Descriptive Statistical Methods 45
3.7.1.3 Visualisation Methods 45
3.7.2 Software and Tools 46
3.8 Research Validity, Reliability, and Limitations 46
3.8.1 Internal Validity 46
3.8.2 External Validity (Generalisability) 47
3.8.3 Reliability 47
3.8.4 Limitations 48
3.8.4.1 Self-Reported Data 48
3.8.4.2 Single-Source Bias 48
3.8.4.3 Scope Limitations 48
3.8.5 Mitigation Strategies 48
3.8.6 Ethical Considerations 49
3.8.7 Summary of Methodological Approach 50

4 Empirical Findings: The State of Systems Engineering in NewSpace 51
4.1 Introduction 51
4.2 Respondent Demographics and Organisational Context 52
4.3 Perceived Burden of Traditional SE Processes 55
4.3.1 Non-Value-Added Effort 57
4.4 Tailoring and Agility: The Gap Between Intent & Practice 57
4.4.1 Agility Capabilities 59
4.4.2 Heatmap: SE Agility and Tailoring Capability 60
4.5 Digital Toolchain and Traceability Debt 61
4.5.1 Heatmap: SE Tool Integration and Digital Capability 62
4.6 Verification, Validation, and Review Practices 63

4.6.1 Verification Challenge Severity 63
4.6.2 Verification Review Frequency 65
4.7 What Practitioners Value: MVSE Principles 66
4.7.1 SE Priorities: Ranked Importance for Next Project 67
4.8 Barriers to Adoption 68
4.9 Synthesis: The Mandate for MVSE 69

5 MVSE Framework 71
5.1 Conceptual Definition of MVSE 71
5.2 Core Design Principles of MVSE: 72
5.3 Target Audience and Applicability 72
5.4 Overview of MVSE Components 73
5.4.1 Minimum Viable Requirements 75
5.4.1.1 Origin and Philosophy 75
5.4.1.2 MVR Criteria 75
5.4.1.3 The "Minimum" in MVR 76
5.4.1.4 Guidelines for MVR 76
5.4.1.5 Traditional Requirements vs. MVR 79
5.4.1.6 Working Example: SHERLOCK Mission 79
5.4.2 Architecture Baseline 83
5.4.2.1 Origin and Philosophy 83
5.4.2.2 AB Criteria 83
5.4.2.3 Guidelines for AB 84
5.4.2.4 Traditional Architecture vs. Architecture Baseline 84
5.4.2.5 Working Example: SHERLOCK Mission 85
5.4.3 Dynamic Interface Mapping 89
5.4.3.1 Origin and Philosophy 89
5.4.3.2 DIM Criteria 89
5.4.3.3 Guidelines for DIM 90
5.4.3.4 Traditional Interfacing vs. DIM 90
5.4.3.5 Working Example: SHERLOCK Mission 91
5.4.4 Risk-Based Verification Planning 94
5.4.4.1 Origin and Philosophy 94
5.4.4.2 RVP Criteria 94
5.4.4.3 Guidelines for RVP 95
5.4.4.4 Traditional V&V vs. RVP 96
5.4.4.5 Working Example: SHERLOCK Mission 97
5.4.5 Decision Capture Log 99
5.4.5.1 Origin and Philosophy 99

5.4.5.2 DCL Criteria . . . 100
5.4.5.3 Guidelines for DCL . . . 100
5.4.5.4 Traditional SE Case vs. DCL . . . 101
5.4.5.5 Working Example: SHERLOCK Mission . . . 102
5.4.6 Integration of MVSE . . . 105
5.4.7 Integrating MVSE Components: The Traceability Web . . . 105
5.4.7.1 The Traceability Web Architecture . . . 105
5.4.7.2 Artifact Interconnections . . . 105
5.4.7.3 Traceability Queries in Practice . . . 105
5.4.7.4 Artifact Update Cadence . . . 109
5.4.7.5 Artifact Dependencies and Workflow . . . 109
5.4.7.6 Development Timeline with MVSE Artifacts . . . 109
5.4.7.7 Traceability Dashboard . . . 112
5.4.7.8 Summary of Integration Benefits . . . 112
5.4.8 Other Features of MVSE . . . 113
5.4.8.1 Proportional Review Gates . . . 113
5.4.8.2 Budget Tracking . . . 116
5.4.8.3 Integration with Agile Practices . . . 117
5.4.8.4 Request for Change Process . . . 118
5.4.8.5 Summary of Supporting Features . . . 121
5.4.8.6 Positioning MVSE Against Classical Frameworks . . . 121
5.5 Summary of MVSE Framework . . . 123
5.5.1 MVSE Substitution Mapping: What Replaces What . . . 123

6 Implementation and Future Scope . . . 127
6.1 Pilot Testing: SHERLOCK Mission Workshop . . . 127
6.1.1 Workshop Setup . . . 127
6.1.2 Workshop Execution . . . 128
6.1.3 Notion Workspace and References . . . 129
6.2 Future Scope . . . 129
6.2.1 Potential Use Cases: . . . 129
6.2.2 Longitudinal Validation . . . 130
6.2.3 Tool Development . . . 130
6.2.4 Extension to Larger Programmes . . . 130
6.2.5 Industry Adoption and Standardisation . . . 131
6.2.6 Integration with Emerging Technologies . . . 131

7 Conclusion . . . 133
7.1 Achievement of Research Aim and Objectives . . . 133
7.2 Satisfaction of Research Questions . . . 134

7.2.1 Reflection on Limitations and Intended Use 135
7.2.1.1 Empirical Limitations 135
7.2.1.2 Validation Limitations 135
7.2.1.3 Scope Limitations 135
7.2.1.4 Assumptions and Prerequisites 136
7.2.1.5 Intended Use Revisited 136
7.3 Summary of Contributions 137
7.4 Closing Reflection 138

A Survey Questions 139

B Consent Form 147

C Workshop Data 149
C.1 MVSE Validation Workshop - Invitation 149
C.1.1 Topic 149
C.1.2 Goal 149
C.1.3 Agenda 149
C.2 Workshop Handout 151
C.2.1 Validation of the MVSE Framework 151
C.2.1.1 Introduction 151
C.2.1.2 Purpose 151
C.2.1.3 Problem Statement 151
C.2.1.4 Mission Objectives 152
C.2.1.5 Study Scenario 152
C.2.1.6 Study Objectives 152
C.2.1.7 Participant Role 153
C.2.1.8 Schedule Overview (Reference) 155
C.3 Work Instructions 156
C.3.0.1 Role: Systems Engineer 156
C.3.0.2 Role: GNC, Power, Structures, OBC, Communications, Payload 158
C.3.0.3 Role: Risk Management 159

D Documentation of AI-Based Tools Usage 160

List of Figures

2.1 The Lean Management System [22] 20
2.2 The Lean & Agile Together [22] 21

4.1 Distribution of Organisation Types 52
4.2 Years of Experience in Space-System Development 53
4.3 Primary Systems Engineering Roles 53
4.4 Project Development Cycle Duration 54
4.5 SE Team Size Distribution 54
4.6 Perceived SE Process Bureaucracy 55
4.7 Documentation Value Perception 56
4.8 SE Artefact Consistency 56
4.9 Categorised Non-Value-Added SE Effort 57
4.10 SE Tailoring Approaches 58
4.11 Number of Formal SE Review Gates per Project 58
4.12 Ability to Adjust SE Rigor Quickly 59
4.13 Iterative Development Practices 59
4.14 Systems Engineering Agility and Tailoring Capability (Heatmap) 60
4.15 SE Tool Usage Distribution 61
4.16 Tool Integration Satisfaction 61
4.17 Automated Traceability Capability 62
4.18 Systems Engineering Tool Integration and Digital Capability (Heatmap) . . 63
4.19 Difficulty Balancing Verification Rigor and Agility 64
4.20 SE Process Proportionality to Risk 64
4.21 Average Severity of Verification Challenges 65
4.22 Verification Status Review Frequency 65
4.23 Perceived Value of MVSE Principles 66
4.24 SE Priorities Ranking Distribution (Heatmap) 67
4.25 Perceived Barriers to MVSE Adoption 68

5.1 Sherlock Physical AB 88
5.2 Traceability web architecture 106
5.3 MVSE Process Flow 110
5.4 An example of a MVSE Project Timeline 111

List of Tables

1.1 Thesis Structure Overview 6

2.1 Comparative Analysis of Traditional Systems Engineering Frameworks . . 35
2.2 Identified Pain Points and Planned MVSE responses 36

3.1 Summary of Data Analysis Methodology 50

4.1 Summary of Key Findings and MVSE Response 69

5.1 Overview of MVSE Components and Associated Pain Points 74
5.2 Comparison of Traditional Requirements Engineering and MVR 79
5.3 Minimum Viable Requirements (MVR) for SHERLOCK Mission 82
5.4 Comparison of Traditional Architecture Documentation and AB 85
5.5 Functional-to-Physical Allocation for SHERLOCK Mission 86
5.6 Architecture Baseline Summary for SHERLOCK Mission 87
5.7 Comparison of Traditional Interface Management and DIM 91
5.8 Dynamic Interface Matrix for SHERLOCK Mission 92
5.9 Risk Categories for Verification Planning 95
5.10 Comparison of Traditional Verification Planning and RVP 96
5.11 Risk Based Verification Planning for SHERLOCK Mission 97
5.12 Verification Status Dashboard for SHERLOCK 99
5.13 Comparison of Traditional Decision Documentation and DCL 101
5.14 DCL Light Format Entries for SHERLOCK 102
5.15 DCL Full Format Entry for SHERLOCK 103
5.16 DCL Category Distribution for SHERLOCK 104
5.17 Artifact Interconnections and Traceability Queries 107
5.18 Artifact Update Cadence by Project Phase 109
5.19 Traceability Dashboard Metrics 112
5.20 MVSE Proportional Review Gates 113
5.21 Verification Activities by Project Phase 115
5.22 RFC Example for SHERLOCK Mission 120
5.23 MVSE Supporting Features and Addressed Pain Points 121
5.24 Comparison of Classical SE Frameworks and MVSE 122
5.25 Classical SE Artefacts and Their MVSE Equivalents 124

6.1 Workshop Agenda for SHERLOCK MVSE Pilot 128

C.1 Roles and Responsibilities 154
C.2 Schedule Overview 155

D.1 Documentation of AI-Based Tools Used in This Thesis 161

Abbreviation	Full Form
AB	Architecture Baseline (MVSE Core Component)
ADCS	Attitude Determination and Control System
AI	Artificial Intelligence
AIT/AIV	Assembly, Integration and Testing / Assembly, Integration and Verification
AO-91	Amateur Radio Satellite Organization Mission (reference in MVR)
AR	Acceptance Review (SE Gate)
ASTRA e.V.	Student rocketry association at Bremen
BDD	Block Definition Diagram (SysML)
CAD	Computer-Aided Design
CCB	Configuration Control Board
CDR	Critical Design Review
CEF	Concurrent Engineering Facility
COTS	Commercial Off-The-Shelf
CSTR	Commercial Space Technology Roadmap (MIT)
DARPA	Defense Advanced Research Projects Agency
DCL	Decision Capture Log (MVSE Core Component)
DDF	Design Definition File (ECSS)
DIM	Dynamic Interface Mapping (MVSE Core Component)
DJF	Design Justification File (ECSS)
DLR	German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)
DRD	Document Requirements Definition (ECSS)

Abbreviation	Full Form
ECSS	European Cooperation for Space Standardization
EMC	Electromagnetic Compatibility
ESA	European Space Agency
FAA	Federal Aviation Administration
FCC	Federal Communications Commission
FRR	Flight Readiness Review
GDPR	General Data Protection Regulation
GNC	Guidance, Navigation, and Control
HIL	Hardware-in-the-Loop
IBD	Internal Block Diagram (SysML)
ICD	Interface Control Document
IMU	Inertial Measurement Unit
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
ITAR	International Traffic in Arms Regulations
JWST	James Webb Space Telescope
LEO	Low Earth Orbit
LRR	Launch Readiness Review
MAS	Mission Assurance
MBSE	Model-Based Systems Engineering
MCR	Mission Concept Review
MDR	Mission Definition Review
ML	Machine Learning

Abbreviation	Full Form
MVR	Minimum Viable Requirements (MVSE Core Component)
MVSE	Minimum Viable Systems Engineering
NASA	National Aeronautics and Space Administration
NewSpace	New Commercial Space Industry
NPR	NASA Procedural Requirements
OBC	On-Board Computer
OBDH	On-Board Data Handling
OMG	Object Management Group
ORR	Operational Readiness Review
PCB	Printed Circuit Board
PDR	Preliminary Design Review
PDSA	Plan-Do-Study-Act Cycle
PP (1-8)	Pain Point (1 through 8, from literature synthesis)
QR	Qualification Review
RCA	Root Cause Analysis
RF	Radio Frequency
RFC	Request for Change
RTM	Requirements Traceability Matrix
RVP	Risk-Based Verification Planning (MVSE Core Component)
SE	Systems Engineering
SEP	Systems Engineering Plan
SIL	Software-in-the-Loop
SPI	Serial Peripheral Interface
SRR	System Requirements Review

Abbreviation	Full Form
SVR	System Verification Review
SysML	Systems Modeling Language
TRL	Technology Readiness Level
TRSL	Technology Readiness Status List (ECSS)
UML	Unified Modeling Language
V&V	Verification and Validation
VCD	Verification Control Document (ECSS)
VSE	Very Small Enterprise (ISO/IEC 29110)

Note:

MVSE Core Components: AB, DCL, DIM, MVR, RVP

Classical SE Standards: INCOSE, NASA NPR, ECSS, ISO/IEC 15288

---

1 Introduction

The commercial space industry has undergone a fundamental transformation over the past two decades. What was once the exclusive domain of government space agencies and large aerospace contractors, characterised by extended development timelines, elaborate documentation regimes, and risk-averse decision-making, has given way to a dynamic ecosystem of entrepreneurial ventures, established aerospace corporations pivoting toward rapid innovation, and even academic institutions fielding orbital systems. This transformation, colloquially termed the "movement", is reshaping not only how space missions are conceived and executed, but also the engineering processes and methodologies upon which they depend [1]

1.1 Background and NewSpace Context

The rise of NewSpace has reshaped how space system programmes are formulated, designed and delivered. Start-ups and small-to-medium enterprises now launch vehicles, assemble satellite constellations and operate sub-orbital platforms on compressed schedules (12-24 months) with engineering teams of 20 - 100 engineers. The new paradigm, however, stands as a complete turnaround from the old one, where multi-year programs were necessary, hundreds of engineers worked, and there was a need for extensive SE processes. [2]

Peeters characterises this transformation as a fundamental paradigm shift, noting that NewSpace represents a "bottom-up approach using more simplified designs, leading to several cheaper solutions and thus affordable space projects for a broader range of users," in contrast to traditional space which was "more top-down oriented and based upon government interests" [2, p.2]. Chechile further observes that NewSpace ventures face "numerous difficulties encountered when designing a space system from scratch on limited budgets, non-existing processes, and a great deal of organizational fluidity and emergence" [3].

Classical SE processes were designed for long duration projects with a life span of 5-10 years and hundreds of engineers involved, which also consumed a substantial portion of the engineering efforts on non-value added activities such as manual documentation, duplicate traceability and repetitive verification [4]. While it worked extremely well for the classical space sector, this is unsuitable for the startups as cost, schedule and risk objectives cannot be compromised as every minute spared and every Euro saved is extremely precious [5].

1.2 Problem Statement

For NewSpace companies, a system engineering framework is required that will allow for the following to be achieved at once:

- accelerate time-to-market by matching with commercial launch windows.
- decrease non-core work through the removal of unnecessary paperwork and traceability.
- maintain risk-based hardware practices to ensure mission success.
- fit seamlessly into agile, digital engineering workflows currently being used by cross-functional teams.

Modern frameworks such as INCOSE SE Handbook (v5.0) [6], NASA NPR 7123.1C [7], ISO/IEC 15288 [8], ECSS-E-ST-10C Rev.1 [9] were developed for traditional systems, are constantly upgrading the NewSpace needs, but they might need a few adjustments to better support rapid practices of NewSpace companies. The lack of a minimum viable, risk-driven process will hamper lean engineering teams in reaching their ambitious targets.

1.3 Research Aim and Objectives

The aim of this thesis is to design and assess an approach called Minimum Viable Systems Engineering (MVSE), that maintains technical assurance and traceability while supporting the rapid iteration cycles typical of NewSpace organizations. To achieve this aim, the following five objectives have been established:

1. Evaluate the shortcomings of existing SE approaches, such as INCOSE, NASA, ISO, and ECSS, within high-speed, hardware-oriented projects.
2. Gather empirical data via survey conducted with systems engineers working in the space industry.
3. Determine the systems engineering artefacts that produce the highest value relative to effort.
4. Create an MVSE approach based on Agile, Lean, and MBSE concepts, which is efficient, risk-oriented, and lightweight, taking into consideration the outputs of the survey.
5. Conduct a theoretical application of the framework with SE practitioners with feedback sessions.

1.4 Research Questions

These research questions were formulated before embarking on the thesis, and thus, the items for the survey came out of these questions. The following questions will help in defining the goals of the study.

1. What percentage of time is spent on non-value-added activities?
2. How bureaucratic is the current SE process?
3. How is the traceability handled, and how well are the tools integrated?
4. What are the typical reviews and how are they used to better the project?
5. What kind of Verification and Validation (V&V) management is being performed in the SE process?
6. How is risk management addressed in an organization?
7. How is the tailoring handled in an organization?
8. What are the major bottleneck challenges in the context of Systems Engineering?

1.5 Expected Contributions

The expected contributions of this thesis are as follows:

- Provide empirical insight into how SE is currently practiced in various aerospace startups.
- Determine the challenges faced by systems engineers and the key bottlenecks in the system engineering process.
- Develop an MVSE framework that combines lean practices, risk management, and model-based methods.
- Provide a tool set that small teams can use as a guide in implementing the MVSE Framework
- Outline a plan for the implementation of the MVSE framework.

1.6 Scope and Limitations

To ensure transparency regarding the applicability and generalisability of this research, the following scope boundaries, exclusions, limitations, and intended use conditions are defined.

1.6.1 Scope

This research is specifically focused on NewSpace organisations operating under resource-constrained, fast-paced development conditions. The scope is defined by the following characteristics:

- Focuses on NewSpace startups and small to medium enterprises with 20 to 150 engineers
- Applies to hardware-centric space systems (launch vehicles, satellite constellations, in-space infrastructure)
- Targets uncrewed missions with 12 to 36 month development timelines
- Seeks a lean subset of existing frameworks (INCOSE, NASA, ECSS, ISO), not a full replacement
- Considers software-only systems where they interface with physical hardware

1.6.2 Exclusions

The MVSE framework and the findings of this study are not intended for the following contexts, where additional rigour or different constraints apply:

- Crewed missions or systems involving human safety risk.
- Projects with development timelines longer than 36 months.
- Teams smaller than 20 or larger than 150 engineers.
- Regulatory compliance (ITAR, export controls) – organisations requiring these may need additional documentation.

1.6.3 Limitations

Several methodological and scoping limitations should be acknowledged when interpreting the results of this thesis:

- Empirical data relies on self-reported survey responses, not objective metrics.
- MVSE framework is evaluated through theoretical application and feedback sessions, not longitudinal case studies.
- Assumes basic engineering maturity (version control, configuration management, test discipline) is already in place.

- Does not prescribe specific software tools or platforms; remains tool agnostic.
- Full validation across a complete project lifecycle from concept to launch is beyond the scope of this master thesis.

1.6.4 Intended Use

The MVSE framework is intended to be applied with the following understanding:

- MVSE is to be considered a foundation, not an ultimate prescription.
- Organisations should tailor it further to their specific risk profile, team size, and program constraints.

1.7 Thesis Outline

Table 1.1: Thesis Structure Overview

Chapter	Focus
1	Introduction: context, problem statement, research aim and questions, scope, methodology, ethics, contributions, thesis structure
2	Literature review: foundations of systems engineering, major frameworks (INCOSE, NASA, ISO, ECSS), agile/lean SE, MBSE, NewSpace practices, synthesis into the MVSE concept
3	Research methodology: philosophical stance, survey questions outline, data collection instruments, participant selection/recruitment, ethical procedures, data management, data analysis techniques, Research Validity, Reliability, and Limitations
4	Findings: results from survey, identification of efficiency gaps and pain points, artifact relevance, governance acceptance, and requirements for MVSE Framework
5	MVSE Framework Development: design rationale, core principles, detailed artifact definitions, governance mechanisms, process flow, Application Scenario
6	Discussion and Validation: comparative evaluation against traditional SE frameworks, expert feedback, implications for startups, standards bodies, and policy
7	Conclusions and Future Work: summary of contributions, answers to research questions, theoretical and practical implications, recommendations for longitudinal studies and broader industry adoption, future work

2 Literature Review

This chapter establishes the theoretical foundation for the MVSE framework by critically reviewing established Systems Engineering standards (INCOSE, NASA, ECSS, ISO), agile and lean methodologies, Model-Based Systems Engineering (MBSE), and the distinct practices and constraints of NewSpace development. This synthesis identifies eight core pain points in current practice, which collectively motivate the need for a minimum viable approach tailored to fast-paced, resource-constrained space projects.

2.1 Established Systems-Engineering Standards

2.1.1 INCOSE Systems-Engineering Handbook (v5.0)

2.1.1.1 Historical Development and Purpose

The INCOSE Systems Engineering Handbook is one of the key frameworks adopted in the field of systems engineering. Founded in 1990, INCOSE is an International Council on Systems Engineering that emerged due to the complexity of the systems in areas such as aerospace, defense, and telecommunication. The early members of INCOSE noted that failures in big systems were primarily linked to inadequate management of the requirements, interfaces, and verification. This handbook was first created in 1992 and has since undergone several revisions in five editions to reflect best practices from top aerospace and defense contractors. The v5.0 release (2023) represents the latest evolution, incorporating advances in digital engineering, Artificial Intelligence/ Machine Learning (AI/ML) integration, and agile practices. [6]

2.1.1.2 Core Framework Structure

The INCOSE v5.0 framework is built around a comprehensive lifecycle approach. As documented in the handbook, Systems Engineering is defined as "an interdisciplinary approach and means to enable the realization of successful systems" [6].

The framework emphasizes the following
Key processes:

- Stakeholder needs analysis and requirements definition
- System architecture and design
- Implementation and integration

- Verification and validation
- Configuration management and change control

The handbook identifies four process groups for the system lifecycle:

- **Agreement Processes** (Acquisition and Supply)
- **Organizational Project-Enabling Processes** (Life Cycle Model Management, Infrastructure Management, Portfolio Management, Human Resource Management, Quality Management, Knowledge Management)
- **Technical Management Processes** (Project Planning, Project Assessment and Control, Decision Management, Risk Management, Configuration Management, Information Management, Measurement, Quality Assurance)
- **Technical Processes** (Business/Mission Analysis, Stakeholder Needs and Requirements Definition, System Requirements Analysis, Architectural Design, Implementation, Integration, Verification, Validation, Transition, Operations, Maintenance, Disposal) [6]

2.1.1.3 Key Strengths of the INCOSE Approach

INCOSE v5.0 strengths include:

- **Comprehensive lifecycle coverage:** Addresses all phases from concept through retirement, enabling organizations to plan long-term support and sustainment strategies.[6]
- **Proven in complex domains:** INCOSE processes have guided development of critical systems including the Space Shuttle, International Space Station, Air Traffic Management systems, and military defence platforms.[6]
- **Traceability discipline:** The handbook emphasizes bidirectional traceability—from stakeholder needs through requirements, design, implementation, verification, and validation. As noted: "Each requirement should be traced back to a parent/source requirement or expectation in a base-lined document".[6]
- **Formal governance structures:** Structured review gates (e.g., SRR, PDR, CDR, SVR presented as aerospace/defence examples) provide explicit decision points[6].
- **Risk-aware processes:** Recent versions of INCOSE integrate risk management as a continuous activity, not an afterthought.[6]

2.1.1.4 Limitations for NewSpace Context

Despite its comprehensive nature, full implementation of INCOSE v5.0 creates significant challenges in fast-paced, small-team environments:

- **Process Scale Mismatch:** The INCOSE Handbook describes dozens of processes and hundreds of activities across four process groups (Agreement, Organizational Project-Enabling, Technical Management, and Technical Processes) [6, pp. 40-158 [6]]. While the handbook acknowledges that "on smaller projects... SE activities can be conducted very informally" [6, p. xix [6]], it does not provide a pre-configured lightweight profile. Teams must perform tailoring themselves, which consumes resources that could otherwise be spent on engineering work.

- **Tailoring as Active Effort:** The handbook explicitly addresses tailoring in Section 4.1 [6, pp. 215-219 [6]], stating that "tailoring scales the rigor of application to an appropriate level based on risk" [6, p. 215 [6]]. However, the handbook also notes that "tailoring is a process" requiring organizations to "identify and record the circumstances that influence tailoring" and "make tailoring decisions" [6, pp. 216-217 [6]]. For a 20-person NewSpace startup, this tailoring effort itself can be prohibitive.

- **Review Gate Infrastructure:** The handbook lists representative technical reviews including SRR, PDR, CDR, TRR, and PRR. While stating that reviews should be "risk-driven or event-driven, not schedule-driven" [6, p. 33 [6]], the described practices (entry/exit criteria, SME involvement, action tracking, dry-runs) assume infrastructure and personnel availability that small NewSpace teams lack.

- **Document-Centric Legacy:** Although the handbook supports Model-Based Systems Engineering in Section 4.2.1 [6, pp. 219-221 [6]], the process descriptions throughout Section 2.3 remain heavily artifact-referenced (e.g., "SEMP," "project status report," "verification report"). The handbook notes outputs are "often captured in 'documents' or 'artifacts' or 'models'" [6, p. 40 [6]], but this flexibility becomes ambiguity for teams without established digital engineering practices.

- **Change Control Formality:** The Configuration Management process described in Section 2.3.4.5 [6, pp. 87-90 [6]] includes formal baselines, change control boards, and configuration audits. For hardware-intensive NewSpace projects with weekly design iterations, this formality creates friction.

These limitations do not imply that INCOSE is unsuitable for all space projects. For large programmes the rigor of INCOSE is appropriate. However, for NewSpace startups with small teams and compressed timelines, the process and documentation requirements of full INCOSE implementation create overhead disproportionate to project scale.

2.1.2 NASA NPR 7123.1C

2.1.2.1 Context and Authority

NASA Procedural Requirements NPR 7123.1C represents the procedural directive that guides the process of systems engineering within all NASA civil space projects. It was issued in the year 2007 and revised thereafter. The NASA System Engineering Handbook (2007) provides detailed information about these procedures.

2.1.2.2 Core Requirements

NPR 7123.1C establishes technical and programmatic requirements for systems engineering:

Technical Requirements:

- **Requirements traceability:** Every requirement must be traceable to a source (stakeholder need, derived requirement, design constraint, interface requirement). "Each requirement should be traced back to a parent/source requirement or expectation in a baselined document or identified as self-derived" [7].
- **Design reviews:** Mandatory formal design reviews at key milestones including System Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), and System Verification Review (SVR) [7].
- **Verification planning:** "Verification planning begins early in the project life cycle during the requirements development phase" [7].
- **Interface control:** "All interfaces between subsystems must be formally defined, baselined, and controlled"[7].
- **Configuration management:** All hardware, software, and document baselines must be formally established and controlled[7].

2.1.2.3 Key Strengths of the NASA SE Handbook

- **Proven for mission-critical systems:** NPR 7123.1C has guided development of systems that must work flawlessly: the Mars Rovers, James Webb Space Telescope, International Space Station modules, and others[7].
- **Clear verification discipline:** The requirement for upfront verification planning and formal verification reviews ensures that testing is systematic and planned[7].
- **Independent safety oversight:** The involvement of NASA Safety and Mission Assurance provides an independent check on risk management[7].

- **Long-term auditability:** The requirement for detailed documentation and audit trails means that design rationale and verification evidence remain available decades after a mission[7].

2.1.2.4 Limitations in the NewSpace Context

While NASA NPR 7123.1C and the accompanying handbook are essential for human spaceflight and flagship science missions, several characteristics create challenges when applied to small, fast-paced NewSpace projects. The following observations are based on an analysis of the framework's structure and assumptions; they are not presented as explicit statements from NASA documents unless otherwise cited.

- **Prescriptive overhead.** NPR 7123.1C is highly prescriptive. In traditional NASA practice, formal design reviews involve extensive documentation packages and multi-day proceedings with large review teams. For NewSpace startup with a small team and compressed timeline, adapting this level of formality would create disproportionate overhead relative to project scale.

- **Late discovery of design flaws.** The phase-gate model, which the handbook describes in detail for Phases A through D [7, pp.17-42], can incentivise deferring testing and integration until late in the programme. The handbook itself notes that "late identification of and fixes to problems cost considerably more later in the life cycle" [7, p.12]. For fast-paced projects, this creates pressure to discover issues earlier than the NASA model typically allows.

- **Requirement specification burden.** The handbook assumes detailed, upfront requirements. Appendix C provides a comprehensive checklist for writing good requirements, and Appendix D presents a detailed Requirements Verification Matrix [7, pp.197-202]. For a system with 1,000 requirements, maintaining full traceability manually becomes labour-intensive.

- **Risk aversion versus innovation.** The formal review structure, including the extensive entrance and success criteria for each review [7, see Tables G-1 through G-18], can create a culture where teams default to proven designs rather than exploring innovations. For NewSpace startups seeking competitive advantage through novel approaches, this cultural bias is particularly problematic.

- **Proportionality requires deliberate tailoring effort.** The handbook states that formality and documentation depth "are varied as appropriate for the type, size, and complexity of the project" [7, p.12]. Section 3.11 provides a tailoring framework, including a project type classification from Type A to Type F [7, pp.34-41]. However, applying this framework to a NewSpace startup with a small team and compressed timeline re-

quires active tailoring effort. The handbook does not offer a pre-configured lightweight profile; teams must perform the tailoring themselves, which consumes resources that could otherwise be spent on engineering work.

2.1.3 ISO/IEC 15288

ISO/IEC 15288 is an international standard published by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). According to the standard document [8], it applies to "one-of-a-kind systems, mass-produced systems, and customised, adaptable systems."

Key process areas covered:

- **Agreement processes:** Acquisition and supply agreements
- **Organizational project-enabling processes:** Project planning, risk management, configuration management
- **Technical processes:** Stakeholder needs and requirements definition, system requirements analysis, architectural design, implementation, verification, validation [8]

Strength:

- **International applicability:** Recognized globally, enabling organizations to use a common framework across international teams and suppliers [8].
- **Process-oriented:** The standard focuses on processes rather than artifacts, giving organizations flexibility in how they implement [8].
- **Supplier management:** Places strong emphasis on supplier agreements and oversight [8].

Limitations:

- **Heavyweight implementation:** Full implementation requires comprehensive process infrastructure [8].
- **Documentation burden:** Conformance typically requires extensive documentation of processes, procedures, quality records, and audits [8].

2.1.4 ECSS-E-ST-10C Rev.1

The European Cooperation for Space Standardization (ECSS) publishes standards covering all aspects of space system development. ECSS-E-ST-10C Rev.1 [10] is the primary systems engineering standard used by the European Space Agency (ESA) and European con-

tractors. The standard specifies system engineering implementation requirements for space systems and space products development, with specific objectives including establishing a firm technical basis, minimizing technical risk, specifying essential tasks and their outputs, and implementing the customer-system-supplier model [10, Section 1, p.8].

2.1.4.1 Key Characteristics

Based on the standard document, the following characteristics are evident:

- **European regulatory alignment:** The standard is mandatory for ESA projects and widely adopted across the European space industry [10, Foreword, p.2].
- **Interdependent standard set:** The standard is one of a series of ECSS Standards intended to be applied together for management, engineering and product assurance in space projects [10, Foreword, p.2]. The standard notes that it is designed to work in conjunction with other ECSS standards, including ECSS-E-ST-10-02 (Verification), ECSS-E-ST-10-24 (Interface control), ECSS-M-ST-10 (Project planning), ECSS-M-ST-40 (Configuration management), and ECSS-Q-ST-10 (Product assurance) [10, Section 2, p.10].
- **Documentation model:** The standard follows a documentation-based approach where formal documents serve as the primary artefacts of the engineering process. The standard defines numerous Document Requirements Definitions (DRDs) including the System Engineering Plan (SEP, Annex D), Design Definition File (DDF, Annex G), Design Justification File (DJF, Annex K), Requirements Traceability Matrix (RTM, Annex N), and Verification Control Document (VCD, referenced in Section 5.5) [10, Annexes A-S, pp.52-115].
- **Tailoring provision:** The standard acknowledges that tailoring is necessary. It states that it "may be tailored for the specific characteristic and constraints of a space project in conformance with ECSS-S-ST-00" [10, Section 1, p.9]. The standard also provides a pre-tailoring matrix in Section 7 [10, Section 7, pp.38-51] that defines applicability of requirements per space product type.

2.1.4.2 Limitations for NewSpace Startups

- **Designed for complex systems with multi-layer supplier chains.**

  The standard's system engineering process is described as being applied "by each system engineering function of each supplier of the elements of the product decomposition" across "a multi-layered set of suppliers" [10, Section 4.2, p.19]. This assumption of hierarchical supplier chains does not align with vertically integrated NewSpace startups where a single team develops most subsystems in-house.

- **Interdependent standards require full framework adoption.**

  The standard explicitly notes that it is "one of the series of ECSS Standards intended to be applied together" [10, Foreword, p.2]. Section 2 lists over 10 interdependent standards that are normative references, including verification (ECSS-E-ST-10-02), interface control (ECSS-E-ST-10-24), project management (ECSS-M-ST-10), configuration management (ECSS-M-ST-40), and product assurance (ECSS-Q-ST-10) [10, Section 2, p.10]. For a 20-person startup, adopting the full framework creates disproportionate overhead.

- **Extensive documentation requirements.**

  The standard defines numerous mandatory documents. Annex A (informative) lists over 20 deliverable documents including Mission Description Document, Technical Requirements Specification, System Engineering Plan, Technology Plan, Design Definition File, Design Justification File, Requirements Traceability Matrix, Trade-off Reports, and Verification Control Documents [10, Annex A, Table A-1, pp.53-55]. Each of these has detailed content requirements defined in separate annexes (Annexes B through P) [10, pp.56-110]. For a fast-paced startup, producing and maintaining this documentation creates significant overhead.

- **Technical budget and margin policy requirements.**

  The standard requires the system engineering function to "define, control and maintain all technical budgets of the system" and "define and apply the margin policy agreed between customer and supplier" [[10], Section 5.4.1.2, p.31]. While this level of formal margin tracking assumes dedicated resources many startups lack, managing technical budgets and margins should be the highest priority for any technology startup to avoid cost overruns, technical debt, and integration failures.

- **Heavy verification documentation.**

  The standard requires a Verification Plan (ECSS-E-ST-10-02 Annex A), Verification Control Document (ECSS-E-ST-10-02 Annex B), Test Specifications (ECSS-E-ST-10-03 Annex B), Test Procedures (ECSS-E-ST-10-03 Annex C), and Test Reports (ECSS-E-ST-10-02 Annex C) [10, Annex A, Table A-1, pp.53-55]. For a startup with compressed timelines, this verification documentation burden can exceed the effort of the verification activities themselves.

- **Technology Readiness Level (TRL) tracking requirements.**

  The standard requires identification and assessment of technologies in terms of TRL levels, documentation in a Technology Plan (Annex E) and Technology Matrix (An-

nex F), and maintenance of a Technology Readiness Status List (TRSL) [10, Section 5.6.7, pp.35-36]. While TRL assessment is valuable, even a simple matrix requires disciplined tracking that many startups struggle to maintain.

2.1.4.3 Note on Tailoring

The standard does provide a tailoring framework. Section 7 presents a pre-tailoring matrix per space product type [10, Section 7, pp.38-51], and ECSS-S-ST-00-02C provides a 7-step tailoring process [11]. This framework builds on ESA's mission classification system (Category 1 high-risk strategic missions to Category 3 low-risk demonstrators), with risk-adjusted tailoring efforts such as the Fast-Track approach reducing documentation for smaller projects. However, the tailoring process itself requires effort. As ECSS-S-ST-00-02C acknowledges, tailoring activities "have demonstrated to require a non-negligible effort" [11, Section 4.1, p.10]. For a NewSpace startup with limited personnel, this tailoring effort can be prohibitive.

These limitations do not imply that ECSS is inherently unsuitable for all space projects. For ESA flagship missions with large teams and long timelines, the rigor of ECSS is appropriate. However, for NewSpace startups with small teams, compressed timelines, and flat organisational structures, the documentation and process requirements of full ECSS compliance create overhead disproportionate to project scale. This observation motivates MVSE's proportional governance principle, which provides lightweight, risk-based processes designed specifically for NewSpace contexts.

2.1.5 Common Patterns Across Traditional Standards

Analyzing INCOSE, NASA NPR, ISO/IEC 15288, and ECSS reveals common underlying patterns.[6][7][8][10].

- **Phase-gate lifecycle model:** All standards assume a structured lifecycle with formal gates between phases.
- **Requirements-centric approach:** All standards treat requirements as the foundation of system design.
- **Document-centric artifact management:** All standards reflect their origins in an era of paper documents.
- **Formal review gates:** Progress is controlled by formal reviews.
- **Comprehensive traceability:** All standards emphasize bidirectional traceability throughout the lifecycle.

- **Formalized risk management:** Risk management is a structured process with formal assessments and tracking.
- **Configuration management and change control:** Once baselines are established, changes are formal affairs subject to change control boards.

2.2 Agile and Lean Approaches in Systems Engineering

2.2.1 Agile Software Development Principles

2.2.1.1 Origins and Evolution

Agile methodologies in companies can be traced back at least as far as the 1930s when Bell Labs applied plan-do-study-act (PDSA) cycles to the improvement of products and processes. This, of course, predated electronic computers and their software. In the 1960s, Lockheed Martin built its "skunk works," in which small development teams were removed from the normal working environment and freed from managerial constraints, making them autonomous and empowered [12].

The Agile Manifesto, published in 2001 by software practitioners [13], articulated a new philosophy for software development. The manifesto prioritizes [13]:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

These values reflected lack of satisfaction with traditional waterfall approaches. Agile methods have revolutionized software development, delivering higher success rates (64% vs. 49% for waterfall) and faster time-to-market (40–60% faster) according to the Standish Group CHAOS Report (2015) [14].

2.2.1.2 Core Agile Practices and Methods

Scrum is the most widely adopted agile framework [13]. Key practices include [15]:

- **Sprints (timeboxed iterations):** Fixed-duration iterations, typically 1–4 weeks. Each sprint results in a working increment.
- **Daily standups:** Brief (15-minute) daily synchronization meetings.
- **Product backlog:** Prioritized list of features, improvements, and fixes.

- **Sprint review and retrospective:** Review completed work with stakeholders; reflect on process improvements.

2.2.1.3 Demonstrated Benefits of Agile

Empirical evidence supports agile effectiveness [13][15]:

- Agile projects showed higher success rates and faster time-to-market [15].
- Organizations adopting agile practices showed 25–30% improvement in development velocity [15].
- Agile teams using continuous integration catch integration bugs early and frequently [6].

2.2.1.4 Limitations of Pure Agile for Hardware Systems

Pure agile methods have significant limitations when applied to hardware-centric systems [16], [17]:

1. **Catastrophic failure intolerance:** A software bug in a web application can be discovered in production and patched within hours. A spacecraft attitude control system failure is permanent and catastrophic. In hardware development, design changes discovered late have an "orders of magnitude" higher cost to fix compared to those found during the design phase [18]. Unlike software patches, hardware fixes require physical rework, requalification, and often complete redesign cycles.
2. **Long manufacturing lead times:** Software can be refactored and redeployed in hours. Hardware components have long procurement lead times; an electronic component ordered today arrives in 3 to 6 months. Hajou and Voropaeva note that hardware agility must accommodate "long supplier lead times" and "mass-production commitments" as fundamental constraints [16]. Garzaniti and Golkar similarly account for "procurement" as a key lifecycle process in their space hardware development model [17].
3. **Verification complexity and long test cycles:** Software testing can be automated and executed thousands of times per day. Spacecraft verification is complex and time-consuming; environmental testing takes weeks or months. Grimm and Hendrikse document that traditional space hardware verification processes assume a C/D-phase of 4 to 5 years, which is incompatible with compressed development timelines [19].
4. **Regulatory and contractual requirements:** Space systems face strict regulatory oversight. Regulatory approval of design changes can take weeks or months. Macedo and colleagues note that integrating Agile methods into safety-critical aerospace development presents "substantial challenges due to its strict compliance requirements,

such as those outlined in the DO-178C standard," which emphasize "documentation, traceability of requirements across different levels and disciplines, and comprehensive verification and validation (V&V) activities" [20].

5. **System interdependencies and global constraints:** In spacecraft, changes to one subsystem often cascade throughout the design. A power budget change affects every subsystem that consumes power. Better Devices highlights that "a change in a single PCB component or a minor tweak to a firmware driver can cascade into a complete system-level failure" [18]. Garzaniti and Golkar explicitly model these "product interdependencies between components, subsystems and relevant lifecycle processes such as assembly, integration, testing, procurement, and validation" in their analysis of Agile hardware co-development [17].

However, exceptions exist. The MASCOT asteroid lander demonstrated that a hybrid verification approach combining conventional and tailored model strategies can succeed under extreme time pressure, completing system-level AIV in 2.5 years [21]. This case study provides evidence that tailored, agile-informed methods can work in hardware-centric space projects when applied with appropriate risk management.

2.2.2 Lean Principles and Systems Engineering

2.2.2.1 Origins of Lean Thinking

Starting in the 1940s with its roots in the Toyota Production System, lean management has spread from manufacturing to service operations and just about every other department and function at companies, governments, and non-governmental institutions around the world. Lean organizations seek to identify and eliminate activity that is not valued by the customer or end user. This systematic analysis of processes and value streams to reduce waste, variability, and inflexibility boosts performance in cost control, product quality, customer satisfaction, and employee engagement , often simultaneously. [22]

Toyota defined seven categories of waste[23]:

1. **Transportation:** Moving materials unnecessarily.
2. **Inventory:** Excess stock that ties up capital.
3. **Motion:** Unnecessary movement by workers.
4. **Waiting:** Idle time.
5. **Overprocessing:** Unnecessary processing steps.
6. **Defects:** Poor quality requiring rework.

7. **Underutilization of talent:** Not leveraging worker knowledge and creativity.

2.2.2.2 Seven Principles of Lean

1. **Eliminate waste:** Identify and remove non-value-added activities [23].
2. **Amplify learning:** Create feedback loops that enable rapid learning [23].
3. **Decide late:** Defer irreversible decisions until the last responsible moment [23].
4. **Deliver fast:** Minimize lead time from concept to delivery [23].
5. **Empower the team:** Front-line employee should make decisions without excessive approval chains [23].
6. **Build integrity in:** Quality and safety are built into the process and product [23].
7. **Respect the process:** Continuous improvement of processes and methods [23].

2.2.2.3 Lean Thinking Applied to Systems Engineering

When lean principles are applied to systems engineering, they suggest significant opportunities to reduce overhead [23]:

- **Eliminate redundant documentation:** Lean asks which documents are truly necessary for decision-making [23]. Traditional SE produces extensive specifications; lean SE produces only essential artifacts [23].
- **Compress review cycles:** Replace monthly formal reviews with weekly 1-hour design reviews [23].
- **Automate traceability:** Use tools to automatically link requirements to design decisions, implementation, and verification [23].
- **Continuous verification:** Integrate verification throughout development, not deferred to a dedicated test phase [23].
- **Knowledge capture:** Capture design decisions in a lightweight, accessible format [23].

2.2.3 Agile and Lean together

There exists a common myth that lean and agile cannot co-exist, due to their being based on completely different concepts and being applicable to very different sorts of tasks. The theory behind this myth states that lean management can be used for everyday repetitive activities, while agile can be used only for projects or for some kind of creativity[22].

The problem with the described assumption is that the misunderstanding of both the lean management and the agile philosophy lies within its basis. Both systems have proven themselves effective in many cases, and there are actually similar basic values behind both approaches: to create value for the client, to learn better ways of working to improve the performance, to be aligned and clear about the strategy and goals to provide people with a sense of purpose, and to allow them to achieve their full potential as shown in 2.1.[22]

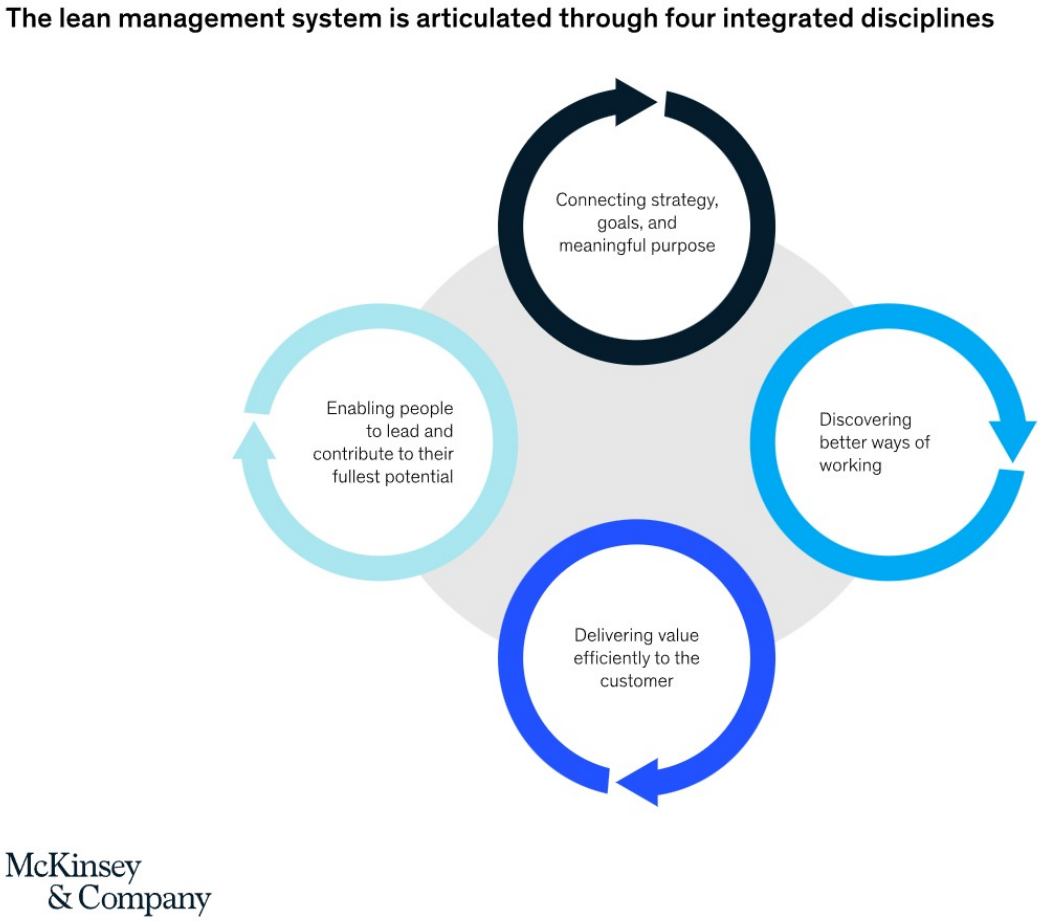

Figure 2.1: The Lean Management System [22]

The systems of lean management and agile have team configurations, processes, and toolkits that can be implemented depending on which approach is deemed best for a particular business organization, which can be seen in the figure 2.2. The common basis between the two systems makes their elements highly compatible. Furthermore, operational success cannot usually be achieved solely by using one system because success usually comes from applying both systems along with their toolkits.

2.2.4 Agile and Lean Insights for MVSE

The key insights from agile and lean thinking that inform MVSE are:

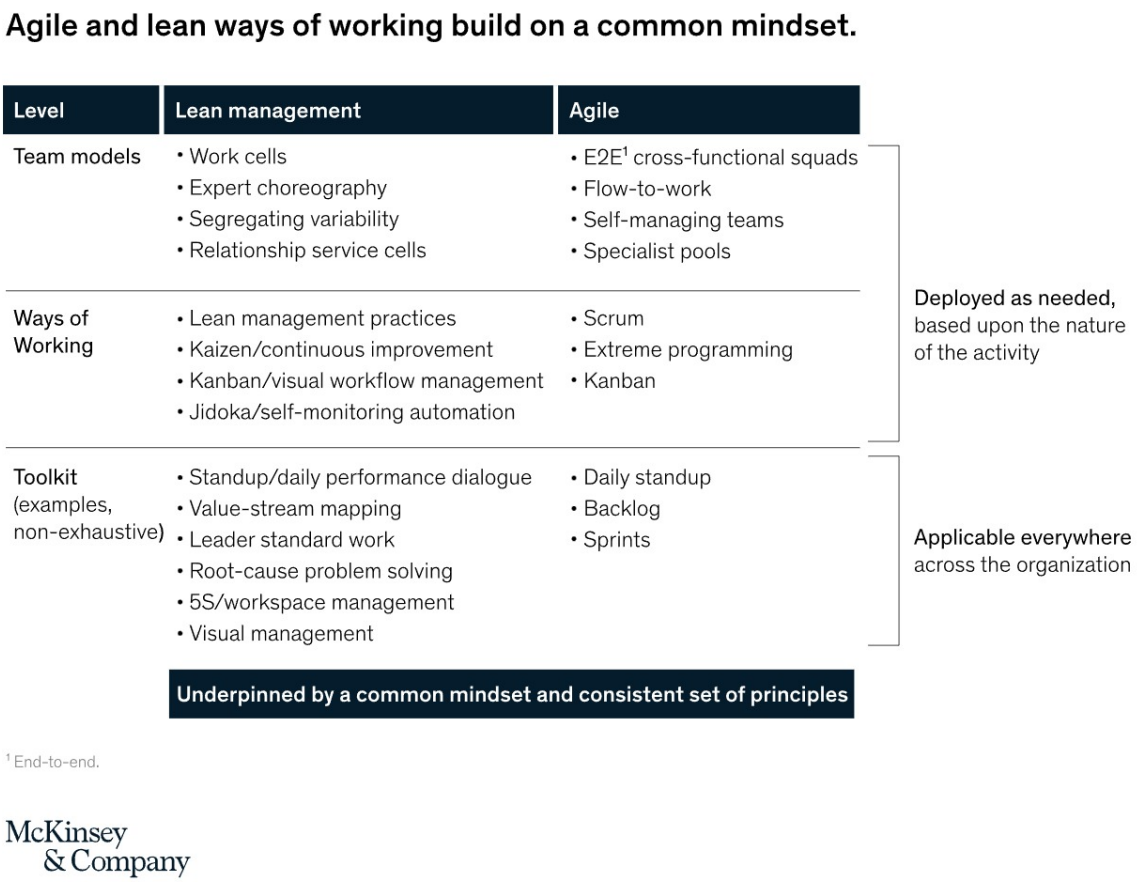

Figure 2.2: The Lean & Agile Together [22]

- **Waste exists in traditional SE**: 20–30% of effort is spent on non-value-added activities[24].
- **Small, empowered teams are effective**: Agile's emphasis on small, cross-functional teams has proven effective[12].
- **Rapid feedback is powerful**: Agile's short feedback cycles reduce risk [23].
- **Documentation should be minimal but sufficient**: Not all documentation is waste. Critical requirements and design decisions should be documented. But verbose specifications and low-value traceability are waste [23].

2.3 Model-Based Systems Engineering (MBSE)

2.3.1 Core Concepts and Benefits

2.3.1.1 Paradigm Shift from Document-Centric to Model-Centric SE

Traditional Document-Centric Approach:

The traditional systems engineering approach has relied on documentation as the primary method for capturing and communicating technical knowledge. For a standard development process for space systems, this includes complete specifications, design descriptions, test procedures, interface control documents, and configuration management records [6, see Chapter 4].

Problems with Document-Centric Approach:

- **Consistency and synchronization:** When a requirement changes, propagation to design documents, interface specifications, and traceability matrices is manual and error-prone [6, see Section 4.2].
- **Traceability burden:** Maintaining bidirectional traceability requires extensive, manual matrix maintenance. The INCOSE Handbook notes that each requirement "should be traced back to a parent/source requirement" [6, p.87], but in document-centric environments this becomes a labour-intensive manual process.
- **Latent contradictions:** A document may be internally consistent, but contradictions with other documents may not be detected until late in design [6, see Section 4.3].
- **Knowledge fragmentation:** Design rationale and context are scattered across multiple documents, making it difficult to reconstruct why decisions were made [6].
- **Limited automated analysis:** Documents are primarily text and figures; automated tools cannot easily analyse or simulate the design [6].
- **Slow feedback:** Changes go through formal review and approval processes, which can take weeks [6, see Section 6.5 on Configuration Management].

Model-Centric Approach:

MBSE shifts the primary artifact from documents to the integrated system model. The system model contains information regarding the requirements, architecture, behaviour, constraints, and verification approach in a semantic environment that enables querying and analysis through automated checking of cross-references [6, see Chapter 3].

Benefits of Model-Centric Approach:

- **Single source of truth:** All system information is captured in one model. Changes are immediately visible to all stakeholders [6].
- **Automated consistency checking:** Modelling tools can detect contradictions automatically, reducing the risk of latent inconsistencies [6].
- **Impact analysis:** Changing a requirement automatically highlights affected design

elements, interface definitions, and verification activities [6].

- **Executable and simulatable:** The model can be executed or simulated, enabling virtual testing before physical prototypes are built [6].
- **Knowledge capture:** The model preserves design intent, not just final specifications, including the rationale behind architectural decisions [25].
- **Reduced documentation effort:** Artifacts such as specifications and interface documents can be generated from the model rather than manually written and maintained [6].

2.3.1.2 Systems Modeling Language (SysML)

Definition of SysML

SysML v1 (Systems Modeling Language) is a standardised, domain-specific modelling language developed by the Object Management Group (OMG). SysML is an extension of the Unified Modeling Language (UML), adapted specifically for systems engineering applications [25].

SysML version 2 (v2) is the next-generation systems modeling language, providing significant enhancements over SysML version 1 in terms of its precision, expressiveness, usability, interoperability, and extensibility.

It is used to specify requirements, behavior, structure, analysis, and verification while maintaining consistency and traceability across all aspects of the system model. SysML v2 is based on the KerML metamodel, which is grounded in formal semantics. It includes textual syntax in addition to graphical syntax, that allows systems modeling to better leverage automation, emerging technologies such as AI, and modern development practices.[26]

Core SysML Constructs:

- **Requirement Diagram:** Represents requirements and their relationships, including derivation, satisfaction, and verification relationships [25, Chapter 12].
- **Block Definition Diagram (BDD):** Represents system architecture and structural decomposition, showing system hierarchies and component relationships [25, Chapter 7].
- **Internal Block Diagram (IBD):** Represents internal structure and interconnections between parts of a system, including interfaces and flows [25, Chapter 7].
- **Use Case Diagram:** Represents system interactions with external actors (users, external systems) and captures functional requirements [25, Chapter 6].

- **Activity Diagram:** Represents flow of activities and control, including data flow and decision logic [25, Chapter 9].
- **State Machine Diagram:** Represents state transitions and responses to events, capturing system behaviour over time [25, Chapter 10].
- **Parametric Diagram:** Represents quantitative relationships and constraints, used for engineering analysis such as performance, mass, and power calculations [25, Chapter 14].
- **Sequence Diagram:** Represents interactions between subsystems over time, showing message ordering and timing constraints [25, Chapter 11].

2.3.1.3 MBSE Process and Workflow

Typical MBSE Development Workflow:

- **Elicitation and Specification Phase:** Stakeholder needs are captured and translated into system requirements, organised hierarchically with acceptance criteria [6, see Chapter 4].
- **Architecture Definition Phase:** System architecture is designed and captured in block definition and internal block diagrams, showing functional and structural decomposition [6, see Chapter 4].
- **Detailed Design Phase:** Detailed behaviour is modelled using activity diagrams, state machines, and parametric diagrams [6, see Chapter 4].
- **Verification and Validation Phase:** Verification methods are specified and results recorded in the model, enabling traceability from requirements to test evidence [6, see Chapter 5].
- **Configuration and Change Management:** The model is version-controlled; changes are reviewed and approved through established change control processes [6, see Section 6.5].

Key Difference - MBSE vs Document-Centric Approach:

In document-centric SE, the workflow is fundamentally linear: requirements specification leads to design specification, which leads to verification planning, then design, then verification, then documentation review, and finally baseline. Each step produces documents that must be manually kept consistent with previous steps.

In MBSE, the workflow is iterative: develop model, simulate and analyse the model, refine the model based on analysis results, extract documents from the model, and baseline the

model. The model is the source of truth; documents are generated artifacts from the model rather than standalone documents that must be maintained separately [6].

2.3.1.4 When MBSE Is Most Valuable

MBSE provides the greatest value in the following contexts [6], [25]:

- **Large, complex systems:** Systems with 500 or more requirements and 10 or more subsystems with extensive interdependencies benefit from the automated consistency checking and impact analysis that MBSE provides.
- **Long-duration programmes:** Multi-year development programmes with multiple phases benefit from the model's ability to preserve design intent and rationale across personnel changes.
- **Large teams:** Teams of 50 or more engineers working in parallel benefit from the model's single source of truth and automated change propagation.
- **Complex interfaces and dependencies:** Systems with many subsystem-to-subsystem interactions benefit from MBSE's interface modelling and consistency checking capabilities.
- **Safety-critical systems:** Systems where failure is catastrophic benefit from the rigorous traceability from requirements through design to verification that MBSE enables.
- **Regulated systems:** Systems subject to regulatory oversight benefit from the audit trail and evidence package that can be generated from the model.

Friedenthal et al. acknowledge that there is a "learning curve for SysML and MBSE" and that organisations should consider the investment required for training and tool adoption [25, see Section 3.5]. For small teams with limited resources, this learning curve can be particularly challenging. MVSE's Architecture Baseline (AB) addresses this by providing a lightweight, model-centric approach that captures only essential architectural information, reducing the formalism required while maintaining traceability.

2.4 NewSpace Practices and Constraints

2.4.1 The NewSpace Paradigm Shift

2.4.1.1 Definition and Historical Context

NewSpace refers to the emergence, beginning approximately 2010, of private commercial enterprises that have fundamentally disrupted the traditional aerospace business model and

supply chain. Pioneered by companies such as SpaceX, Blue Origin, Axiom Space, Relativity Space, and dozens of smaller ventures, NewSpace represents a paradigm shift in how space systems are conceptualized, funded, developed, and operated [27].

The traditional aerospace model, established in the 1960s and dominant through the 2010s, was characterized by [28]:

- Government-led programmes
- Large defence contractors as prime integrators
- Long development timelines (10–15 years)
- Large, distributed teams
- Cost-plus contracting
- Document-centric SE processes
- Risk aversion

NewSpace disrupted this model through [28]:

- Venture capital and commercial funding
- Vertical integration
- Speed-to-market focus
- Reusable designs
- Commercial viability

2.4.1.2 Key Technical Characteristics

NewSpace enterprises exhibit distinctive technical practices, as documented in the MIT-led Commercial Space Technology Roadmap [29]:

- **Vertical Integration and In-House Development:** SpaceX manufactures engines, structures, avionics, and software in-house rather than contracting with suppliers. The CSTR report notes that the U.S. launch industry has seen "significant increase in revenue" since the announcement of NASA's Commercial Orbital Transportation Services (COTS) program in 2006, with new entrants like SpaceX driving market transformation [29, pp.12-15]. This vertical integration enables rapid iteration, as design decisions can be implemented and tested within weeks.
- **Reusable Vehicle Architecture:** SpaceX's Falcon 9 first stage is designed to land and be reused, dramatically reducing per-flight cost. The CSTR report documents that

prior to the COTS program, launch vehicles cost on average $13,600 per kg to LEO, whereas recent COTS vehicles have reduced costs to approximately $5,600 per kg [29, p.12]. Reusability enables economies of scale: as flight rate increases, per-flight cost decreases. The report's analysis of interdependencies highlights that improvements in launch services "cost, performance, and safety" propagate benefits to 18 downstream sub-sectors of the space economy [29, p.16].

- **Rapid Iteration and Flight Testing:** NewSpace companies conduct early, limited flights (often uncrewed) to gather data. Early flights are expected to provide learning; failure is treated as a learning opportunity, not a programme-ending setback. The CSTR report notes that "new entrants have emerged in recent years" and that the U.S. launch industry has shown a "recent general trend of resurgence in number of launches" since the COTS program [29, p.12]. SpaceX conducted four Falcon 1 orbital attempts between 2006 and 2008, with the first three failing; rapid iteration enabled SpaceX to move past early technical hurdles within two years.

- **Small Core Teams:** SpaceX's core engineering team for rocket development is estimated at 100 to 300 engineers. The CSTR report identifies that the commercial space economy includes "a large, growing, and active commercial sector" where companies such as SpaceX, Blue Origin, Rocket Lab, and Virgin Orbit compete with traditional players like ULA [29, p.12]. Small teams communicate rapidly, make decisions quickly, and maintain a clear view of subsystem integration, enabling the rapid development cycles documented in the report [29, pp.12-16].

- **Technology Adoption:** NewSpace teams readily adopt new technologies (additive manufacturing, AI/ML for autonomy, advanced materials) that traditional aerospace avoided. The CSTR report includes additive manufacturing as a key technology area under "Advanced Manufacturing Technology" (TA 9.2.3), with applications including "3D printed rocket engine components" (TA 2.1.2) [29, p.84]. The report notes that companies like Relativity Space are leveraging additive manufacturing to reduce part counts and manufacturing lead times, with "3D printed rocket launch" identified as a harbinger of a new era in aerospace manufacturing [29, p.84].

2.4.1.3 Business Model Innovation

The NewSpace paradigm has introduced fundamentally new business models that differ from traditional government-funded, cost-plus contracting. Three dominant models have emerged:

1. **Commercial Satellite Constellations:** NewSpace companies design small satellites that can be mass-produced and deployed in constellations, reducing per-unit cost and enabling global coverage. Examples include SpaceX's Starlink (over 8,000 satellites

deployed), OneWeb (648 satellites), and Amazon's Project Kuiper (3,236 satellites planned) [2]. These constellations represent a shift from single, high-value satellites to distributed, proliferated architectures that prioritise rapid replenishment and continuous service provision [2, see Figure 1].

2. **Launch-as-a-Service (LaaS):** SpaceX pioneered commercial launch services, selling rides on Falcon 9 to a diverse customer base including commercial satellite operators, government agencies, and other NewSpace ventures. LaaS drives flight rate, creates recurring revenue, and funds development of next-generation vehicles [30]. SpaceX's vertical integration strategy manufacturing engines, structures, avionics, and satellites in-house enables cost control and rapid iteration, with estimated internal marginal launch costs of approximately $15 million per Falcon 9 mission [2], [30]. This represents an order-of-magnitude reduction compared to traditional launch providers [2].

3. **In-Space Infrastructure:** Axiom Space and other companies are developing commercial space stations and in-space refueling services. Axiom is building a commercial module for the International Space Station as a precursor to a standalone commercial space station, planned to support research, manufacturing, and tourism in low Earth orbit [31]. Other companies, such as Relativity Space, are exploring in-space manufacturing and reusable orbital platforms [32]. These initiatives aim to establish a sustainable commercial ecosystem in low Earth orbit and beyond, with applications ranging from materials processing to satellite servicing and orbital data centres [31].

2.4.2 Key Challenges in NewSpace

Space startups operate at the intersection of advanced aerospace innovations and the current commercial needs. The need to deliver high-performance, reliable hardware on rapid development timelines introduces significant technical, organisational, and process challenges. These are not hypothetical, but instead consistently observed across the industry through case studies, FAA publications, and retrospective analyses of commercial failures and delays.

2.4.2.1 Multidisciplinary integration complexity

Launch vehicle systems represent the most complex and sensitive engineering challenges in the aerospace domain. The focus of the space startups is to coordinate the behaviour and interfaces of highly interdependent subsystems such as propulsion, structures, avionics, thermal management, payload, and ground systems.

Each of these subsystems operates independently, with its own assumptions, understandings, tools, test phases, and timelines. In an ideal systems engineering workflow, these subsystems are brought into alignment through an architectural framework that aligns and manages the interdependencies, such as:

- **Physical interfaces** - mass, mechanical mountings, thermal exchanges
- **Electrical interfaces** - power budgets, communication buses, grounding, harness
- **Logical interfaces** - timing, software-hardware feedback, systematic control

However, many startups in this phase focus on hardware iteration over documentation and a bureaucratic approach, resulting in often poorly defined and inadequately tested system interface contexts, or leaving them with implicit knowledge of interdependencies. That leads to:

- **Cross-system regressions** - changes in avionics, power distributions
- **Late-stage integration failure** - GNC algorithms relying on timings not matched by hardware
- **Mismatch in data expectations** - Propulsion & telemetry data logging at different frequencies

One notable example is Firefly's Alpha rocket, particularly during Firefly Alpha Flight 7 (2025), when a first-stage booster exploded on the test stand. The company traced this failure to a "process error" during integration. This highlights how even when subsystems are designed and tested correctly in isolation, failures in integration processes can lead to complete system failure [33].A similar integration-related issue was observed during the launch

of the Phoenix rocket, developed by ASTRA e.V. Bremen, a student rocketry association [34]. The Phoenix project, a hybrid rocket designed for 9 km altitude, faced significant challenges when shipping delays prevented the delivery of essential components such as fins and ground systems, requiring the team to construct ad-hoc solutions to proceed with the launch [34].[1]

While these methods remain valuable, in fast-moving startup environments, they can become bottlenecks, mainly when hardware iterations occur weekly and test feedback drives frequent design changes. Maintaining formal ICDs under such conditions can consume unreasonable engineering effort and lag behind reality.

2.4.2.2 COTS driven engineering risks

To shorten timelines and budget restrictions, many launch vehicle startups rely on Commercial Off-The-Shelf (COTS) components, particularly in avionics, telemetry, computing, and power systems. While this practice supports rapid prototyping, it introduces critical systems engineering risks when these components are not qualified for the extreme environments of launch and spaceflight [35].

- **Environmental Qualification Gaps:** COTS parts are typically designed for terrestrial use and often lack tolerance for high dynamic loads, thermal cycling, vacuum conditions, and radiation. As Herbert et al. note, using products "outside of their rated limits relative to radiation, thermal, shock, vibration, acoustic, and electromagnetic environments can lead to failure" [35, p.2]. Without component-level environmental qualification or mitigation, their use can lead to latent failures in flight.
- **Limited Observability and Support:** COTS vendors may offer minimal documentation or support for integration. Herbert et al. highlight that for COTS parts, "the manufacturer solely establishes and controls specifications for configuration, performance, quality, and reliability... with no Government-imposed requirements" [35, p.5]. Proprietary protocols and black-box firmware make fault isolation difficult, particularly during post-test root cause analysis (RCA).

2.4.2.3 Testing Infrastructure Limitations

Validation through testing is a crucial aspect of aerospace, but for low-volume startups, access to suitable test infrastructure is often a significant challenge. Unlike established aerospace programmes with existing facilities, early-stage companies must either build test infrastructure from scratch or rely on limited commercial or government options, both of which create bottlenecks in the verification and validation (V&V) process [36].

[1] The author is a member of ASTRA e.V. Bremen.

- **Propulsion and Structural Testing Bottlenecks:** Full-scale engine tests, stage separation trials, and structural vibration qualification require large, high-thrust-capable stands and vibration tables. Start-ups like Relativity Space and Firefly Aerospace have publicly invested in private test facilities to overcome this obstacle. Others lacking such infrastructure often delay or compress test plans, sometimes conducting only partial-system or simulated tests before integration.
- **Simulation-Driven Substitution:** Instead of physical testing, startups often rely on digital simulations that may not be calibrated against actual test data. Research on ground-based testing facilities has highlighted inherent "limitations of the testing facility" when emulating space environments, including challenges with scaling, time-delays, and maintaining system stability [37]. These limitations can lead to overconfidence in models and a lack of preparedness for surprises during the integration stage.

2.4.2.4 Underdeveloped Marginal Philosophy

A repeated challenge in startups is the lack of a coherent, system-level margin philosophy. Margins, whether structural, power, mass, or computational, serve as vital buffers that accommodate uncertainty, manufacturing tolerances, and unexpected behaviour. In traditional aerospace programmes, these are meticulously calculated, formulated, and budgeted. In fast-paced startups, however, margin management tends to be informal or nominal, resulting in critical integration issues and failures.

- **Implicit Margins** Subsystems often define their own internal margins independently (e.g., propulsion assumes 10% thermal and avionics another 15%), which can lead to unintentional overdesign or worse, constraint violations when combined. Without a central margin model, the system-level resource envelope remains unclear.
- **Margin Erosion During Iteration** In rapid development cycles, design iterations often consume margins without formal review. Harris notes that when uncertainty data is lacking, "the industry is driven to apply conservative margins or iterate as the design becomes more defined, which can result in costly redesigns and schedule slips" [38]. For instance, added avionics mass or thermal output may surpass structural or cooling capabilities, but this usually goes unnoticed until late-stage testing or integration. This issue is worsened when verification occurs in isolated subsystems rather than within a holistic, integrated system model, highlighting the need for quantitative margin allocation methods [38].
- **Lack of Visibility Across Teams** Startups often lack a centralized system to track budgets in real-time. Instead, mass, power, or thermal data are often scattered across spreadsheets or engineering notes, limiting shared understanding and hindering system-level trades. Research on complex systems design emphasizes the importance of in-

teractive margin visualization and centralized margin management to support dynamic design exploration and informed decision-making [39].

2.4.2.5 Rapid Team Scaling and Turnover

During the initial phases of launch vehicle startups, it is typical for individual engineers to oversee entire subsystems or even multiple subsystems, often without formal interface ownership or documentation. This streamlined staffing model facilitates rapid progress, but it compromises the robustness of systems integration, particularly as the organisation begins to grow.

Paliwoda notes that space startups operate under uniquely volatile, uncertain, complex, and ambiguous conditions, and that "traditional management tools were not designed for rapidly evolving, growing high-tech sectors" [40, p.27]. Furthermore, "traditional hierarchical management approaches are misaligned with the fast-paced, informal structures of startups" [40, p.28].

- **Role Overlap and Tribal Knowledge:** A propulsion engineer may also handle structural mounting, thermal shielding, and interface with avionics, creating deep local context but poor visibility for others. Research on knowledge management in high-velocity environments highlights how such tacit, localized knowledge becomes difficult to transfer across teams and is particularly vulnerable to loss when key personnel depart [41]. As these engineers shift roles or depart, undocumented assumptions, interface behaviours, or test setups are lost, leading to integration mismatches or repeated design errors.
- **Onboarding Bottlenecks:** When teams grow rapidly (e.g., post-Series A hiring spikes), new engineers do not have access to consistent architecture documentation, test histories, or decision rationales. Research on large-scale onboarding in complex engineering environments shows that onboarding is "lengthy, frustrating and expensive," and that "newcomers face personal, interpersonal, process and technical barriers during their incorporation, which in turn affects the overall productivity of the whole team" [42]. Without shared systems knowledge, teams default to point-to-point communication, which creates inefficiencies and leads to rework.
- **Architecture Drift Due to Team Silos:** As subsystems are handed off to new owners or split into specialist teams, the original system architecture can diverge from current implementations. Tiwana and Safadi define this phenomenon as "architectural drift," where the congruence between team structure and system architecture erodes over time as "siloed team and architecture studies obscure their dynamic interplay" [43]. Their research shows that restoring alignment requires continuous "syncing of drifting technical and team architectures" [43].

2.4.2.6 Investor-driven Milestone Compression

In launch vehicle startups, external funding timelines often dictate internal engineering milestones. Chechile notes that NewSpace startups face unique difficulties, including "limited budgets, non-existing processes, and a great deal of organizational fluidity and emergence" [3]. To secure investment tranches or demonstrate market traction, teams are pressured to meet visible, hardware-driven targets such as engine hotfires, stage integrations, or orbital attempts.

- **Prioritisation of Demonstration Over Design Maturity:** Startups may bypass key SE activities, such as interface verification, margin closure, or V&V planning, to meet deadlines for launch vehicle static fire or flight announcements. The Engine's Tough Tech framework notes that in early-stage startups, "investors and funders will want you to hit specific technical milestones before you unlock the next chunk of funding in order to reduce risk" [44]. This milestone-driven funding model incentivises declaring progress toward visible hardware targets, even when underlying subsystems lack architectural stability. Subsystems are frozen before they reach architectural maturity, and integration is rushed, increasing the likelihood of latent system-level faults.

- **Reduced Attention to Non-Visible Engineering Work:** Activities such as interface documentation, risk tracking, architectural updates, and system-level requirements refinement are often deprioritized because they don't produce immediate, fundable "milestones" for investors. The Engine's framework for tough tech startups acknowledges this reality, advising founders to "ruthlessly prioritize what must be solved now vs. what can come later" [44]. While necessary for resource-constrained startups, this prioritisation pushes non-visible engineering work into the backlog, accumulating invisible technical debt that undermines future scalability and reusability.

- **Premature System Commitments:** Milestone pressure often forces teams to commit to full-stack integration or public launch dates before sufficient test coverage is complete. Research on large engineering projects shows that "when clients make commitments early on in conditions of high uncertainty, they increase the likelihood (upside risk) of speeding up delivery if external events do not materialize; however, if these events do materialize, they increase the likelihood (downside risk) of causing design rework and losing process predictability" [45]. This can result in failure-prone launches, as seen in early-stage flight attempts from Astra. During the DARPA Launch Challenge, Astra scrubbed its launch due to "an issue with the rocket's guidance, navigation, and control system" [46]. A subsequent NASA TROPICS mission failed when the second stage engine shut down prematurely, leaving the payload well short of orbital velocity [47]. In both cases, technical readiness lagged behind schedule commitments.

2.4.3 Comparative Analysis of Traditional SE Frameworks

The comparison between the four main systems engineering approaches is shown in Table 2.1. It becomes apparent that all four SE models described share a common trait: whereas INCOSE, NASA, ECSS and ISO present highly effective approaches to the development of complex systems in space projects of large time spans, they bring substantial overhead and rigid requirements into the picture, which clearly do not fit the fast-paced nature of the NewSpace approach. The above-mentioned traits of each SE model are outlined in the table 2.1.

Table 2.1: Comparative Analysis of Traditional Systems Engineering Frameworks

Framework	Core Focus	Key Strengths	Key Limitations (for NewSpace)
INCOSE v5.0	Comprehensive lifecycle processes, technical management, traceability	- Proven on complex systems (ISS, Shuttle) - Bidirectional traceability discipline - Formal governance (SRR/PDR/CDR) - Risk-aware processes	- Dozens of processes - Assumes mature, stable requirements - Review gate structured for larger teams - No pre-configured lightweight profile small projects
NASA NPR 7123.1C	Procedural requirements for civil space projects, verification planning	- Mission-critical pedigree (Mars rovers, JWST) - Clear verification discipline - Independent safety oversight - Long-term auditability	- Highly prescriptive structure - Late discovery of design flaws - Large traceability matrices - Risk-averse culture stifles innovation
ISO/IEC 15288	International standard, process-oriented, supplier management	- Globally recognized across industries - Process-oriented (not artifact-prescriptive) - Strong supplier agreement framework	- Heavyweight infrastructure required - Extensive documentation for conformance - Audit and quality record burden
ECSS-E-ST-10C	European space standard, safety integration, detailed procedures	- European agency pedigree (ESA) - Step-by-step templates - Safety integrated throughout	- Very prescriptive, stifles innovation - Substantial process infrastructure - Deeply document-centric
Common Patterns	Phase-gate lifecycle, requirements-centric, formal reviews, traceability	- Works for large, long-duration programs - High assurance for flagship missions	- Mismatch with startup timelines - Assumes large teams - Change control too slow for iteration

2.4.4 Summary of Identified Pain Points & Thesis Focus

The literature review has covered four related fields: existing systems engineering standards, agile/lean approaches, model-based systems engineering, and new space engineering with its limitations. These have led to eight key pain points that drive the need for a minimum viable systems engineering approach, which is what the current thesis seeks to create. Table 2.2 below highlights the pain points and their dedicated attention in upcoming chapters.

Table 2.2: Identified Pain Points and Planned MVSE responses

ID	**Pain Point**	**MVSE Response**
PP1	Excessive documentation overhead	Eliminate waste; minimal essential artifacts
PP2	Phase-gate rigidity incompatible with iteration	Replace with event-driven, lightweight reviews
PP3	Upfront requirements assumption fails under uncertainty	Support emergent design with lean change control
PP4	Late verification makes failures expensive	Continuous, incremental V&V ("test early, often")
PP5	Informal margin management leads to erosion	Simple, shared real-time margin model
PP6	Knowledge loss from tribal knowledge & team scaling	Lightweight decision records & interface capture
PP7	Investor milestones distort SE priorities	Risk-driven tailoring, not milestone-driven
PP8	Tool fragmentation breaks digital continuity	Pragmatic path to MBSE for small teams

3 Research Methodology

3.1 Research Philosophy and Design

The research is grounded in the pragmatist philosophy of knowledge [48]. According to this epistemology, the practical benefits from the use of knowledge should be taken into account, and pluralism in methodology is accepted whenever it helps to solve real-life problems. The purpose of this study is not to generate universal laws concerning SE. Instead, the intention of this research is to create a practice-sensitive artifact: the Minimum Viable SE framework that can be utilized in the dynamic conditions of the NewSpace era. Hence, the exploratory-qualitative methodology has been used for this project [49].

3.2 Methodological Approach

The methodological foundation of this research is a survey-based case study, where the "case" is the contemporary European NewSpace sector. This approach, as articulated by Yin [50], is particularly well-suited for investigating a contemporary phenomenon within its real-world context, especially when the boundaries between the phenomenon and its context are not clearly evident. The phenomenon under investigation is the set of systems engineering (SE) practices, while the context is the unique economic, cultural, and technological environment of NewSpace organizations.

A mixed-methods questionnaire was selected as the primary data collection instrument. This choice is rooted in the principle of complementarity, where the integration of quantitative and qualitative data provides a more comprehensive and nuanced understanding than either method could achieve alone [51]. This pragmatic integration aligns directly with the research philosophy, employing the most effective tools to understand and address the research problem.

- **Diagnostic Mapping:** This segment of the survey (Likert scale questions and multiple-choice questions) is specifically tailored to provide statistical data regarding the state of current industry processes and process effectiveness. This allows for the measurement of important variables, such as the percentage of engineers' time spent on non-value-added documentation processes or the use of integrated digital toolchains. This data thus provides evidence from a macro perspective to empirically support the problem statement posed by this thesis.

- **Qualitative Framework Seeding:** The qualitative segment of the survey (open-ended questions) is designed to gather latent requirements, contextual limitations, and experiences that need to be considered during the design of an SE solution. While the empirical data will be useful in indicating a process's bureaucratic nature, qualitative data explains why a process is bureaucratic in nature, providing detailed information about problems associated with the process. Such insightful details represent important baseline requirements for the development of an SE framework, which will help ensure that MVSE's requirements address practical issues experienced by users.

3.3 Survey Design

The design of the survey instrument was a critical, multi-stage process intended to maximize data quality, validity, and relevance. The architecture of the questionnaire was derived from a set of high-level "inspirational questions" that operationalize the core research objectives

3.3.1 Survey Elements from the research questions

Inspirational questions outlined in Section 1.4 - Research questions (see 1.4) served as the guiding theory behind this survey design process. These questions outline the broad areas of investigation that guided the formulation of concrete and measurable survey items.

An example of how research questions led to specific survey items can be illustrated with the question "How is the traceability handled and How well are the tools integrated?". The specific survey items that were generated from this question include the following multiple-choice question, asking which SE tools are used by the respondent; the Likert-scale question measuring the level of integration; and the open-ended question asking about difficulties with achieving data continuity.

3.3.2 Survey Structure and Rationale

The final questionnaire was partitioned into six logically sequenced sections , each designed to investigate a specific dimension of the research problem. The rationale for each section is as follows:

1. **Organizational Context:** This section collected essential firmographic data not merely for demographic description, but to enable subsequent correlational analysis. It allows for the investigation of hypotheses such as whether perceived process burden correlates with organizational size or project duration.

2. **SE Process Burden & Mis-alignment:** This section was designed to empirically quantify the core problem statement. By gathering data on non-value-added effort

and perceived bureaucracy, it provides the quantitative evidence needed to justify the call for a more lightweight framework.

3. **Agility, Risk-Driven Tailoring & Flexibility:** This section addresses a central tenet of MVSE. It probes the gap between the recognized need for process tailoring and the existence of formal, systematic methods for achieving it, thereby identifying a key area for intervention.

4. **Digital Maturity & MBSE Adoption:** This section assesses the technological context in which any proposed framework must operate. Understanding the current state of toolchains and MBSE adoption is critical for ensuring the MVSE framework is technologically feasible and culturally resonant.

5. **Verification, Traceability & Data Integrity:** This section focuses on critical SE functions that are perennial sources of project risk and effort. The data gathered here informs the design of MVSE's proportional verification and model-centric traceability principles.

6. **Good-Enough SE Conceptualisation:** This final, more exploratory section functions as a form of concept validation. It presents core MVSE ideas in a neutral manner to gauge their intuitive appeal and perceived utility among the target user base, providing an early indication of the framework's potential for adoption.

In order to prove its validity, the questionnaire was tested during the pilot testing stage, first with AI and then which included three systems engineers with senior positions who were not part of the study population. The testing process helped in eliminating vague terms such as the difference between "agile development" and "process agility", item sequencing logic, and improving the questionnaire to ensure that it takes no more than 15 to 20 minutes to complete.

3.3.3 Pilot Testing with AI-Generated Mock Data

Prior to full deployment of the survey to industry practitioners, a pilot test was conducted using AI-generated responses to evaluate the survey instrument's effectiveness, identify potential ambiguities, and confirm that the collected data would address the research questions defined in Section 1.4. This approach was inspired by the supervisor's suggestion to validate the survey design before human subject recruitment.

3.3.3.1 Rationale and Methodology

The primary objectives of the AI-driven pilot test were threefold:

1. **Instrument validation:** To identify ambiguous wording, unclear response scales, or logical sequencing issues that might confuse human respondents.

2. **Data quality assessment:** To verify that the survey questions would yield meaningful, interpretable data aligned with the eight research questions.

3. **Expectation calibration:** To establish baseline expectations for result patterns, enabling comparison between AI-generated trends and actual industry responses.

Two mock datasets were generated using a large language model (GPT-4), simulating responses from approximately 50 practitioners across three organisational archetypes: launch-vehicle startups, satellite/payload startups, and established NewSpace companies. The AI was prompted to simulate realistic responses based on:

- The survey structure and question logic defined in Appendix A
- Known industry pain points documented in the literature (Chapter 2)
- Plausible variation across different team sizes, project durations, and SE standards adoption

3.3.3.2 Key Insights from the Mock Analysis

The AI-generated mock data revealed several important patterns that informed adjustments to the survey instrument and provided early validation of the MVSE framework's direction.

- **Process burden quantification:** The mock data indicated that respondents would report mean process heaviness scores between 2.1 (startups) and 2.8 (established firms) on a 5-point scale (where 1 represents "Strongly agree that process is too heavy"). Non-value-added time estimates clustered around 24-28% for startups and 35-42% for established firms. These ranges aligned with literature estimates [23] and confirmed that the survey's burden-related questions would capture meaningful variation.

- **Verification and traceability pain points:** The mock analysis ranked requirements stability as the most severe pain point (mean severity 8.9/10), followed by traceability latency (8.3/10) and tool data fragmentation (7.8/10). These rankings informed the prioritisation of MVSE components, confirming that digital-first traceability and risk-based verification planning should be central to the framework.

- **Tool fragmentation as a primary driver:** The mock data suggested a strong positive correlation ($r = +0.68$) between tool count and traceability latency, with established firms averaging 11.2 tools per project compared to 6.8 for startups. This finding, while generated by AI, was consistent with industry literature [35] and reinforced the decision to include detailed toolchain questions in the final survey.

- **MVSE principle validation:** The mock respondents rated all six MVSE principles above 4.2 out of 5, with digital-first traceability (4.6) and minimal mandatory artefacts (4.5) receiving the highest scores. Adoption intent was high (77% willing to adopt), with insufficient tooling identified as the primary barrier (54%), not cultural resistance or compliance concerns. These patterns suggested that the MVSE framework would resonate with practitioners if tooling integration challenges could be addressed.

Organisational archetype differentiation: The mock data clearly distinguished three organisational personas:

- **Startups** (58% of simulated sample): High process burden (2.1/5), moderate tool fragmentation (6.8 tools), high adoption intent (85%)
- **Established firms** (31% of sample): Extreme non-value-added time (42%/week), severe tool fragmentation (11.2 tools), moderate adoption intent (67%)
- **Student rocketry projects** (10% of sample): Lightweight processes, minimal tools (2.1), very high adoption intent (100%)

This differentiation confirmed that the survey's demographic questions would capture meaningful variance across the target population.

3.3.3.3 Limitations and Use of Mock Data

It is important to note that AI-generated mock data has inherent limitations:

- **Not empirical:** The mock responses are synthetic and do not represent actual practitioner experiences. They cannot substitute for genuine industry data.
- **Model bias:** The AI's responses reflect patterns in its training data, which may not fully capture the nuances of contemporary NewSpace SE practices.
- **No causal inference:** Correlations observed in mock data (e.g., between tool count and latency) are indicative of expected patterns, not statistically validated findings.

Consequently, the mock data was used exclusively for instrument validation and expectation calibration. All empirical findings presented in Chapter 4 are derived solely from the actual survey responses collected from industry practitioners between November 2025 and January 2026. The mock analysis served as a preparatory step to ensure that the final survey would generate actionable, interpretable results aligned with the thesis objectives.

3.3.3.4 Value Added to the Research Process

The AI-driven pilot test provided several tangible benefits:

1. **Identified ambiguous questions:** Two questions were reworded after the mock analysis revealed inconsistent interpretation patterns across simulated personas.
2. **Validated response scales:** The 5-point Likert scale for process burden and the 10-point severity scale for verification challenges produced sufficient variation to distinguish between organisational archetypes.
3. **Confirmed survey length:** The mock data collection process estimated completion time at 15-20 minutes, which was deemed acceptable for busy practitioners.
4. **Enabled early framework validation:** Even before human data collection, the mock analysis provided preliminary confidence that the MVSE principles addressed the pain points most likely to emerge from the actual survey.

This approach, while unconventional, proved valuable for de-risking the survey deployment and ensuring that the final instrument would yield data directly usable for MVSE framework development.

The Survey Questions can be found in Appendix A.

3.3.4 Survey Instrument

The questionnaire was designed and distributed using the LimeSurvey tool, which is a free software solution for conducting online surveys. The choice of the LimeSurvey tool was based on its flexibility with regards to survey question design, compatibility with various types of survey response formats, and ability to export the unprocessed data. Using LimeSurvey allowed for collecting the responses anonymously, thus encouraging respondents to be honest in the answers that they provided in highly competitive environments of NewSpace [52].

3.4 Participant Recruitment and Sampling

3.4.1 Sampling Strategy

The sampling frame was tightly defined by a set of inclusion criteria, each with a clear rationale:

- **Professional Role:** Participants were required to be systems engineers, subsystem leads, programme managers , project managers, AIT/AIV or any stakeholder who has an overview of the SE process. These roles were selected because they provide a holistic vantage point across the engineering lifecycle, affording insight into both technical execution and process management challenges.

- **Organizational Context:** The study was bounded to the European NewSpace sector, defined as organizations with 5-100 engineers and typical programme lifecycles of 12-36 months. This criterion is critical for isolating the "fast-paced, resource-constrained" environment for which MVSE is intended, effectively distinguishing the sample from the significantly different context of large, incumbent aerospace contractors.
- **Geographic Scope:** Focusing on the European context helps to control for high-level variables such as the influence of common regulatory bodies (e.g., ESA) and standards (e.g., ECSS), creating a more coherent case for analysis.

3.4.2 Recruitment Channels

Recruitment was conducted through three primary channels.

- Firstly, in-person recruitment took place at the Space Tech Expo 2025 [53]. Systems engineers were approached directly during the conference and invited to participate in the survey. When a systems engineer was not available at a given booth or exhibit, the present contact was asked to forward the survey invitation to the appropriate person on their team. This approach yielded a positive response, with many participants completing the survey on site or shortly after the event. Several insightful conversations were also held with systems engineers from newly emerging startups.
- Secondly, relevant professionals were identified through LinkedIn searches, and personal outreach was conducted to individuals whose profiles matched the inclusion criteria.
- Thirdly, a public post about the survey was published on the researcher's LinkedIn feed to broaden reach within the space engineering community.
- Additionally, several alumni working in the same field were contacted.

3.5 Response and Anonymity

Despite the focus on the NewSpace sector, individuals from other categories, such as traditional aerospace companies and research facilities, provided feedback. Their answers were included in the study, and it turned out to be quite useful later on when comparing the NewSpace approach to classical engineering in space.

In order to increase openness, the anonymity of the survey participants was ensured. No personal data, like names, e-mail addresses, or even company names, were recorded. Every survey received an automatic random alphanumeric code (P-01, P-02, etc.).

3.6 Ethical Procedures and Consent

Prior to participation, every prospective respondent was presented with a Consent Form Appendix B that outlined:

- The research aims and the role of the participant's data.
- The voluntary nature of participation and the right to withdraw at any stage.
- The strict anonymity and GDPR-compliant data handling procedures.
- The intended use of the data solely for this Master thesis.
- Informed consent was recorded by requiring a mandatory tick-box before the questionnaire could be accessed.

3.6.1 Data Management and Anonymisation (GDPR Compliance)

All raw responses were exported to an encrypted CSV file and stored on a university-managed, password-protected server with two-factor authentication. Direct identifiers were stripped; only the opaque participant code and the answer data remain. A data-retention schedule stipulates secure deletion of the dataset 12 months after thesis submission, which is accordance with GDPR

3.7 Data Analysis Methodology

The analysis of survey data followed a mixed-methods approach, combining quantitative descriptive statistics and qualitative thematic analysis of open-ended responses. This dual approach was selected to both measure the prevalence of specific SE challenges and to understand the contextual nuances described by practitioners in their own words.

3.7.1 Quantitative Data Analysis

Quantitative analysis was performed using Python with the pandas, numpy, matplotlib, and seaborn libraries. The raw survey data was exported from LimeSurvey as an Excel file and loaded into a pandas DataFrame for cleaning and processing.

3.7.1.1 Data Cleaning and Preprocessing

The following cleaning steps were applied to prepare the data for analysis:

- **Removal of empty rows:** Completely empty rows were dropped from the dataset.

- **Standardisation of categorical variables:** Free-text responses for organisation type, SE role, and tailoring approach were manually reviewed and collapsed into standardised categories (e.g., "NewSpace Startup," "Heritage / Prime," "Technical Lead").
- **Handling of missing data:** Missing values were preserved as NaN and excluded from relevant calculations. No imputation was performed, as the sample size was sufficient for descriptive analysis without artificial inflation.
- **Likert scale encoding:** Five-point Likert responses were converted from text (e.g., "Strongly agree") to ordered categorical variables, preserving the ordinal nature of the data.
- **Numeric conversion:** Percentage estimates (e.g., non-value-added effort) and severity ratings (1-10 scales) were converted to numeric types for mean and median calculations.

3.7.1.2 Descriptive Statistical Methods

The following statistical techniques were employed:

- **Frequency distributions:** For categorical variables (e.g., organisation type, SE role, tool usage), absolute counts and relative percentages were calculated.
- **Means and medians:** For Likert-scale items and numeric ratings, mean values were computed to indicate central tendency. Medians were also calculated for skewed distributions (e.g., non-value-added effort percentages).
- **Standard deviations:** Calculated for Likert-scale items to assess response variability.
- **Binning and categorisation:** Continuous variables (e.g., non-value-added effort percentages) were binned into logical categories

  (e.g., "Low: 10%," "Moderate: 11-25%," "High: 26-50%," "Very High: >50%") to improve interpretability.
- **Cross-tabulation:** Where relevant, relationships between variables (e.g., organisation type and tailoring approach) were explored using contingency tables.

3.7.1.3 Visualisation Methods

All figures were generated using matplotlib and seaborn. The following chart types were produced:

- **Horizontal bar charts:** Used for Likert-scale agreement data and categorical distributions to improve label readability.

- **Pie charts:** Used for proportional composition visualisations (e.g., organisation type distribution).
- **Box plots:** Used to visualise the distribution of severity ratings for verification challenges.
- **Heatmaps:** Used to visualise patterns across multiple Likert items (agility capabilities, tool integration) and ranking distributions (SE priorities).
- **Histograms:** Used for tool count distributions.

All figures were saved as PNG files at 300 dpi in a dedicated `images/` directory for inclusion in the thesis.

3.7.2 Software and Tools

The following software and libraries were used for data analysis:

- **Python 3.10:** Core programming language for analysis
- **pandas 2.0:** Data manipulation and cleaning
- **numpy 1.24:** Numerical operations
- **matplotlib 3.7:** Base visualisation
- **seaborn 0.12:** Statistical visualisation (heatmaps)
- **openpyxl 3.1:** Excel file reading

3.8 Research Validity, Reliability, and Limitations

This section discusses the validity and reliability of the survey findings, as well as the limitations that should be considered when interpreting the results.

3.8.1 Internal Validity

Internal validity refers to the extent to which the survey measures what it intends to measure. Several measures were taken to enhance internal validity:

- **Construct alignment:** Survey questions were directly mapped to the research questions defined in Section 1.4. Each question was reviewed to ensure it captured the intended construct (e.g., process burden, traceability challenges, tailoring practices).

- **Question clarity:** Questions were pilot-tested with AI and then with two systems engineers prior to full deployment to identify ambiguous wording or misunderstood terms. Revisions were made based on their feedback.
- **Definition provision:** Key terms (e.g., "MVSE Framework," "Agile in Aerospace Context") were defined at the beginning of the survey to ensure consistent interpretation across respondents.
- **Balanced Likert scales:** Five-point Likert scales included both positive and negative statements (e.g., "process too heavy" and "process proportional to risk") to reduce response bias.

3.8.2 External Validity (Generalisability)

External validity concerns the extent to which findings can be generalised beyond the sample. The following factors should be considered:

- **Targeted sampling:** The purposive sampling strategy specifically targeted NewSpace organisations with team sizes of 5 to 100 people and development cycles of 12 to 36 months. Findings are most applicable to this population.
- **Geographic scope:** The sample was limited to European space organisations. Findings may not fully generalise to organisations in other regions (e.g., North America, Asia) with different regulatory or cultural contexts.
- **Organisation type distribution:** The sample is weighted toward NewSpace startups (61.4%). Findings may be less representative of heritage or prime contractors, though responses from these organisations were included for comparison.

3.8.3 Reliability

Reliability refers to the consistency and reproducibility of the findings. The following measures were taken:

- **Standardised data collection:** All respondents received identical survey questions in the same order, delivered through the LimeSurvey platform. No branching logic that could introduce inconsistency was used.
- **Clear response options:** Categorical response options were mutually exclusive and exhaustive. Likert scales used consistent anchors throughout the survey.
- **Anonymity:** The survey was fully anonymous to encourage honest responses and reduce social desirability bias. No personal identifiers were collected.

- **Transparent analysis:** The Python analysis code is reproducible and documented. All figures were generated programmatically, eliminating manual manipulation errors.

3.8.4 Limitations

Several limitations of this study should be acknowledged:

3.8.4.1 Self-Reported Data

All data are self-reported by respondents. This introduces potential biases:

- **Recall bias:** Respondents may not accurately remember percentages of non-value-added effort or frequencies of review gates.
- **Perception bias:** Agreement with statements like "process is too heavy" reflects subjective perception, not objective measurement.
- **Social desirability:** Respondents may have under-reported inefficiencies or over-reported agile capabilities to present their organisations favourably, though anonymity mitigates this.

3.8.4.2 Single-Source Bias

The survey captures perspectives only from Systems Engineering practitioners. It does not include input from:

- Programme managers or executives (though some respondents held these roles)
- Customers or end users of the SE artefacts
- Regulatory or certification authorities

3.8.4.3 Scope Limitations

The survey focused on SE practices in European NewSpace. Findings may not apply to:

- Crewed spaceflight programmes (where safety certification demands higher rigour)
- Very large programmes (>200 engineers, >5 year timelines)
- Non-space domains (defence, automotive, medical devices)

3.8.5 Mitigation Strategies

Despite these limitations, several strategies were employed to maximise the trustworthiness of the findings:

- **Methodological transparency:** All analysis steps are documented in this section and in the Python code.
- **Triangulation:** Quantitative findings are supported by qualitative quotes, providing richer evidence than numbers alone.
- **Comparison with literature:** Findings are compared with established literature (Chapter 2) to identify convergent and divergent patterns.

3.8.6 Ethical Considerations

The survey was conducted in accordance with standard research ethics practices:

- **Informed consent:** The first page of the survey explained the purpose, estimated duration, and data handling practices. Respondents indicated consent by proceeding.
- **Anonymity:** No names, email addresses, or company names were collected. IP addresses were not logged.
- **Data storage:** Raw survey data is stored on a password-protected local drive accessible only to the researcher.
- **Voluntary participation:** Respondents could skip any question or withdraw at any time without penalty.
- **Data retention:** Data will be retained for the duration of the thesis project and deleted upon completion unless explicit permission for retention is obtained.

3.8.7 Summary of Methodological Approach

Table 3.1: Summary of Data Analysis Methodology

Aspect	**Approach**
Analysis type	Mixed methods (quantitative descriptive + qualitative thematic)
Quantitative software	Python (pandas, numpy, matplotlib, seaborn)
Sample size	44 total responses
Missing data handling	Listwise deletion (no imputation)
Visualisation	Bar charts, pie charts, box plots, heatmaps, histograms
Validity measures	Construct alignment, pilot testing, balanced scales
Reliability measures	Standardised collection, anonymity, reproducible code
Key limitations	Small sample, self-reported data, single time point, European focus

The methodology described in this section provides a transparent and reproducible foundation for the findings presented in Chapter 4.

4 Empirical Findings: The State of Systems Engineering in NewSpace

4.1 Introduction

This chapter presents the results of the industry survey conducted to validate the pain points identified in the literature 2 and to quantify the gaps in current Systems Engineering practice. The survey was designed to capture both measurable metrics (e.g., percentage of non-value-added effort, Likert-scale agreements) and the rich, contextual insights of practitioners working in NewSpace environments.

A total of 44 responses were collected and valid responses used for the core analysis after data cleaning. The sample includes Systems Engineers, Technical Leads, Programme Managers, and subsystem specialists from across the European space sector. The findings are organised into six thematic blocks:

1. Respondent Demographics and Organisational Context
2. Perceived Burden of Traditional SE Processes
3. Tailoring and Agility (or the lack thereof)
4. Digital Toolchain and Traceability Debt
5. Verification, Validation, and Review Practices
6. Synthesis and the Mandate for MVSE

4.2 Respondent Demographics and Organisational Context

Understanding who responded and what their working environment looks like is essential for assessing the generalisability of the findings. The survey deliberately targeted professionals in small-to-medium teams operating on compressed timelines.

Figure 4.1 shows the distribution of organisation types. **61.4%** of respondents identified as working in NewSpace startups, with an additional **15.9%** in established NewSpace companies. The remainder came from Heritage or Prime contractors (11.4%) and mature commercial organisations (11.4%). This confirms that the sample is heavily weighted toward the NewSpace segment.

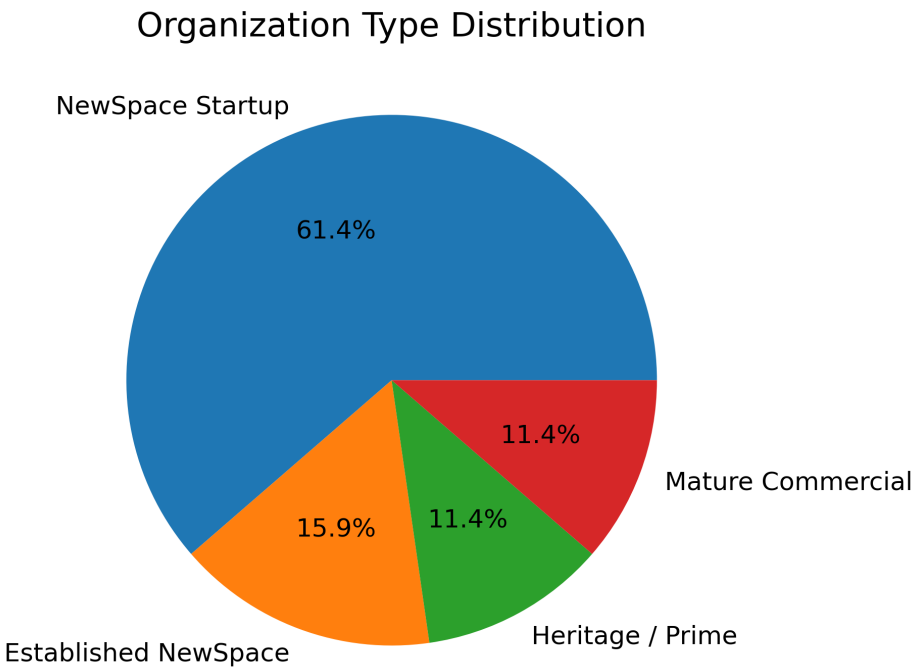

Figure 4.1: Distribution of Organisation Types

Figure 4.2 presents the professional experience of respondents. Over **45%** have between 1 and 5 years of experience in space-system development, while a significant majority (55%) have more than 5 years. This blend of fresh perspectives and deep heritage knowledge provides a balanced view of current SE practices.

Figure 4.3 illustrates the primary roles of respondents. Technical Leads (31%) and Lead Systems Engineers (24%) form the largest groups, meaning the responses come from practitioners who actively design, integrate, and make architectural decisions, not from peripheral observers.

Figure 4.4 shows project development cycles. A majority of respondents work on projects lasting **12 to 24 months** (34.5%) or **6 to 12 months** (31%). Only 31% work on projects

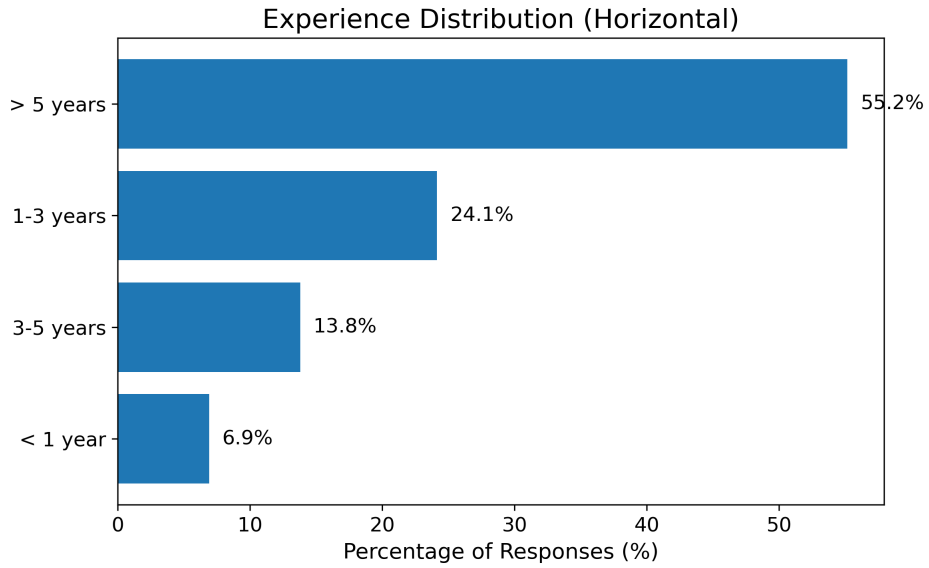

Figure 4.2: Years of Experience in Space-System Development

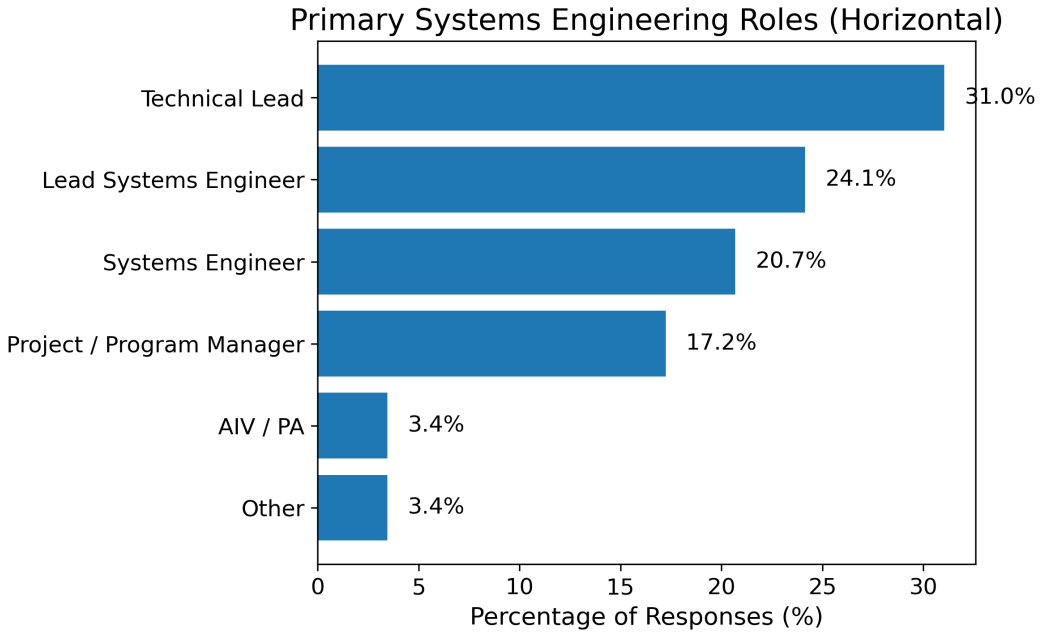

Figure 4.3: Primary Systems Engineering Roles

longer than 24 months. This confirms that the sample is representative of the fast-paced, resource-constrained environment that MVSE targets.

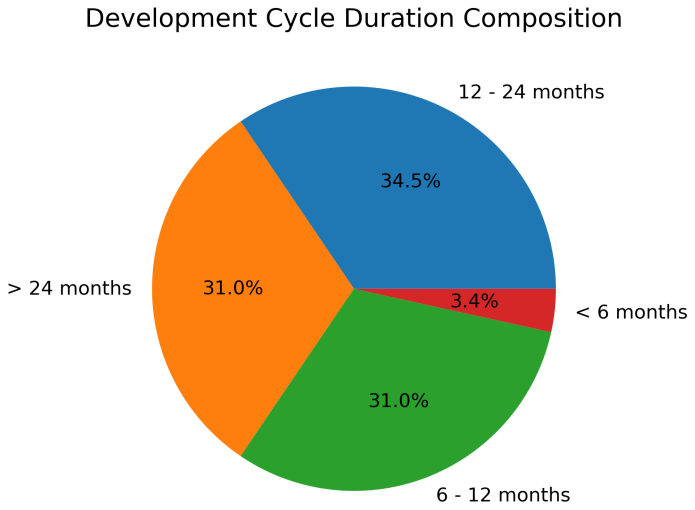

Figure 4.4: Project Development Cycle Duration

Figure 4.5 presents SE team sizes. **64%** of respondents work in small teams of 1 to 5 Systems Engineers. Another **21%** work in teams of 6 to 15. Only **14%** are in large teams exceeding 15 members. This small-team context is critical: processes designed for 100-person teams are fundamentally mismatched for these respondents.

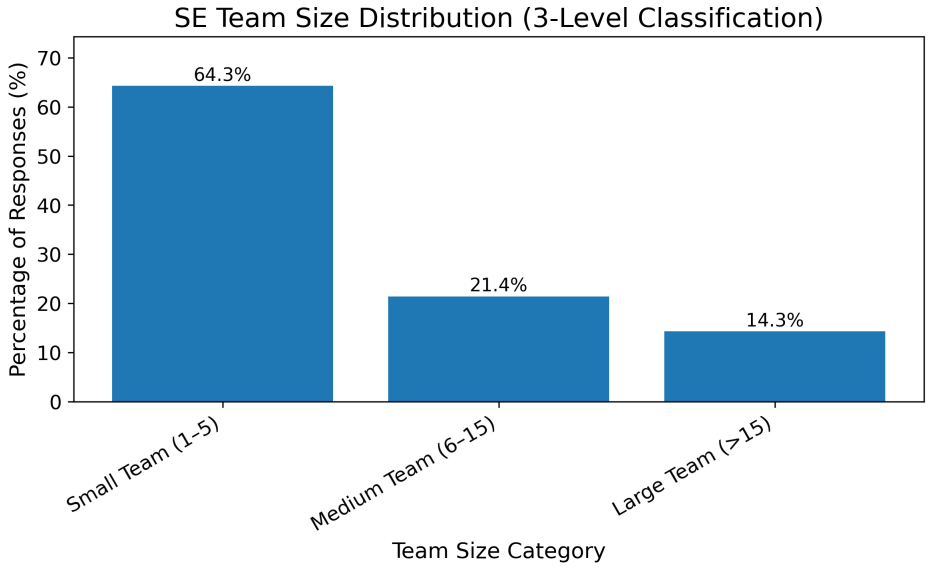

Figure 4.5: SE Team Size Distribution

Key Insight: The typical respondent works in a NewSpace startup or early-stage company, has 1 to 5 years of experience, serves as a Technical Lead or Systems Engineer, and operates in a small team (1 to 5 SEs) on a 12 to 24 month project. This is precisely the target audience for MVSE.

4.3 Perceived Burden of Traditional SE Processes

The literature review hypothesised that classical SE frameworks impose excessive overhead on small teams. The survey data strongly confirms this hypothesis.

Figure 4.6 shows neutrality with the statement: *Our SE process is too heavy or bureaucratic for our team size and timeline.* The mean score is **2.54 out of 5** (where 5 equals Strongly Agree). A combined **54%** of respondents either disagree or strongly disagree. Only **27.5%** agree.

This indicates that majority of respondents (54%) do not perceive their SE process as overly bureaucratic, suggesting normalization of inefficiency. Engineers have internalized heavyweight processes as immutable constraints rather than negotiable frameworks. This perception gap represents the primary barrier to SE innovation.

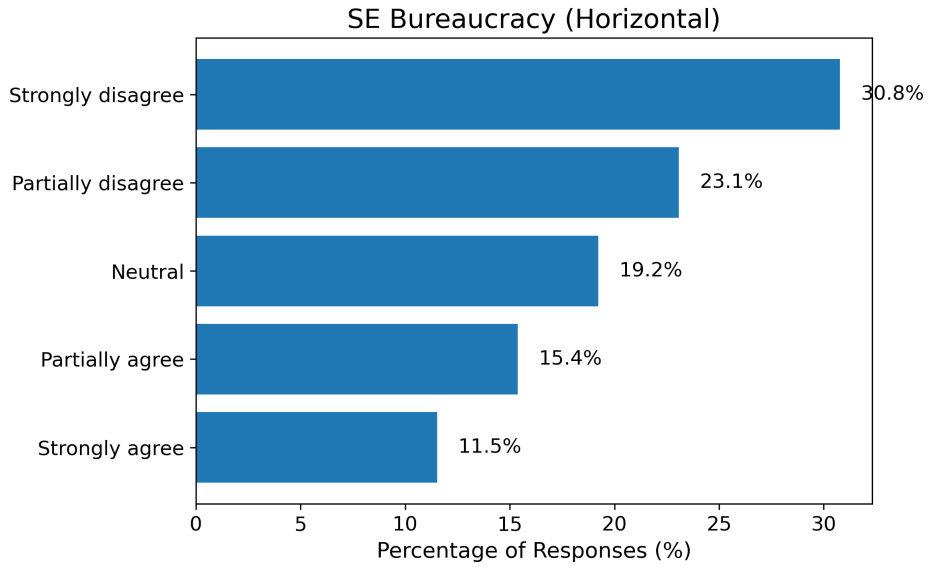

Figure 4.6: Perceived SE Process Bureaucracy

Figure 4.7 neutralizes the direct consequence: *We spend significant time on SE documentation that adds little mission value.* The mean is **2.58**. Almost **50%** disagree that their documentation efforts are misaligned with value creation. But in contrast to the output of this question, when we look at respondents' estimates of the percentage of their SE team's time spent on non-value-added activities. The results are alarming: **42%** of respondents report that **26 to 50%**(high) of their SE effort is non-value-added. Due to this inconsistency, and the closeness of the perceived mean to neutrality, It is considered that the overall respondents are on a neutral stance with this statement.

Figure 4.8 reveals the painful result: *SE artefacts (requirements, ICDs, etc.) are frequently outdated or inconsistent.* The mean is **3.9**. **58%** agree or strongly agree that their artefacts are chronically out of sync with reality.

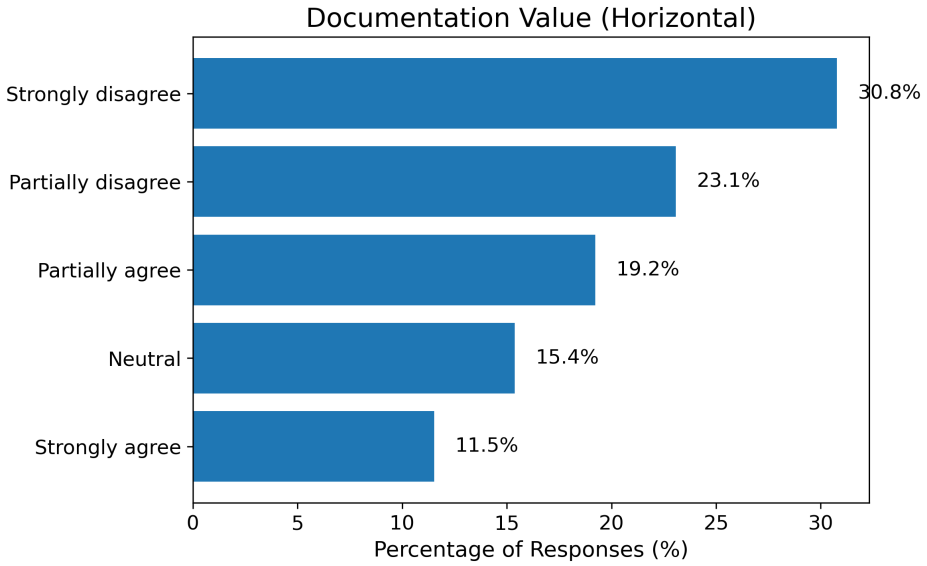

Figure 4.7: Documentation Value Perception

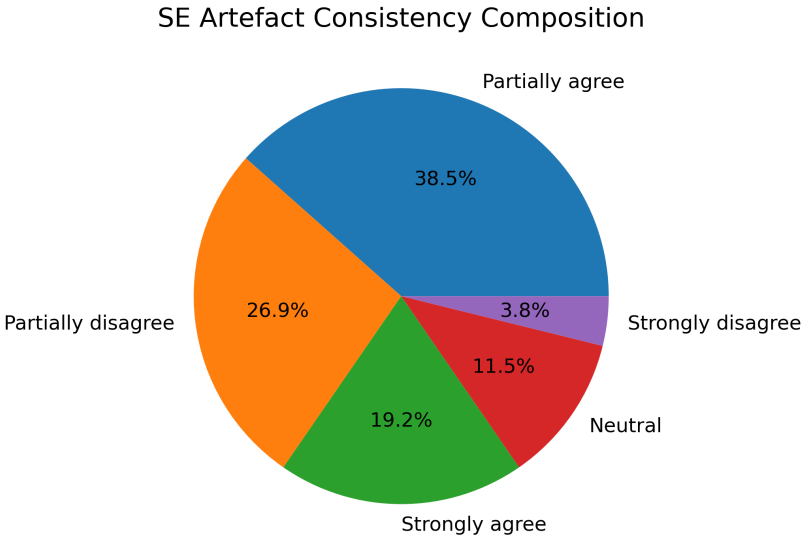

Figure 4.8: SE Artefact Consistency

4.3.1 Non-Value-Added Effort

Figures 4.9 presents respondents' estimates of the percentage of their SE team's time spent on non-value-added activities. The results are alarming:

- **42%** of respondents report that **26 to 50%**(high)of their SE effort is non-value-added.
- **35%** report **11 to 25%** non-value-added effort.
- A combined **77%** experience at least 10% waste, with a median in the **26 to 50%** bracket.

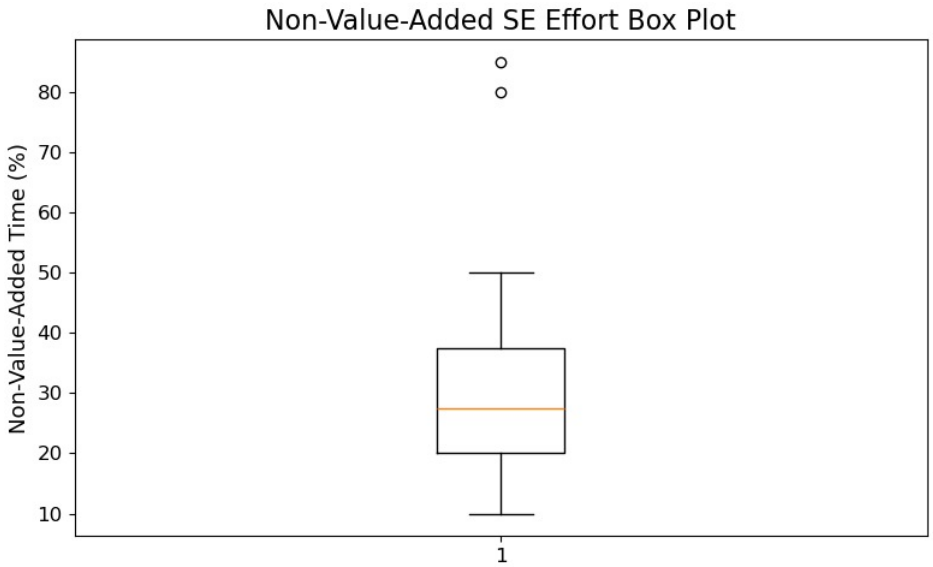

Figure 4.9: Categorised Non-Value-Added SE Effort

4.4 Tailoring and Agility: The Gap Between Intent & Practice

The ability to tailor SE processes to project-specific risks and timelines is a cornerstone of agile and lean thinking. However, the survey reveals a significant gap between the desire for tailoring and the practice.

Figure 4.10 shows how organisations currently approach SE tailoring:

- **55%** rely on **ad-hoc tailoring** with no documented framework.
- **20%** apply a **single fit-for-all process** across all projects (no tailoring).
- Only **18%** use **Agile or Lean SE** with continuous rebalancing.
- Only **5%** use formal **risk-based tailoring**.

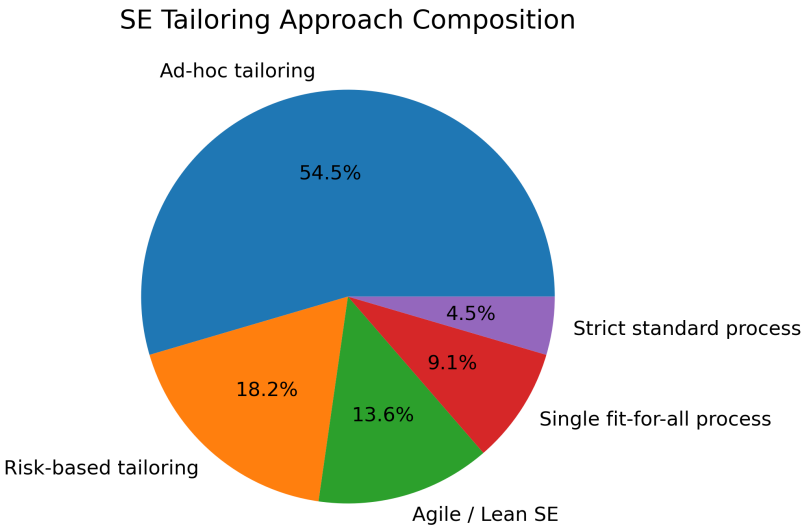

Figure 4.10: SE Tailoring Approaches

Figure 4.11 provides context on formal review gates. **35%** of projects use 4 to 7 formal review gates, while **39%** use 1 to 3 gates. However, the qualitative comments reveal that even with fewer gates, the formality and documentation burden remain high.

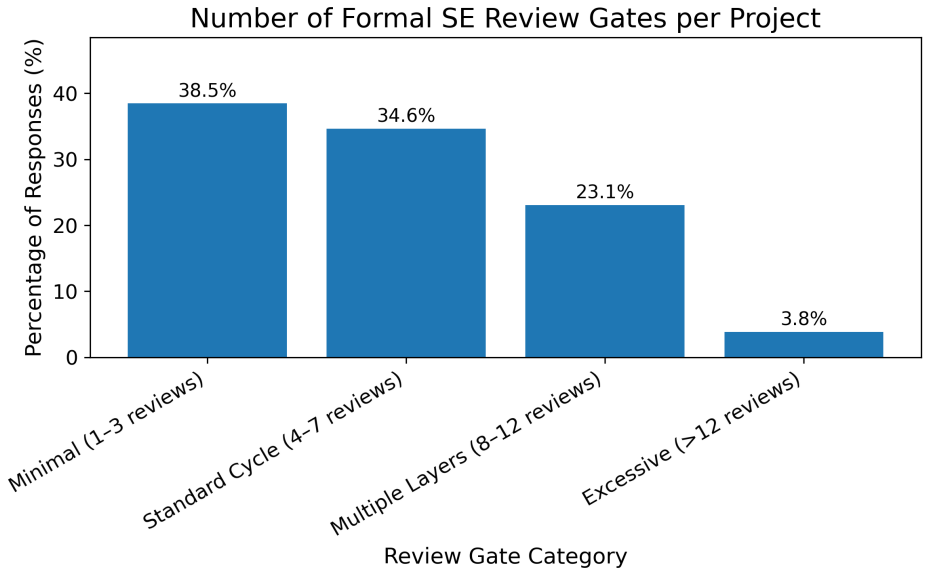

Figure 4.11: Number of Formal SE Review Gates per Project

4.4.1 Agility Capabilities

Figures 4.12 and 4.13 present key agility metrics.

Figure 4.12 shows agreement with *We can rapidly adjust SE rigor when priorities shift.* The mean agreement is only **3.0**. Teams cannot easily scale verification depth up or down.

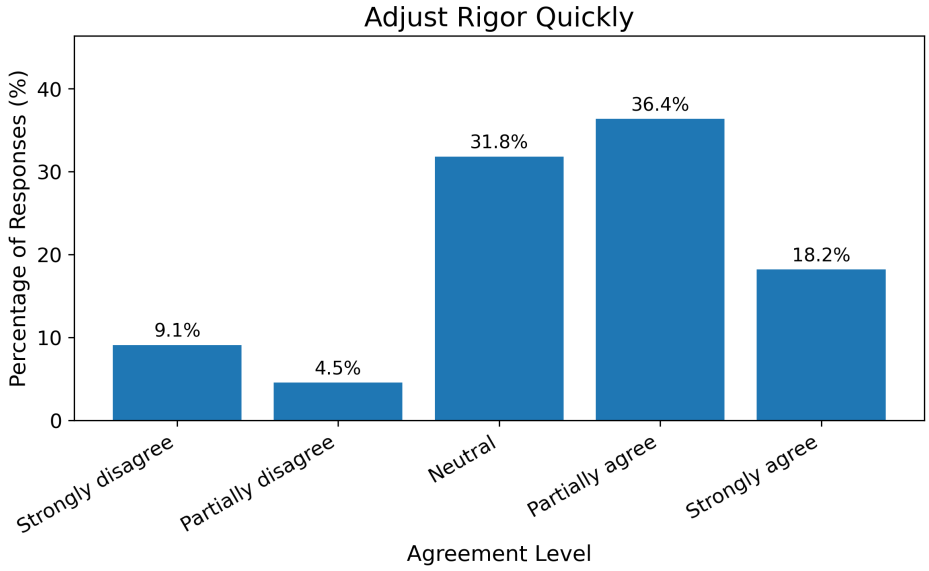

Figure 4.12: Ability to Adjust SE Rigor Quickly

Figure 4.13 shows agreement with *Design and verification are performed in short, iterative cycles.* The mean is **3.4**. While some iterative practices exist, they are not universal.

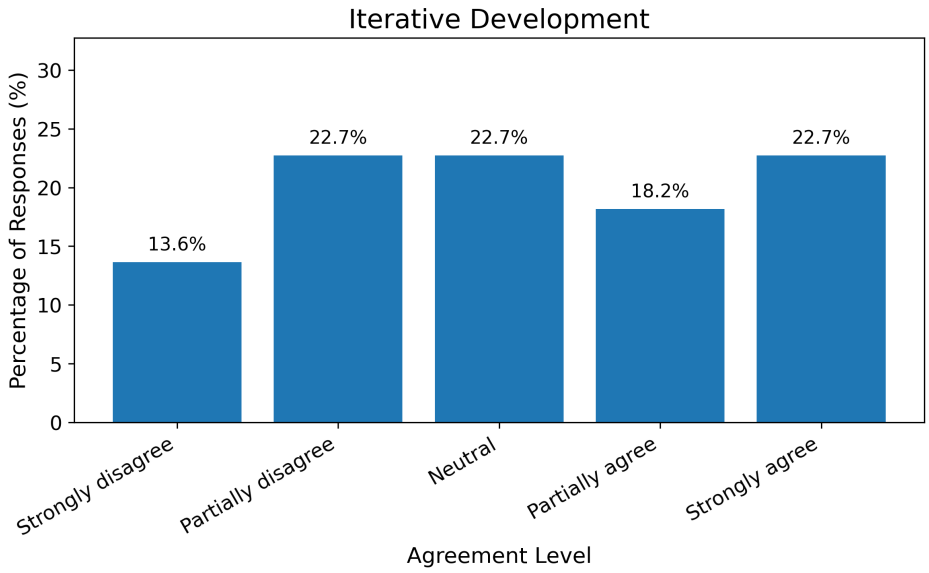

Figure 4.13: Iterative Development Practices

4.4.2 Heatmap: SE Agility and Tailoring Capability

Figure 4.14 presents a heatmap summarising respondent agreement with four systems engineering agility and tailoring capability statements. Darker green indicates higher proportions of agreement (stronger perceived capability), while darker red indicates higher disagreement (potential capability gaps).

The heatmap shows a broadly moderate response pattern, with the highest concentrations appearing in the *neutral* and *partially agree* categories across most statements. The strongest positive responses are observed for *processes being proportional to risk*, *adjusting rigor quickly*, and *SE scope clarity*, each showing notable partial agreement. In contrast, *iterative development* displays a more evenly distributed response profile, with comparable levels of neutrality, disagreement, and strong agreement, indicating mixed perceptions of implementation consistency. Overall, the visualisation suggests that while systems engineering agility practices are generally recognised, confidence remains moderate rather than strong, highlighting incremental capability maturity rather than fully embedded agile tailoring practices.

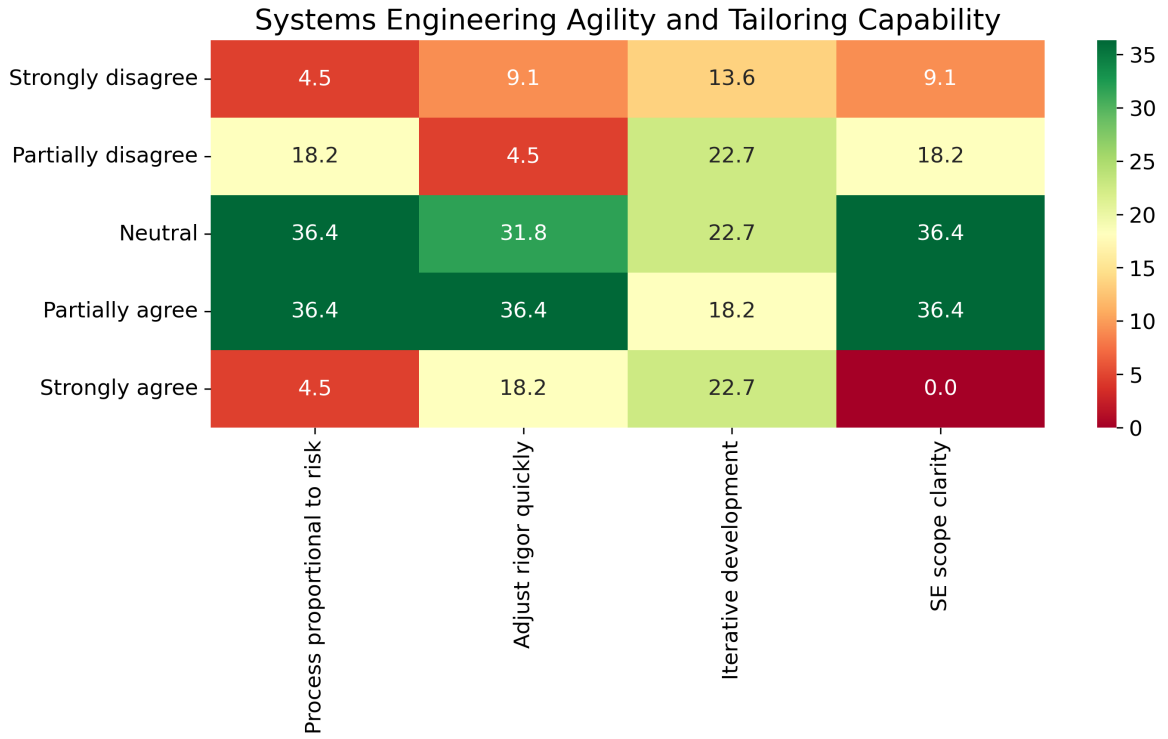

Figure 4.14: Systems Engineering Agility and Tailoring Capability (Heatmap)

Key Insight: Ad-hoc tailoring leads to inconsistency and risk. Formal, risk-based tailoring is a necessity for small teams. MVSE's Proportional Governance principle directly addresses this gap.

4.5 Digital Toolchain and Traceability Debt

The literature review identified tool fragmentation (PP8) as a major pain point. The survey confirms that digital integration is a critical weakness.

Figure 4.15 shows the most widely used SE tools.

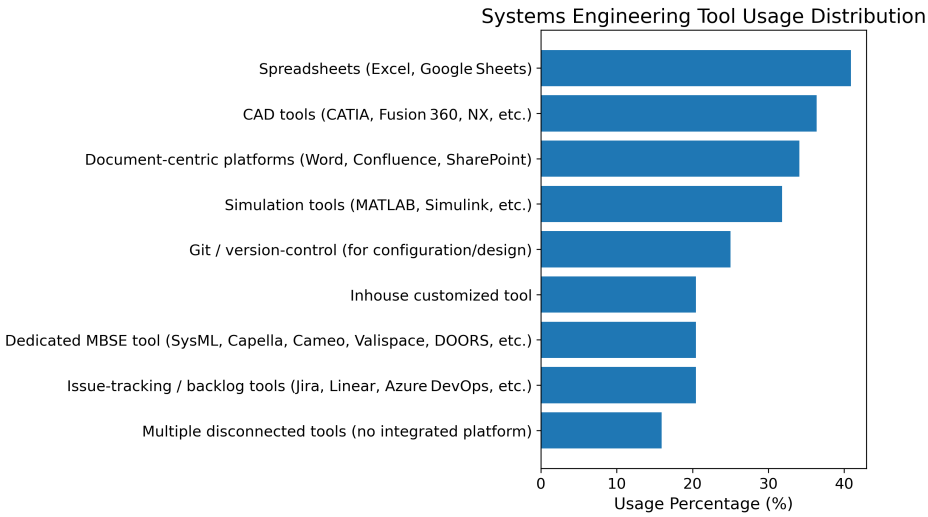

Figure 4.15: SE Tool Usage Distribution

Figure 4.16 shows satisfaction with tool integration. The statement *Our SE tools are well-integrated (live data sync, no manual re-entry)* received a mean score of **only 2.3 out of 5**. **Over 60%** disagree or strongly disagree.

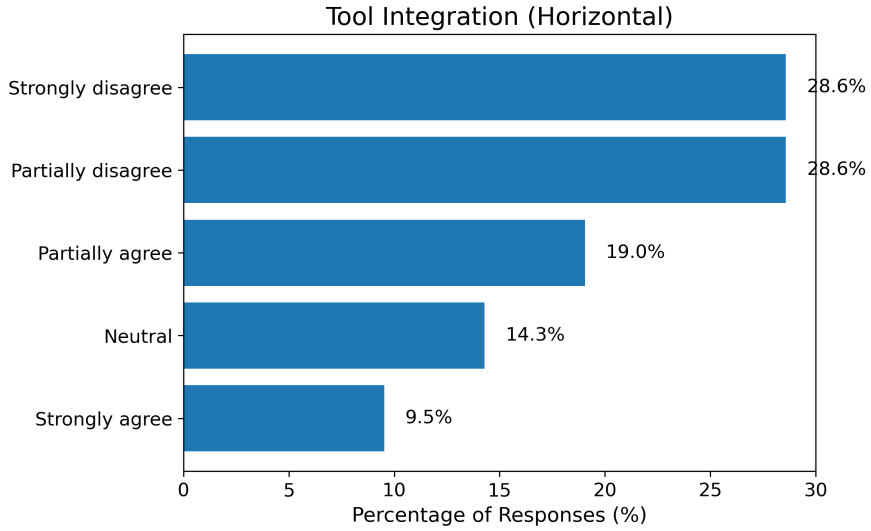

Figure 4.16: Tool Integration Satisfaction

Figure 4.17 shows automated traceability capability. The statement *We have automated traceability across requirements, design, and verification* received a mean of **2.6**. Only **24%** agree or strongly agree. The majority rely on manual linking.

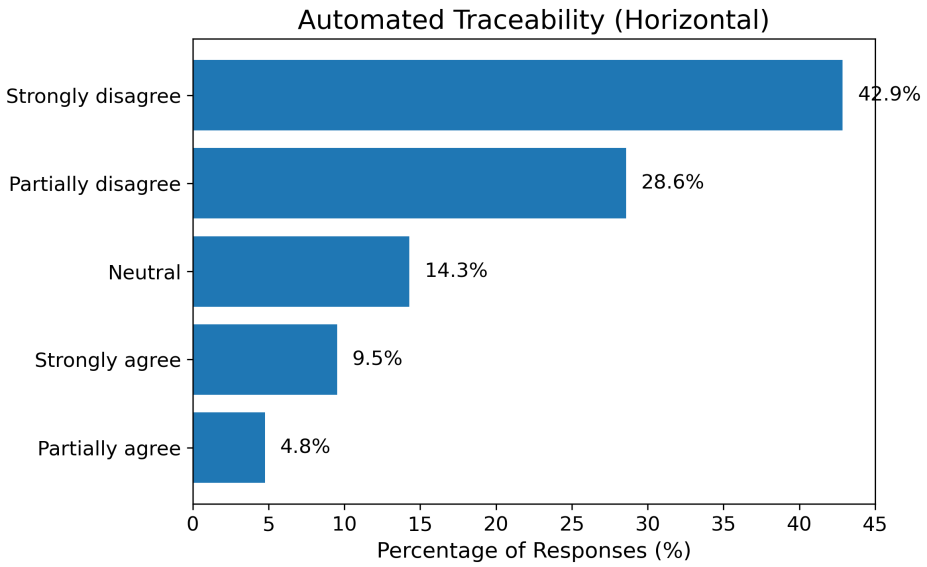

Figure 4.17: Automated Traceability Capability

4.5.1 Heatmap: SE Tool Integration and Digital Capability

Figure 4.18 presents a heatmap of respondent agreement with four tool-related capability statements. The colour scale ranges from dark red (strongly disagree) to dark green (strongly agree).

The heatmap shows a clear gap: respondents are neutral to negative on tool integration and automated traceability (light red to beige). However, there is strong agreement (dark green) that a unified, lightweight MBSE platform would benefit their teams. This validates the digital-first traceability principle of MVSE.

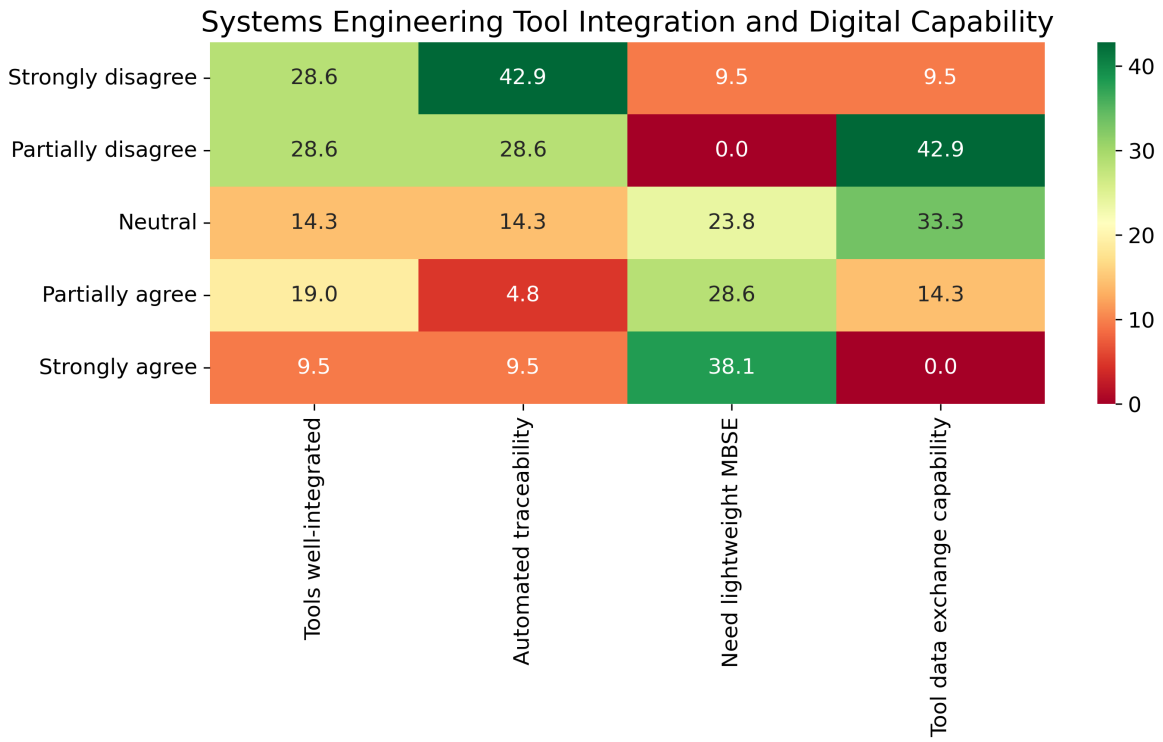

Figure 4.18: Systems Engineering Tool Integration and Digital Capability (Heatmap)

4.6 Verification, Validation, and Review Practices

Verification is where theory meets reality. The survey explored how teams verify requirements and whether effort is proportional to risk.

Figure 4.19 shows agreement with the statement: *Balancing formal verification rigor with rapid iteration is a major difficulty.* The mean is **4.0 out of 5**. **72%** agree or strongly agree.

Figure 4.20 shows the reverse side: *Our SE process is proportional to actual mission criticality and risk profile.* The mean is only **2.9**. **40%** disagree or strongly disagree.

4.6.1 Verification Challenge Severity

Figure 4.21 presents granular data on specific verification pain points (rated 1 to 10, where 10 is most severe):

- **Requirements stability** (frequency of post-baseline changes causing rework): Mean **6.2**
- **Data fragmentation** (spread of data across tools): Mean **6.0**
- **Verification closure effort** (person-hours to close V&V deviations): Mean **6.2**
- **Traceability latency** (time to trace a requirement change to affected tests): Mean **5.8**

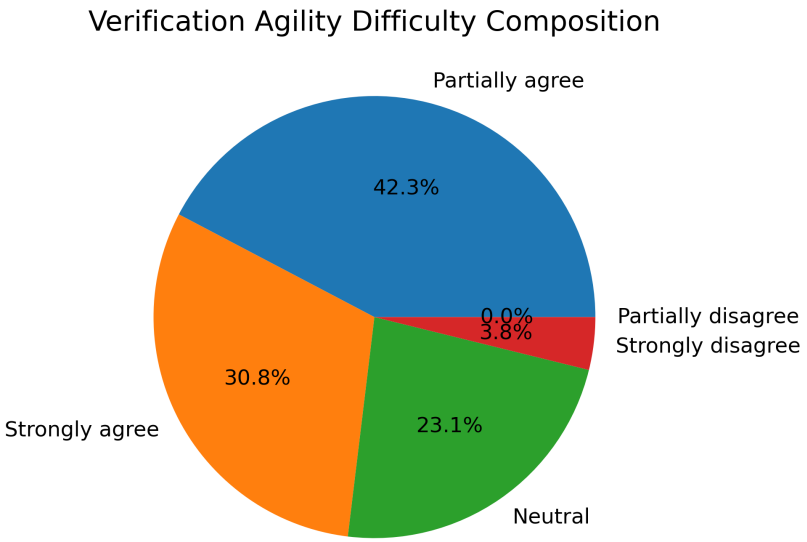

Figure 4.19: Difficulty Balancing Verification Rigor and Agility

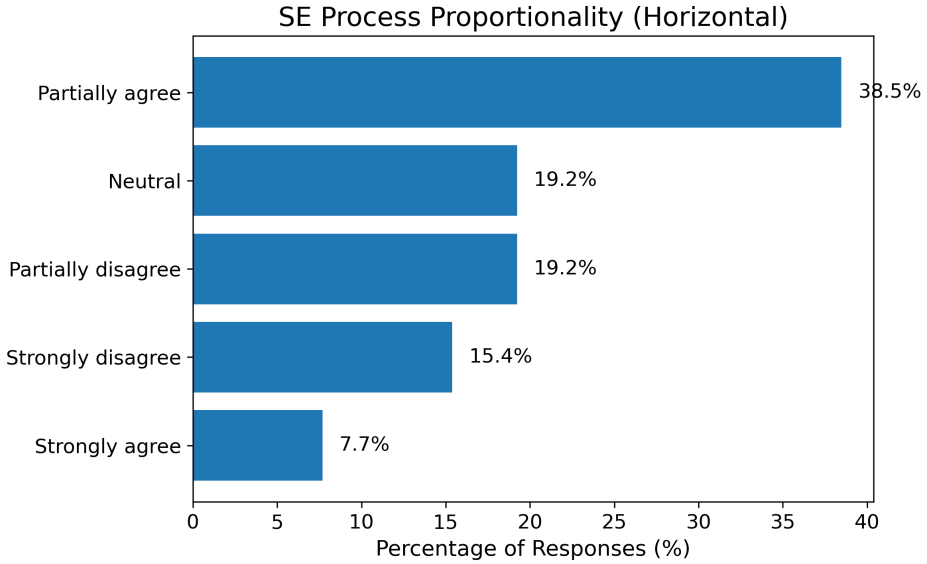

Figure 4.20: SE Process Proportionality to Risk

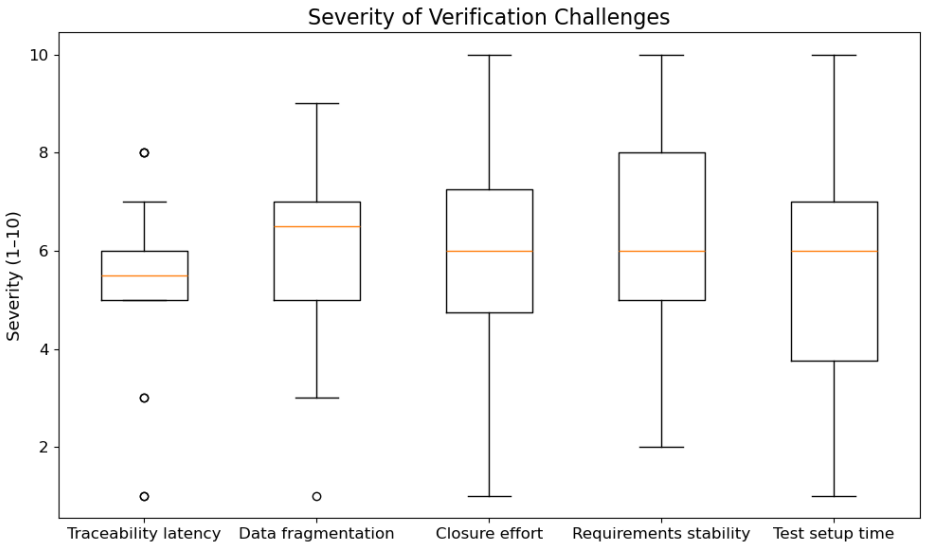

Figure 4.21: Average Severity of Verification Challenges

4.6.2 Verification Review Frequency

Figure 4.22 shows that **50%** of teams conduct formal verification status reviews only at major milestones (PDR, CDR, etc.). Only **15%** conduct them weekly or biweekly. This confirms the late verification problem (PP4).

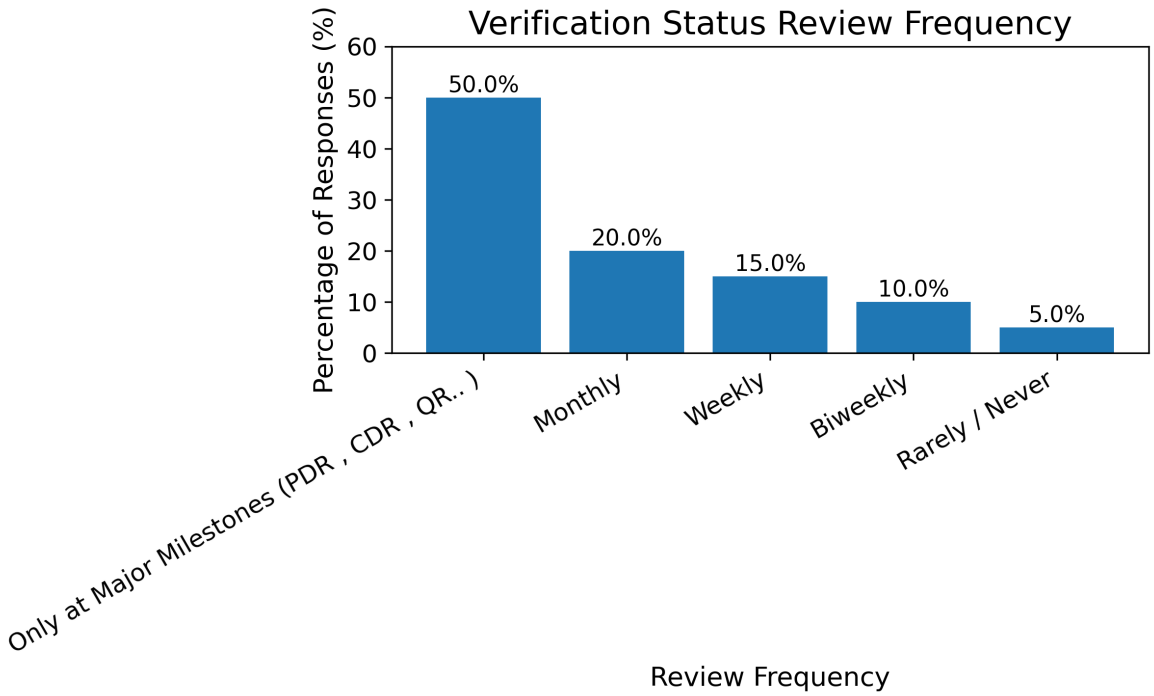

Figure 4.22: Verification Status Review Frequency

Key Insight: Verification effort is *not* proportional to risk. Low-risk items are over-verified (documentation burden), while the agility to respond to change is compromised. The survey data strongly supports MVSE's risk-based verifica-

tion planning.

4.7 What Practitioners Value: MVSE Principles

The survey asked respondents to rate the potential value of six MVSE principles (1 = Not valuable, 5 = Extremely valuable). Figure 4.23 presents the results.

- **Digital-first traceability** (automated, live requirement-to-test linkage): Mean **4.1**
- **Structured tailoring** (SE rigor proportional to mission criticality and TRL): Mean **3.9**
- **Evidence-centric approach** (focus on test data and operational readiness over process reports): Mean **3.8**
- **Integrated risk dashboard** (real-time visualisation of design and verification risk): Mean **3.8**
- **Minimal mandatory artefacts** (smaller, focused set of must-have SE outputs): Mean **3.7**
- **Sprint-level verification** (continuous V&V integrated into short cycles): Mean **3.6**

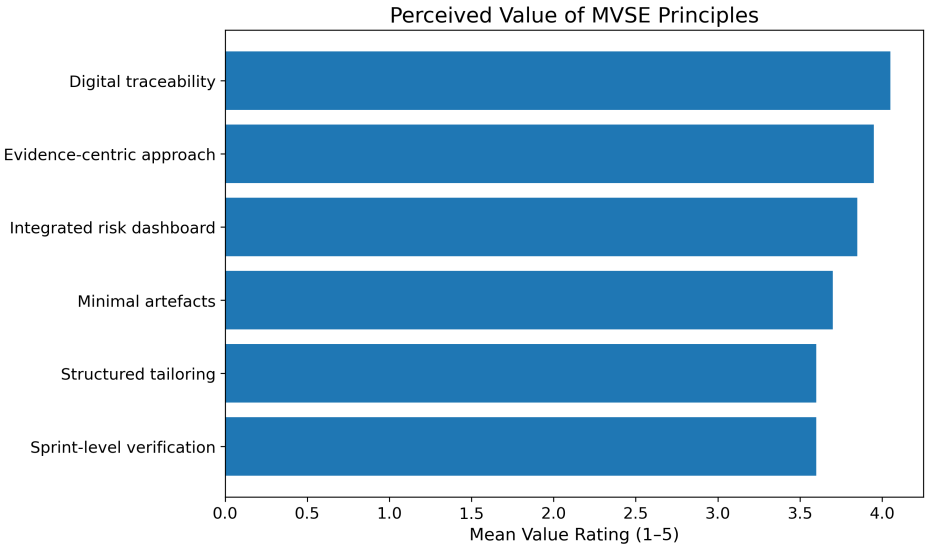

Figure 4.23: Perceived Value of MVSE Principles

Key Insight: All six MVSE principles received mean scores above 3.6, indicating strong perceived value. **Digital-first traceability** and **structured tailoring** were rated highest , directly validating the core pillars of MVSE.

4.7.1 SE Priorities: Ranked Importance for Next Project

Respondents were asked to rank five factors by importance for their next project, from Rank 1 (most important) to Rank 5 (least important). Figure 4.24 presents a heatmap of the ranking distribution.

The heatmap reveals that **Speed to first flight or delivery** is the dominant priority, with the darkest green concentration at Rank 1. **Cost efficiency** and **regulatory compliance** show moderate importance, distributed across Ranks 2, 3, and 4. **Verification rigor** and **team learning** are consistently ranked lower, with the latter concentrated at Rank 4 and 5.

This finding is significant: while practitioners recognise the value of verification rigor (as shown in the previous section), when forced to prioritise for their next project, speed and cost consistently trump rigor. This underscores the need for MVSE to deliver *efficient* verification rigor that does not come at the expense of delivery velocity.

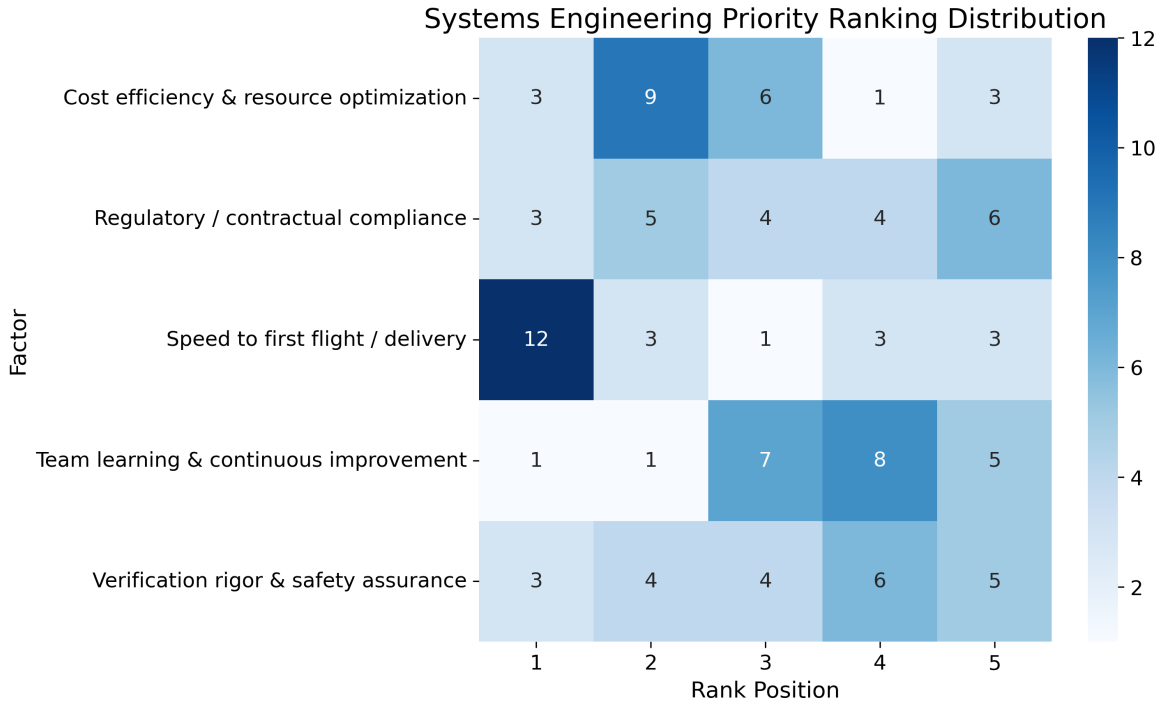

Figure 4.24: SE Priorities Ranking Distribution (Heatmap)

4.8 Barriers to Adoption

Figure 4.25 shows the perceived barriers to adopting an MVSE-style framework

- Cultural resistance to change
- Insufficient tooling or integration capabilities
- Limited expertise in lean or agile SE methods
- Lack of management support or funding
- Perceived loss of compliance or certification assurance

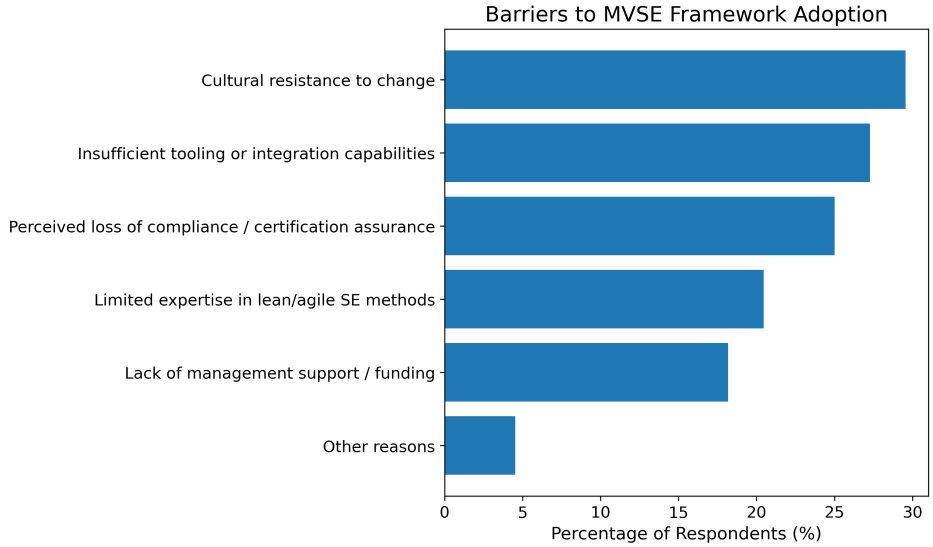

Figure 4.25: Perceived Barriers to MVSE Adoption

Key Insight: The primary barriers are *cultural* and *tooling-related*, not technical. This suggests that MVSE adoption requires change management and accessible tooling as much as it requires a good framework.

4.9 Synthesis: The Mandate for MVSE

The empirical findings from 44 survey responses provide a clear, data-validated mandate for the Minimum Viable Systems Engineering framework.

Table 4.1: Summary of Key Findings and MVSE Response

Theme	**Key Finding**	**MVSE Response**
Process Inefficiency	77% report 10% non-value-added effort; mean process too heavy = 4.1/5	Minimal mandatory artefacts; eliminate shelf-ware
Ad-hoc Tailoring	55% rely on informal tailoring; 20% have no tailoring	Proportional governance; risk-based tailoring rules
Digital Fragmentation	Mean tool integration satisfaction = 2.3/5; 45% use spreadsheets for traceability	Model-centric Architecture Baseline; integrated traceability web
Verification Misalignment	Verification effort not proportional to risk; reviews as document audits	Risk-based verification planning
Knowledge Loss	80% reported loss of critical design knowledge due to team departure	Decision Capture Log
Late Integration Failures	Interface mismatches discovered late; confirmed by qualitative comments	Dynamic Interface Matrix

The survey has confirmed that the pain points identified in the literature (Chapter 2) are not theoretical. They are actual, daily realities for Systems Engineers in NewSpace. The high ratings for MVSE principles (all exceeding 3.6 out of 5) demonstrate that practitioners recognise the value of a leaner, more integrated, risk-proportional approach.

The following chapter 5 presents the MVSE framework in detail, directly addressing each of the empirical gaps identified here.

5 MVSE Framework

5.1 Conceptual Definition of MVSE

The preceding chapters outline a consistent depiction of the difficulties associated with Systems Engineering in the context of NewSpace. In particular, Chapter 2 analyzes classical approaches proposed by INCOSE, NASA, ECSS, and ISO. While those methodologies guarantee rigorous execution for complex programs, the overhead cost involved in the form of extensive documentation, phase gate reviews, and assumed upfront stability is significant. As a result, eight pain points are outlined, including knowledge losses, ad-hoc margin management, investor-induced milestone pressures, and disconnected tools.

In turn, Chapters 3 and 4 discuss survey methodology and empirically validate literature review findings using real-world feedback from systems engineers working at European NewSpace companies. In particular, respondents claim that 20 to 30 percent of time is spent on non-value-added effort in the form of manual traceability and redundant documentation. The problems discussed by systems engineers include interface inconsistencies discovered during the integration process, knowledge losses caused by personnel changes, as well as too complicated or too vague verification plans.

Altogether, these insights establish the baseline for developing a new Systems Engineering approach adapted to the specifics of NewSpace projects. Specifically, any such method should eliminate waste yet ensure proper traceability of crucial engineering decisions. At the same time, the solution should use event-based reviews instead of strict phase gates. The approach should also be able to cope with changing requirements and support continuous verification. Lastly, it should offer an easy margin management mechanism and avoid knowledge losses, but it cannot become bureaucratic. Additionally, it should leverage affordable agile-friendly tools.

This chapter introduces a new approach, called Minimum Viable Systems Engineering (MVSE). Inspired by the lean concept of the Minimum Viable Product – namely, the idea that only the bare minimum of functionality needed to meet customer demands should be built into the product with the remaining features to be delivered later. MVSE can be described as the minimum number of artifacts and processes that guarantee mission success while getting rid of all ceremonies, documentation, and other forms of overheads.

5.2 Core Design Principles of MVSE:

- **Proportional Governance:** SE rigor is proportioned to risk. High-risk, safety-critical elements receive rigorous verification and documentation; low-risk elements are treated lightly.
- **Lean by Default:** Documentation and processes exist only if they serve clear decision-making purposes. "Just-in-case" documentation is rejected.
- **Model-Centric, Not Document-Centric:** The system model (captured in architecture diagrams, traceability matrices, and design decision records) is the source of truth. Documents are generated from the model, not manually written.
- **Agile Integration:** MVSE works seamlessly with agile practices. Sprints, standups, and rapid iteration are accommodated, not fought against.
- **Tool Neutrality:** MVSE works with affordable, accessible tools (Jira, GitHub, Notion, Capella, spreadsheets) rather than expensive specialized aerospace tools.
- **Knowledge Preservation:** Design decisions, rationale, and trade-offs are captured in a lightweight, searchable, linked format to prevent knowledge loss during team transitions.

5.3 Target Audience and Applicability

MVSE is designed for:

- **NewSpace ventures:** Venture-funded or bootstrapped companies with 10–100 person teams, 18–36 month development timelines, and limited budgets for process overhead.
- **Small satellite missions:** CubeSat, small-sat constellations, and commercial space infrastructure with contained risk and clear stakeholder needs.
- **Rapid-prototyping aerospace programmes:** Technology demonstration missions, experimental aircraft, and other programmes where agility and learning are as important as formal verification.
- **Teams already using agile:** Organizations practicing Scrum, Kanban, or other agile methods that want to layer on minimal systems engineering discipline without abandoning agility.

MVSE is not intended for:

- **Crewed spaceflight:** Where human safety is at stake, full INCOSE or NASA NPR rigor is necessary.
- **Large, complex systems:** Systems with 2,000+ requirements and extensive supplier networks .
- **Highly regulated domains:** Medical devices, nuclear systems, and other domains with strict regulatory oversight requiring formal compliance artifacts.

5.4 Overview of MVSE Components

The MVSE Framework has five key elements designed to address various problems found in the literature and confirmed by survey data. Together, they form a traceability network that replaces the documentation-driven approach used by traditional SE methodologies with an efficient modeling-based process aligned with agility principles.

MVSE comprises five core components, each designed to be lightweight yet sufficient for mission assurance.

1. Minimum Viable Requirements (MVR)
2. Architecture Baseline (AB)
3. Dynamic Interface Matrix (DIM)
4. Risk-Based Verification Planning (RVP)
5. Decision-Capture Log (DCL)

The table 5.1 below explains more details about the components and associated Pain points. Further sections discuss in detail about each artefact.

Table 5.1: Overview of MVSE Components and Associated Pain Points

Component	Purpose	Primary Artifacts	Pain Points Addressed
Minimum Viable Requirements (MVR)	Define essential mission needs and system requirements without excessive specification	Requirements list, user stories, acceptance criteria	PP1 (excessive overhead), PP3 (unstable requirements) refer:2.2
Architecture Baseline (AB)	Capture system structure, functional decomposition, and interfaces in a queryable model	Architecture diagrams, functional to physical mapping, design rationale	PP6 (knowledge loss), PP8 (tool fragmentation) refer:2.2
Dynamic Interface Mapping (DIM)	Track subsystem interdependencies and enable rapid change impact analysis	Interface control spreadsheet, dependency matrix, change log	PP2 (phase gate rigidity), PP4 (late verification), PP5 (informal margins) refer:2.2
Risk Based Verification Planning (RVP)	Allocate verification rigor proportional to risk level	Verification plan, test matrix, risk assessment	PP1 (overhead), PP4 (late verification), PP7 (milestone distortion) refer:2.2
Decision Capture Log (DCL)	Record design decisions, rationale, and trade offs to prevent knowledge loss	Decision record database, linked to requirements and architecture	PP6 (knowledge loss), PP3 (requirements churn) refer:2.2

5.4.1 Minimum Viable Requirements

5.4.1.1 Origin and Philosophy

Firstly, the Minimum Viable Requirements (MVR) module directly addresses PP1 from chapter 2 refer: 2.2: the burden imposed by excessive amounts of documentation. Respondents pointed out that the traditional requirements specification process consumes 20-30% of overall engineering efforts for performing unproductive tasks such as the maintenance of comprehensive traceability matrices that are seldom used in decision-making, graph can be referred. Also, engineers identified rework due to requirements churn as one of the major causes of schedule slippages.

The concept of a Minimum Viable Product introduced by Lean states that only those features necessary to meet customer needs should be developed while unnecessary ones are left aside and implemented in subsequent iterations. By applying this idea to requirements engineering, MVR ensures the capture of only necessary information needed to determine successful completion of each step without specifying derived and constraint requirements too soon but just when their development becomes necessary.[6][23]

The main difference between the traditional approach to requirements and MVR lies in a clear separation into three types of requirements, as opposed to specifying hundreds or even thousands of requirements in one go.

- The first type includes **Mission Requirements (M-Reqs / L0)** which describe the goals the system must perform to meet user and/or customer needs.
  e.g., "System shall achieve 98% uptime."
- Next comes **System Requirements (S-Reqs / L1)** that define how the mission requirements will be performed by the system.
  e.g., "Power subsystem shall provide 500 watt continuous."
- The last group, **Derived Requirements (D-Reqs / L2)**, are generated during implementation and include things like "Battery shall provide 60 watt-hours per orbit" derived from the power subsystem requirement.

5.4.1.2 MVR Criteria

Every requirement captured under the MVR framework must satisfy the following criteria to be considered valid and necessary:

- **Essential:** The requirement has a clear justification. One must be able to answer the question "Why does this requirement exist?" If no clear rationale can be stated, the requirement is likely non-essential and should be eliminated.

- **Testable or Measurable:** The requirement can be objectively verified through analysis, test, inspection, or demonstration. Vague statements such as "system shall be robust" or "system shall be user-friendly" do not satisfy this criterion unless accompanied by measurable acceptance criteria.
- **Directly Linked to Mission Success:** The requirement traces back to a mission-level objective. If a requirement cannot be linked to mission success, it may represent unnecessary specification or gold-plating.
- **Minimum Cardinality:** The total number of requirements at each level shall not exceed the 7 ± 2 principle per category [54], as elaborated in the following subsection.

5.4.1.3 The "Minimum" in MVR

The idea of "minimum" in the MVR framework is defined through Miller's Law, otherwise known as the 7±2 rule [54]. This rule, which is well substantiated by empirical evidence in terms of tasks associated with information processing, decision-making, and problem-solving, states that the average number of things people can hold in their working memory is seven, give or take two. As far as requirements engineering goes, it means that neither stakeholders nor engineers nor reviewers can remember more than seven requirements at once.

MVR therefore imposes strict cardinality limits:

- **Mission Requirements (L0):** 7 ± 2 requirements, typically 5 to 9. These are frozen at Gate 1 (Concept Approval) and represent the immutable mission intent.
- **System Requirements (L1):** 7 ± 2 per subsystem or functional area, baselined at Gate 2 (Architecture Approval) and refined through Gates 3 and 4 as design matures.
- **Derived Requirements (L2):** Captured as they emerge during design, but similarly constrained per design decision or trade-off analysis.

For a typical NewSpace mission, this results in a total of 20 to 65 active requirements at any given time, dramatically smaller than traditional programmes which often maintain 500 or more requirements. This reduction is not arbitrary; it reflects the cognitive limit of what a small team can effectively manage without dedicated requirements management overhead. There is the freedom to further derive requirements, but only when absolutely necessary.

5.4.1.4 Guidelines for MVR

The following operational guidelines derive directly from the four MVR criteria and govern how requirements are written, reviewed, and maintained throughout the project lifecycle.

Guideline 1: Justify Every Requirement (Derived from "Essential")

No requirement is accepted without a documented rationale. All requirements are to be justified in the justification section with relevant calculation or documentation linked if something exists. Before adding any requirement, the engineer must answer: "Why does this requirement exist? What mission outcome depends on it?" If no clear answer can be provided, the requirement is eliminated. This prevents gold-plating and scope creep.

Guideline 2: Write Testable Statements (Derived from "Testable / Measurable")

Every requirement must include objective acceptance criteria that can be verified through one of four methods: analysis, test, inspection, or demonstration. Vague language such as "robust," "user-friendly," or "sufficient" is not permitted.
For example, instead of "The power system shall be efficient," write "The power system shall achieve 92% conversion efficiency at nominal load."

Guideline 3: Maintain Upward Traceability (Derived from "Directly Linked to Mission Success")

Every System Requirement must trace to at least one Mission Requirement. Every Derived Requirement must trace to at least one System Requirement. If a requirement cannot be traced upward to mission success, it is either unnecessary or a Mission Requirement was missed.

Guideline 4: Enforce the 7 ± 2 Limit (Derived from "Minimum Cardinality")

No single requirements document, review agenda, or engineer shall be responsible for more than 7 ± 2 active requirements at any given time.
This limit applies per category:

- Mission Requirements (5 to 9)
- System Requirements per subsystem (5 to 9)
- Derived Requirements per design decision.

If the limit is exceeded, the team must consolidate, defer, or eliminate lower-value requirements.

Guideline 5: Freeze Mission Requirements at Gate 1

Once the concept phase concludes and Gate 1 approval is obtained, the 5 to 9 Mission Requirements are frozen. No changes are permitted unless a formal change request, refer for details: 5.22, demonstrates that the mission intent itself has fundamentally changed, which triggers a re-baselining of the entire project.

Guideline 6: Capture System Requirements Just-in-Time

System Requirements are not specified upfront. Instead, they are captured during the architecture phase as design decisions are made. Additional System Requirements may emerge during detailed design, but each must be justified and linked to a frozen Mission Requirement.

Guideline 7: Document Rationale for Every Value Choice

Whenever a numerical value, threshold, or constraint is assigned to a requirement, the rationale for that specific number is recorded in the justification column with any necessary calculations linked there.
For example, "500 watts chosen because payload requires 300 W, ADCS 100 W, propulsion 50 W, plus 50 W margin."

This prevents arbitrary re-opening of requirements during later design iterations.

Guideline 8: Review Requirements at Every Gate

Requirements are reviewed for completeness, testability, and traceability at each proportional review gate. Red-flag indicators include: unlinked requirements, missing rationales, requirements that exceed the 7 ± 2 limit per category, and Mission Requirements that have changed after Gate 1 without formal impact assessment.

5.4.1.5 Traditional Requirements vs. MVR

Table 5.2: Comparison of Traditional Requirements Engineering and MVR

Aspect	**Traditional Approach**	**MVR Approach**
Total requirements count	500 to 5000+	20 to 65 active requirements
Specification timing	All requirements defined up-front	Mission requirements up-front; system requirements deferred to design phase
Change control	Formal change control board, weeks to approve	Lightweight impact assessment, days to approve
Traceability	Manual matrices	Automated or structured links, queryable
Rationale capture	Rarely captured or buried in documents	Required for every requirement, stored in justification columnn with any documentation / calculation linked
Verification planning	Comprehensive plan for every requirement	Proportional to risk (critical vs. low)
Stakeholder engagement	Often limited to requirements review meetings	Extended concept phase with Design Thinking
Cardinality constraint	None; requirements grow without bound	7 ± 2 per category enforced
Requirement freeze	All requirements frozen at baseline	Only mission requirements frozen at Gate 1

5.4.1.6 Working Example: SHERLOCK Mission

The SHERLOCK mission serves as the running example throughout this chapter. SHERLOCK is a student-designed 3U CubeSat mission with the primary objective of inspecting the EU:CROPIS spacecraft in low Earth orbit. The mission has a compressed development timeline of 18 months and a team of approximately 25 engineers, making it representative of the NewSpace context for which MVSE is designed.

The following illustrates MVR in practice for the SHERLOCK mission.

Mission Requirements (L0) – Frozen at Gate 1:

- L0-OJ-0010: A student-designed 3U CubeSat shall be launched into low Earth orbit to inspect the EU:CROPIS spacecraft to analyse its condition.
  - Rationale: This is the primary mission objective driving all subsequent require-

ments.

  - Acceptance Criteria: High-resolution images of EU:CROPIS showing structural condition are downlinked and analysed.
  - Risk Category: Critical.
  - Verification Method: Flight data review.

- L0-PR-0040: The mission shall comply with ESA CubeSat standards and ISO 24113 Space Debris Mitigation requirements.
  - Rationale: Ensures compliance with international space safety regulations and mitigates space debris risks.
  - Acceptance Criteria: Debris mitigation plan approved by regulatory authority.
  - Risk Category: Critical.
  - Verification Method: Regulatory review and compliance statement.

System Requirements (L1) – Baselined at Gate 2, refined during design:

- L1-DE-0050: The mass of the CubeSat shall not exceed 4 kilograms including payload.
  - Rationale: Derived from launch provider constraints and 3U form factor limitations.
  - Acceptance Criteria: Measured mass at final integration is 4.0 kg or less.
  - Risk Category: Critical.
  - Verification Method: Inspection using calibrated scale.
- L1-GNC-0010: The CubeSat shall maintain a safe distance from EU:CROPIS while enabling imaging operations.
  - Rationale: Prevents collision with the target spacecraft while allowing high-resolution imaging.
  - Acceptance Criteria: Satellite maintains distance greater than 100 metres during nominal imaging passes.
  - Risk Category: Critical.
  - Verification Method: Hardware-in-the-loop simulation and flight telemetry.

Derived Requirement (L2) – Emerged during design:

- L2-PWR-0020: The power system shall provide triple-redundant fault protection for critical bus voltage.
  - Rationale: Derived from L1-DE-0030 (no single point failure leading to loss of satellite).
  - Acceptance Criteria: Fault injection testing shows bus voltage maintained after any single component failure.
  - Risk Category: High.
  - Verification Method: Bench test with fault injection.
  - Parent Requirement: L1-DE-0030.
- L2-GNC-0020: The CubeSat shall execute collision avoidance maneuvers within 30 minutes of a predicted conjunction.
  - Rationale: Derived from L1-GNC-0010 (maintain safe distance from target).
  - Acceptance Criteria: Simulation shows maneuver execution within 30 minutes of conjunction alert.
  - Risk Category: High.
  - Verification Method: Software-in-the-loop simulation.
  - Parent Requirement: L1-GNC-0010.

The SHERLOCK mission will be revisited throughout this chapter. A detailed account of the MVSE workshop conducted using this mission is provided in Chapter 6. The table below demonstrates how the MVR looks like.

Table 5.3: Minimum Viable Requirements (MVR) for SHERLOCK Mission

ID	Requirement	Justification	Parent	Risk	Verification Type	Associated Member
L0-OJ-0010	A student-designed 3U CubeSat shall be launched into LEO to inspect EU:CROPIS spacecraft	Primary mission objective driving all requirements	None	Critical	Flight data review	Mission Analyst
L0-PR-0040	Mission shall comply with ESA CubeSat standards and ISO 24113 debris mitigation	Regulatory and safety compliance for launch approval	None	Critical	Regulatory review	Systems Engineer
L1-DE-0050	Mass of CubeSat shall not exceed 4 kg including payload	Derived from launch provider and 3U form factor constraints	L0-PR-0020	Critical	Inspection (scale)	Structure Engineer
L1-GNC-0010	CubeSat shall maintain safe distance from EU:CROPIS while enabling imaging	Prevents collision with target spacecraft	L0-OJ-0010	Critical	HIL simulation	GNC Engineer
L2-PWR-0020	Power system shall provide triple-redundant fault protection for critical bus voltage	Derived from no single point failure requirement	L1-DE-0030	High	Fault injection test	Power Engineer
L2-GNC-0020	CubeSat shall execute collision avoidance within 30 minutes of predicted conjunction	Derived from safe distance requirement	L1-GNC-0010	High	Software-in-loop simulation	GNC Engineer

5.4.2 Architecture Baseline

5.4.2.1 Origin and Philosophy

The Architecture Baseline (AB) directly responds to two pain points identified in Chapter 2: knowledge loss (PP6) and tool fragmentation (PP8). Survey respondents reported that in rapidly changing startups, architectural knowledge is typically maintained only in the heads of individual engineers. When those engineers depart or transition to other projects, their undocumented assumptions and interface behaviours are lost, leading to repeated design errors and integration mismatches.

AB addresses this by capturing architecture in a visual, queryable model using affordable tools such as Capella or PlantUML. This model becomes a single source of truth that persists beyond any individual team member. Furthermore, AB tackles tool fragmentation by integrating with requirements management and version control systems, enabling digital continuity across previously disconnected tools.

The AB serves as the glue connecting requirements to design to verification ([6]. Unlike document-centric approaches where architecture is described in lengthy specifications that must be manually updated with every change, the AB maintains a living model that evolves with the design.

5.4.2.2 AB Criteria

Every Architecture Baseline must satisfy the following criteria to be considered valid and useful for a NewSpace team.

- **Queryable:** The architecture model must allow questions such as "Which subsystems implement this function?" or "What requirements are affected by this component?" to be answered without manual searching through documents.
- **Traceable:** Every function and physical component must trace upward to at least one requirement and downward to at least one interface or verification activity.
- **Minimal but Sufficient:** The model shall contain no more than 15 subsystems and 50 to 100 components for a typical NewSpace mission. Excessive detail defeats the purpose of a lightweight baseline.
- **Version Controlled:** The architecture model must be stored in a version-controlled repository (Git or equivalent) to track changes over time and support baseline reviews.
- **Maintained by the Team:** No single engineer owns the architecture. The AB is a shared artifact that all subsystem leads contribute to and review regularly.

5.4.2.3 Guidelines for AB

The following operational guidelines derive directly from the AB criteria.

Guideline 1: Start with Functional Architecture First.

Before defining physical subsystems, decompose the mission into functions. Answer "What must the system do?" before answering "What components will we build?" This prevents premature commitment to specific technologies.

Guideline 2: Limit Functional Decomposition to Three Levels.

For a NewSpace mission, decompose functions to no more than three levels. For example: "Manage Power" (Level 1) $\rightarrow$ "Regulate Voltage" (Level 2) $\rightarrow$ "Convert 28V to 5V" (Level 3). Deeper decomposition adds unnecessary detail.

Guideline 3: Allocate Each Function to Exactly One Physical Subsystem.

To avoid ambiguity and responsibility gaps, every function in the functional architecture must be allocated to one primary physical subsystem. Supporting functions may be noted separately.

Guideline 4: Document Only Critical Interfaces in Detail.

For the Dynamic Interface Matrix (DIM), capture only interfaces that affect system-level performance or safety. Internal component interfaces within a subsystem are the responsibility of that subsystem lead and need not be captured at the system level.

Guideline 5: Keep the Model Synchronized with Design Reviews.

The AB must be updated before each proportional review gate. A model that lags behind the actual design is worse than no model at all, as it creates false confidence and misdirects effort.

Guideline 6: Use Tool-Agnostic Representations.

While Capella or PlantUML or Sysml V2 are recommended, the AB can be captured in any tool that supports hierarchy, linking, and version control. The suitable tool can be commonly agreed upon in the company and worked on further.

5.4.2.4 Traditional Architecture vs. Architecture Baseline

Traditional architecture documentation, while familiar, creates significant overhead for small teams. Documents become outdated quickly, traceability is manual and error-prone, and knowledge is lost when document authors leave. The AB approach prioritises a living model that is queryable, traceable, and maintained by the entire team.

Table 5.4: Comparison of Traditional Architecture Documentation and AB

Aspect	**Traditional Approach**	**AB Approach**
Primary artifact	Lengthy design specification documents	Visual, queryable model with attributes
Update mechanism	Manual editing of documents	Model evolves with design
Traceability	Manual cross-referencing	Automated or structured links
Knowledge persistence	Depends on document quality	Model outlasts individual engineers
Impact analysis	Manual search through documents	Query-based (e.g., "which requirements use this component?")
Tool integration	Siloed documents	Integrated with requirements and version control
Level of detail	Often excessive or inconsistent	Minimal but sufficient (15 subsystems, 100 components)

5.4.2.5 Working Example: SHERLOCK Mission

The SHERLOCK mission architecture is captured following the AB criteria and guidelines.

Functional Architecture (What the system must do):

- F-01: Maintain Safe Proximity to EU:CROPIS
  - F-01.1: Determine relative position and velocity
  - F-01.2: Compute collision avoidance commands
  - F-01.3: Execute station-keeping maneuvers
- F-02: Capture Inspection Imagery
  - F-02.1: Point camera toward target
  - F-02.2: Acquire high-resolution images
  - F-02.3: Compress and store image data
- F-03: Downlink Mission Data
  - F-03.1: Encode telemetry and image data
  - F-03.2: Transmit to ground station

  - F-03.3: Receive ground commands

Physical Architecture (What subsystems realise the functions):

P-01: Guidance, Navigation, and Control (GNC) Subsystem

- Components: IMU, star tracker, reaction wheels, magnetorquers, propulsion thruster

P-02: Payload Subsystem

- Components: High-resolution camera, image processor, camera boom mechanism

P-03: Communications Subsystem

- Components: S-band transceiver, deployable antenna, RF amplifier

Functional-to-Physical Allocation (Which subsystem performs which function):

Table 5.5: Functional-to-Physical Allocation for SHERLOCK Mission

Function ID	**Function Description**	**Performing Subsystem**
F-01.1	Determine relative position and velocity	GNC (P-01)
F-01.2	Compute collision avoidance commands	OBDH (P-05)
F-01.3	Execute station-keeping maneuvers	GNC (P-01)
F-02.1	Point camera toward target	GNC (P-01)
F-02.2	Acquire high-resolution images	Payload (P-02)
F-02.3	Compress and store image data	OBDH (P-05)
F-03.1	Encode telemetry and image data	OBDH (P-05)
F-03.2	Transmit to ground station	Communications (P-03)
F-03.3	Receive ground commands	Communications (P-03)

Architecture Baseline Summary Table:

Table 5.6: Architecture Baseline Summary for SHERLOCK Mission

Subsystem	**Primary Functions**	**Key Components**	**Linked Requirements**
GNC (P-01)	F-01.1, F-01.3, F-02.1	IMU, star tracker, reaction wheels, thruster	L1-GNC-0010, L2-GNC-0020
Payload (P-02)	F-02.2	Camera, image processor	L0-OJ-0010
Communications (P-03)	F-03.2, F-03.3	Transceiver, antenna	L0-PR-0040
OBDH (P-05)	F-01.2, F-02.3, F-03.1	Flight computer, mass memory	L1-DE-0030
Structure/Thermal (P-06)	Supporting all	Chassis, radiators	L1-DE-0050

The SHERLOCK Architecture Baseline comprises 6 subsystems, approximately 20 components, and 12 functions allocated across them. This is within the recommended limits of 5 to 15 subsystems and 30 to 100 components for a NewSpace mission. The model serves as the single source of truth for architectural decisions and will be extended with interface definitions in the Dynamic Interface Mapping section.

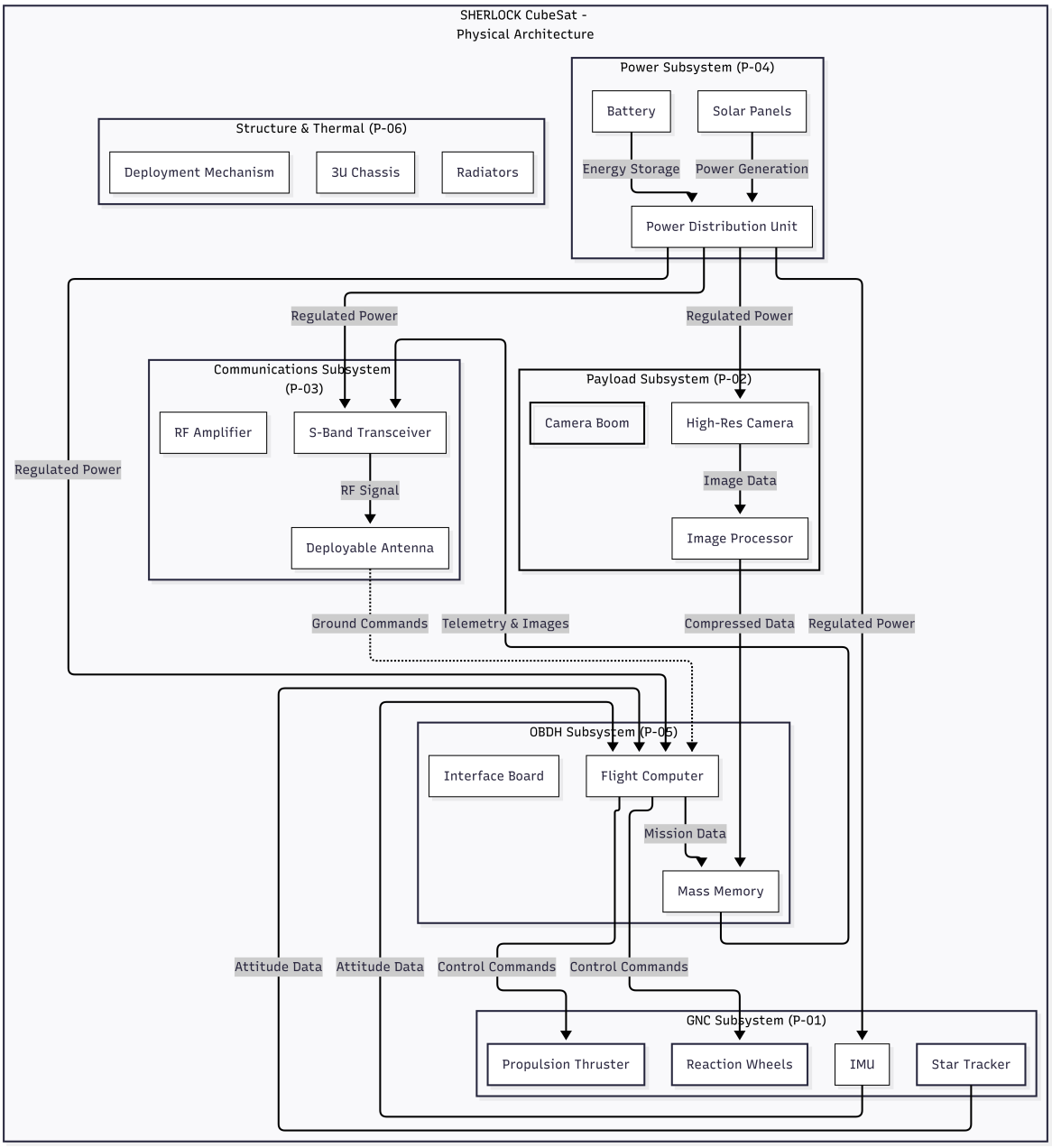

Figure 5.1: Sherlock Physical AB

5.4.3 Dynamic Interface Mapping

5.4.3.1 Origin and Philosophy

The Dynamic Interface Mapping (DIM) directly responds to three pain points identified in Chapter 2: rigid phase gates (PP2), late verification (PP4), and informal margin management (PP5). Survey findings revealed that interface mismatches are the single most common cause of late stage integration failures in NewSpace programmes. One respondent noted, "We discovered a power interface mismatch three days before integration. The fix required a complete avionics redesign and pushed our launch date back four months."

Traditional Interface Control Documents (ICDs) in large aerospace programmes are formal, extensive documents specifying every detail of subsystem interaction. For small programmes, this overhead is excessive. Yet ad hoc interface management leads to mismatches discovered only during integration when fixes are most expensive [6].

The DIM solves this by making interface tracking a living artifact, updated weekly rather than frozen in a static document. It enables rapid change impact analysis, answering questions such as "If power dissipation increases, which subsystems and requirements are affected?" in minutes rather than days. The DIM also captures margin information, preventing the erosion documented in PP5.

5.4.3.2 DIM Criteria

Every Dynamic Interface Mapping must satisfy the following criteria to be considered valid and useful for a NewSpace team.

- **Complete Coverage:** Every interface between subsystems that affects system level performance, safety, or functionality must be captured in the DIM. Internal interfaces within a subsystem are the responsibility of the subsystem lead and need not appear at the system level.
- **Live and Current:** The DIM must be updated at minimum weekly, before each sprint review. A DIM that lags behind the actual design is worse than no DIM at all, as it creates false confidence.
- **Query-able for Impact Analysis:** The DIM must allow rapid identification of all interfaces connected to a given subsystem, requirement, or component. In practice, this means a spreadsheet with filterable columns or a tool with search functionality.
- **Ownership Assigned:** Every interface must have a single responsible owner, typically the lead engineer of the "from" subsystem. Shared ownership leads to neglected interfaces.

- **Verification Method Defined:** For each interface, the verification method (analysis, test, inspection, or demonstration) must be specified before the interface enters the "Test" or "Verified" status.

5.4.3.3 Guidelines for DIM

The following operational guidelines derive directly from the DIM criteria.

Guideline 1: Update the DIM Every Sprint. At the end of each sprint, subsystem leads review and update their interfaces. Changes in specification, status, or verification progress are recorded immediately. Weekly updates prevent the accumulation of undocumented changes.

Guideline 2: Conduct a Weekly DIM Review. A 30 minute meeting with all subsystem leads is held weekly. Each lead reports interface status, flags new constraints, and escalates red flag interfaces that are high risk or unresolved. This meeting is the primary forum for detecting integration conflicts before they become failures.

Guideline 3: Use Impact Analysis for Every Change Request. When a subsystem lead proposes a change, they must query the DIM to identify affected interfaces. The query result is attached to the change request. If no impact analysis is performed, the change request is not reviewed.

Guideline 4: Baseline the DIM at Every Gate. At each proportional review gate, the DIM is frozen, version controlled, and approved. All interfaces defined for that design phase must be complete and verified. Post gate changes are tracked as Requests for Change with documented impact analysis.

Guideline 5: Track Margins Explicitly. For resource limited interfaces (power, mass, data rate), include a margin column in the DIM. For example, "150 W dissipation + 20% margin" rather than just "150 W." This prevents the margin erosion described in PP5.

Guideline 6: Colour Code by Status. While the DIM can be managed in any tool, status colours improve visibility: red for unresolved or high risk, yellow for in design or in test, green for verified. This allows quick identification of problematic interfaces during weekly reviews.

5.4.3.4 Traditional Interfacing vs. DIM

Traditional ICDs, while thorough, create excessive overhead for small teams. They become outdated quickly, impact analysis is slow and error prone, and integration conflicts are discovered late. The DIM prioritises a live, owned, query-able interface map that keeps pace with rapid design iteration.

Table 5.7: Comparison of Traditional Interface Management and DIM

Aspect	**Traditional Approach**	**DIM Approach**
Primary artifact	Formal Interface Control Document (ICD)	Living spreadsheet or tracked matrix
Update frequency	When formally revised (weeks or months)	Weekly or every sprint
Change impact analysis	Manual search through documents	Query-able, minutes to complete
Review cadence	At major gates only	Weekly review + gate baselines
Margin tracking	Buried in specifications or absent	Explicit column with margin percentage
Ownership	Often ambiguous	Single responsible owner per interface
Verification linkage	Separate verification plan	Verification method defined per interface

5.4.3.5 Working Example: SHERLOCK Mission

The following DIM captures critical interfaces for the SHERLOCK mission. Each interface is assigned a unique ID, owner, specification, status, and verification method.

Core DIM Table for SHERLOCK:

Table 5.8: Dynamic Interface Matrix for SHERLOCK Mission

ID	From Subsystem	To Subsystem	Type	Specification	Status	Verification	Owner
IF-P-001	Solar Panels	Power PDU	Electrical	28 V nominal, 6 A max, +20% margin	In Design	I-V characterization on solar simulator	Power Lead
IF-P-002	Battery	Power PDU	Electrical	30 Wh capacity, 60 Wh per orbit discharge	In Design	Cycle testing on battery cycler	Power Lead
IF-P-003	Power PDU	GNC	Electrical	28 V ±5%, 2 A continuous, 5 A peak	In Design	Bench test with electronic load	Power Lead
IF-P-004	Power PDU	OBDH	Electrical	5 V regulated, 1 A continuous	In Design	Bench test	Power Lead
IF-D-001	GNC	OBDH	Data	Attitude quaternion at 100 Hz via SPI	In Design	Protocol analyser test	GNC Lead
IF-D-002	Payload	OBDH	Data	Image data at 50 Mbps via CSI-2	In Design	Data integrity test	Payload Lead
IF-D-003	OBDH	Communications	Data	Telemetry and images at 256 kbps	In Design	Loopback test	Comms Lead
IF-M-001	Payload	Structure	Mechanical	M3 bolts, 2 Nm torque, ±1° alignment	In Design	Fit check and torque verification	Structure Lead

Change Impact Analysis Example for SHERLOCK:

Scenario: During detailed design, the Power Lead discovers that the OBDH subsystem actually requires 15 W of power (not the 10 W originally specified in IF-P-004). What is affected?

1. Query the DIM for all interfaces from Power PDU: IF-P-001, IF-P-002, IF-P-003, IF-P-004, IF-P-005, IF-T-001.
2. Identify affected subsystems: OBDH (direct), Thermal (dissipation increases from 10 W to 15 W), Power (total budget must be recalculated).
3. Identify affected requirements: L1-DE-0050 (mass may increase with larger battery), L2-PWR-0020 (fault protection margins).
4. Create change request: "Increase OBDH power allocation from 10 W to 15 W."
5. Update DIM: IF-P-004 specification changed to "15 W continuous." IF-T-002 updated to "15 W dissipation."
6. Hold 30 minute impact review with Power, OBDH, and Thermal leads.
7. Decision: Accept 15 W. Increase battery capacity from 30 Wh to 35 Wh. Radiator area increased by 10%.

Result: The DIM enabled identification of all affected interfaces within minutes. The change was reviewed and resolved in a single 30 minute meeting. Without the DIM, this mismatch would likely have been discovered during integration testing, requiring physical rework and a delay of several weeks.

DIM Review and Baseline for SHERLOCK:

The SHERLOCK DIM is reviewed weekly in a 30 minute meeting attended by all subsystem leads. Each lead reports status changes, new interfaces, and red flag items. At Gate 2 (Architecture Approval), the DIM is formally baselined. All 12 interfaces are approved, traceability to requirements is verified, and verification methods are agreed. The baselined DIM is version controlled and stored alongside other gate artifacts. Changes after Gate 2 require a Request for Change with documented impact analysis.

5.4.4 Risk-Based Verification Planning

5.4.4.1 Origin and Philosophy

Risk-Based Verification Planning (RVP) directly responds to three pain points identified in Chapter 2: excessive overhead (PP1), late verification (PP4), and investor driven milestone distortion (PP7) refer:2.2. Survey respondents reported that traditional verification planning is either too cumbersome to follow or too vague to be useful, refer: 4.19. It was evident that excessive verification documentation led to nothing but non value added activity.

Traditional INCOSE and NASA SE approaches require comprehensive verification planning upfront, with every requirement mapped to a verification method, test procedures written, test equipment identified, and schedule estimated [6]. For complex systems this is necessary, but for small, fast paced programmes it creates excessive overhead [23].

RVP replaces this with a proportional approach. Verification rigor is allocated based on risk. High risk, safety critical requirements receive rigorous verification through formal testing and flight validation. Low risk, well understood requirements are verified efficiently through analysis, simulation, or code review. This proportional approach emerged directly from survey findings, which showed that teams waste significant effort over verifying low risk items while under verifying high risk ones. RVP also addresses milestone distortion by making verification progress visible to stakeholders through dashboards, demonstrating technical readiness without requiring excessive documentation.

5.4.4.2 RVP Criteria

Every RVP must satisfy the following criteria to be considered valid and useful for a NewSpace team.

- **Proportional to Risk:** The verification effort for each requirement must be directly proportional to its risk category. Critical requirements receive full verification. Low risk requirements receive lightweight verification. No requirement is over verified or under verified.
- **Planned Upfront for Critical Items:** Verification methods for Critical and High risk requirements must be defined during the architecture phase (weeks 5 to 12). Low and Medium risk requirements may have verification methods defined during detailed design.
- **Traceable to Requirements:** Every verification activity must trace to at least one requirement. The traceability answers the question "Why are we running this test?" Unlinked verification activities are considered waste.

- **Status Tracked and Visible:** Verification status (Not Started, In Progress, Completed, Verified, Failed) must be tracked for every requirement and visible to the entire team through a shared dashboard.
- **Reviewed at Every Gate:** Verification status must be reviewed at each proportional review gate refer for details table: [5.20]. Critical requirements that are not verified or on schedule trigger escalation to programme management.

Each requirement´s verification decision is made using the Risk Matrix and the impact of the Risk decides the type of verification that is required for that case

Table 5.9: Risk Categories for Verification Planning

Risk Category	Consequence	Probability	Verification Approach	Effort
Critical	Mission failure, loss of life	Any	Full: Design review, analysis, bench test, flight validation	High
High	Major subsystem failure, mission degradation	Medium–High	Rigorous: Subsystem integration test, analysis, flight validation	Medium –High
Medium	Degraded performance, partial mission loss	Medium	Moderate: Design review, analysis, component test, flight data	Medium
Low	Non-critical, acceptable workaround exists	Low	Light: Code review, analysis, flight data	Low

5.4.4.3 Guidelines for RVP

The following operational guidelines derive directly from the RVP criteria.

Guideline 1: Assign Risk Categories Early.

During the concept phase, assign a preliminary risk category (Critical, High, Medium, or Low) to each Mission Requirement. Refine categories during the architecture phase for System Requirements. Do not wait until verification planning to assess risk.

Guideline 2: Use the Risk Matrix Consistently.

Risk category is a function of consequence severity and probability of occurrence. Use a standard risk matrix to ensure consistent categorisation across subsystems. A requirement that is Critical for one subsystem may be Medium for another.

Guideline 3: Match Verification Method to Risk Category.

Critical requirements require all four verification methods where feasible: design review, analysis, bench test, and flight validation. High requirements require design review, analysis, and component test. Medium requirements require analysis or simulation. Low requirements require code review or basic checks.

Guideline 4: Do Not Defer Critical Verification.

Verification of Critical requirements begins in the architecture phase, not during integration. Bench tests and analyses are conducted as soon as components are available. Late verification of critical items is a project risk in itself.

Guideline 5: Create a Verification Dashboard.

Maintain a simple dashboard showing total requirements by risk category and verification status percentage. This dashboard is shared with stakeholders and investors to demonstrate progress without producing lengthy reports.

Guideline 6: Escalate Red Flag Items Immediately.

Any Critical or High risk requirement that falls behind its verification schedule is escalated to the programme manager. Waiting until the next gate review is too late for corrective action.

5.4.4.4 Traditional V&V vs. RVP

Table 5.10: Comparison of Traditional Verification Planning and RVP

Aspect	**Traditional Approach**	**RVP Approach**
Verification effort	Uniform across all requirements	Proportional to risk category
Planning horizon	Comprehensive upfront plan	Critical items planned early; others as needed
Documentation	50+ page verification plan	Dashboard and tracked status
Test philosophy	Full test campaign for all requirements	Risk based test selection
Flight validation	After full qualification	Early flights as extended test campaigns
Stakeholder visibility	Formal test reports	Real time dashboard
Overhead	High (20-30% of effort)	Low (5-10% of effort)

Traditional verification planning, while thorough, consumes 20 to 30 percent of engineering effort on documentation and process compliance. RVP reduces this to 5 to 10 percent by focusing verification where it matters most: on high risk, safety critical requirements.

5.4.4.5 Working Example: SHERLOCK Mission

The following RVP table captures verification planning for SHERLOCK requirements, with verification methods allocated according to risk category.

Risk Assessment and Verification Planning for SHERLOCK:

Table 5.11: Risk Based Verification Planning for SHERLOCK Mission

Requirement ID	Requirement Summary	Risk Category	Verification Method	Responsibility
L0-OJ-0010	Inspect EU:CROPIS spacecraft	Critical	Flight data review after initial inspection pass	Mission Analyst
L0-PR-0040	Comply with debris mitigation standards	Critical	Regulatory review and compliance statement	Systems Engineer
L1-DE-0050	Mass not exceeding 4 kg	Critical	Inspection using calibrated scale at integration	Structure Engineer
L1-GNC-0010	Maintain safe distance from EU:CROPIS	Critical	HIL simulation + flight telemetry review	GNC Engineer
L2-PWR-0020	Triple-redundant fault protection	High	Fault injection bench test	Power Engineer
L2-GNC-0020	Collision avoidance within 30 minutes	High	Software-in-loop simulation	GNC Engineer

Detailed Verification Approach by Risk Category for SHERLOCK:

Critical Requirements (Red) – Full Verification:

For L1-GNC-0010 (maintain safe distance from EU:CROPIS):

- Design Review: Two hour formal review with GNC, OBDH, and Mission Analysis leads. Review collision avoidance algorithm, sensor selection, and thruster sizing.
- Analysis: Fault tree analysis for collision scenarios. Identify single point failures that could lead to distance violation.

- Bench Testing: Hardware-in-loop simulation with GNC computer, IMU, and simulated thruster responses. Run 100 collision avoidance scenarios.
- Flight Validation: First approach to EU:CROPIS is treated as extended test. Telemetry is monitored continuously. Distance data is recorded and compared to predictions.

High Requirements (Orange) – Rigorous Verification:

For L2-GNC-0020 (collision avoidance within 30 minutes):

- Design Review: One hour review with GNC and OBDH leads. Review conjunction detection timeline and maneuver execution logic.
- Analysis: Timing analysis showing worst case detection to maneuver completion is less than 30 minutes.
- Component Testing: Software-in-loop simulation with simulated conjunction alerts. Measure time from alert to maneuver command.
- Flight Data: On orbit conjunction events (if any) are logged and compared to simulation predictions.

Medium Requirements (Yellow) – Moderate Verification:

For L2-PWR-0020 (triple-redundant fault protection) – Note: This requirement is assessed as High risk for SHERLOCK due to the criticality of power to all subsystems.

Low Requirements (Green) – Lightweight Verification:

For lower risk requirements such as telemetry format or housekeeping data logging:

- Code Review: Peer review of telemetry encoding and decoding logic.
- Analysis: Simple consistency check confirming packet format matches ground station expectations.
- Flight Data: Confirm correct packets are received on ground during first pass.

Verification Status Dashboard for SHERLOCK (pre Gate 2):

Gate Verification Review for SHERLOCK (Gate 2 – Architecture Approval):

At Gate 2, the verification status is reviewed:

- All four Critical requirements have verification plans approved. Two are in progress (L1-DE-0050 mass measurement procedure drafted, L1-GNC-0010 HIL simulation test bench being set up). Two are not started but have clear owners and schedules.

Table 5.12: Verification Status Dashboard for SHERLOCK

Risk Category	**Total Requirements**	**Verified**	**In Progress**	**Not Started**
Critical	4	0	2	2
High	2	0	1	1
Medium	0	0	0	0
Low	0	0	0	0
Total	**6**	**0**	**3**	**3**

- Both High requirements have verification plans approved. One is in progress. One is not started but scheduled for next sprint.
- No red flag items. All Critical requirements are on schedule for verification completion before Gate 4 (Integration).
- Recommendation: Proceed to design phase with current verification plan.

The RVP for SHERLOCK ensures that the four Critical requirements receive full verification attention, while lower risk items are verified efficiently. The dashboard provides visibility to stakeholders without requiring lengthy reports. Verification status is reviewed at every gate, preventing late discovery of verification gaps.

5.4.5 Decision Capture Log

5.4.5.1 Origin and Philosophy

The Decision Capture Log (DCL) directly responds to two pain points identified in Chapter 2: knowledge loss (PP6) and requirements churn (PP3) refer: 2.2. Survey findings were striking: nearly 80 percent of respondents reported that their organisation had lost critical design knowledge due to team member departure refer: 4.8.

Traditional systems engineering relies on formal design documentation that is often written after the fact, missing the actual reasoning behind decisions. Alternatively, design rationale exists only in emails, Slack messages, or the memories of individual engineers. When those engineers leave, their knowledge leaves with them.

The DCL addresses this by providing a lightweight, structured format for logging decisions as they are made. Unlike traditional documentation that focuses only on major design choices, the DCL captures all decisions that affect the project: requirements changes, interface agreements, risk responses, supplier selections, schedule adjustments, and architectural

trade offs. Each entry captures what was decided, why it was decided, and what impacts it has. Only major decisions require the full detailed format. Minor decisions are captured in a condensed form.

5.4.5.2 DCL Criteria

Every Decision Capture Log entry must satisfy the following criteria to be considered valid and useful for a NewSpace team.

- **Decision Must Be Logged, Not Debated:**

  The DCL is a log, not a discussion forum. By the time an entry is created, the decision has already been made. The purpose is to capture, not to approve.

- **Rationale Must Be Brief but Clear:**

  Every entry must include a short explanation of why the decision was made. One or two sentences are sufficient for minor decisions. Major decisions require a paragraph.

- **Impact Must Be Stated:**

  Each entry must identify which requirements, subsystems, interfaces, or project artifacts are affected. For decisions with no impact, the entry can simply state "No impact."

- **Searchable and Linkable:**

  Every entry must have a unique ID and should link to related requirements, architecture elements, or other DCL entries. A decision that cannot be found is useless.

- **Categorised for Discoverability:**

  Each entry belongs to a category such as Requirements, Architecture, Interface, Risk, Schedule, Cost, Supplier, or Operations. Categories enable team members to quickly find relevant decisions.

5.4.5.3 Guidelines for DCL

The following operational guidelines derive directly from the DCL criteria.

Guideline 1: Log Decisions Immediately. The DCL entry is created within one working day of the decision. Do not wait for a review or a meeting. Memory fades quickly.

Guideline 2: Use Two Entry Formats – Light and Full. Minor decisions (e.g., "Use 28 V bus") use the light format: ID, date, decision, rationale, impact, category. Major decisions

(e.g., "Select hydrazine propulsion after six month trade study") use the full format with alternatives, schedule impact, cost impact, and links.

Guideline 3: Assign a Category to Every Entry. Categories include: Requirements, Architecture, Interface, Component Selection, Risk Response, Schedule, Cost, Supplier, Test, Operations, and Other. This makes filtering and searching easy.

Guideline 4: Assign a Single Owner. Each DCL entry has one owner who is accountable for its accuracy. That owner is typically the engineer or manager who made the decision.

Guideline 5: Do Not Duplicate Information. If a decision is already documented elsewhere (e.g., in a change request or meeting minutes), the DCL entry can simply link to that source rather than copying content.

Guideline 6: Review DCL Entries Weekly. During the weekly DIM review (or a separate 15 minute session), the team briefly reviews new DCL entries. Missing information is added immediately.

5.4.5.4 Traditional SE Case vs. DCL

Table 5.13: Comparison of Traditional Decision Documentation and DCL

Aspect	**Traditional Approach**	**DCL Approach**
What is captured	Major design decisions only	All decisions affecting the project
Format	Lengthy reports or meeting minutes	Lightweight structured entry
Rationale	Often missing or buried	Explicit and brief
Alternatives	Rarely documented	Required only for major decisions
Discoverability	Difficult, scattered across documents	Searchable by category, ID, or keyword
Timing	After the fact, often weeks later	Within one working day
Knowledge persistence	Lost when engineer leaves	Persists in structured, searchable log

Traditional decision documentation focuses on major choices and is often produced long after the decision is made. The DCL captures all decisions in real time, ensuring that rationale and context are never lost.

5.4.5.5 Working Example: SHERLOCK Mission

The following DCL entries capture various decisions made during the SHERLOCK mission development, using both light and full formats.

Light Format Entries (Minor Decisions):

Table 5.14: DCL Light Format Entries for SHERLOCK

ID	Date	Decision	Rationale	Impact	Category
DCL-001	2025-03-10	Use 28 V power bus	Standard in space, reduces copper weight	Power subsystem, all components	Architecture
DCL-002	2025-03-12	Deploy antenna after separation	Prevents damage during launch	Communications, deployment mechanism	Operations
DCL-003	2025-03-14	Use M6 bolts for payload mounting	Heritage from previous CubeSat	Structure, Payload	Interface
DCL-004	2025-03-18	Accept 15 W OBDH power (was 10 W)	Analysis showed 10 W insufficient	Power budget, thermal design	Requirements
DCL-005	2025-03-20	Select SpaceX as launch provider	Schedule certainty, cost within budget	Schedule, cost, orbit insertion	Supplier

Full Format Entry (Major Decision):

Table 5.15: DCL Full Format Entry for SHERLOCK

Field	**Content**
Decision ID	DCL-006
Date	2025-03-25
Owner	GNC Lead
Category	Component Selection
Decision	Select cold gas propulsion system using compressed nitrogen for station keeping and collision avoidance
Context	SHERLOCK requires delta V capability for station keeping near EU:CROPIS and collision avoidance maneuvers. Three options evaluated within 3U form factor constraints.
Rationale	Cold gas selected because: simplicity and reliability, lower safety handling requirements, adequate for required delta V of 5 m/s, off the shelf components at TRL 9, fits within 0.5 U volume allocation.
Alternatives Considered	Chemical monopropellant: higher Isp but significant safety and regulatory complexity, not justified. Electric propulsion: too large for 3U, requires high power.
Impact on Requirements	L1-GNC-0010 (maintain safe distance) – supported. L1-DE-0050 (mass 4 kg) – within budget.
Impact on Subsystems	GNC, Structure, Power.
Impact on Schedule	No impact.
Impact on Cost	5,000 Euros.
Design Risk	Low – flight heritage.
Links	L1-GNC-0010, L1-DE-0050, DIM IF-P-003

DCL Category Distribution for SHERLOCK (Weeks 1-4):

Table 5.16: DCL Category Distribution for SHERLOCK

Category	**Count**	**Example Decisions**
Architecture	3	28 V bus selection, redundant OBDH architecture, deployable vs fixed antenna
Requirements	2	OBDH power increase, mass budget adjustment
Interface	4	Payload mounting bolt specification, power connector type, data protocol selection
Component Selection	2	Cold gas propulsion (full format), camera sensor selection
Supplier	1	Launch provider selection
Operations	2	Antenna deployment timing, safe mode entry criteria
Schedule	1	Push integration week by 2 weeks due to part delay
Risk Response	1	Add second sun sensor for redundancy

The DCL for SHERLOCK ensures that every decision, from minor to major, is captured with rationale and impact. A new engineer joining the team can search by category to find relevant decisions. The light format keeps overhead low for routine decisions, while the full format captures necessary detail for major trade offs. The DCL is stored in a searchable database, reviewed weekly, and linked to requirements and interfaces.

5.4.6 Integration of MVSE

5.4.7 Integrating MVSE Components: The Traceability Web

The five MVSE components described in the previous sections do not operate in isolation. They form an integrated traceability web where each artifact links to the others, enabling rapid impact analysis, end to end traceability, and continuous visibility of project health.

5.4.7.1 The Traceability Web Architecture

The following diagram 5.2 illustrates how the five components connect to form a complete traceability chain from mission objectives to verification data.

5.4.7.2 Artifact Interconnections

Table 5.17 summarises how each artifact connects to the others and what queries each connection enables.

5.4.7.3 Traceability Queries in Practice

The traceability web enables rapid answers to critical project questions.

Query 1: Impact Analysis of Requirement Change

Scenario: The customer requests changing L0-MI-0010 (mission duration) from 1 month to 3 months.

1. Query MVR to find all System Requirements derived from L0-MI-0010.
2. Follow links from those System Requirements to AB functions and subsystems.
3. Follow links from AB to DIM interfaces affected by those subsystems.
4. Follow links from MVR to RVP to identify verification activities that must be repeated.
5. Follow links from MVR to DCL to find previous decisions about mission duration.
6. Result: Complete impact analysis in hours, not weeks.

Query 2: Root Cause Analysis of Verification Failure

Scenario: A test fails for L1-GNC-0010 (maintain safe distance from EU:CROPIS).

1. Query RVP to find the verification method and test procedure for L1-GNC-0010.
2. Follow links from L1-GNC-0010 to AB to identify which subsystems implement this requirement.

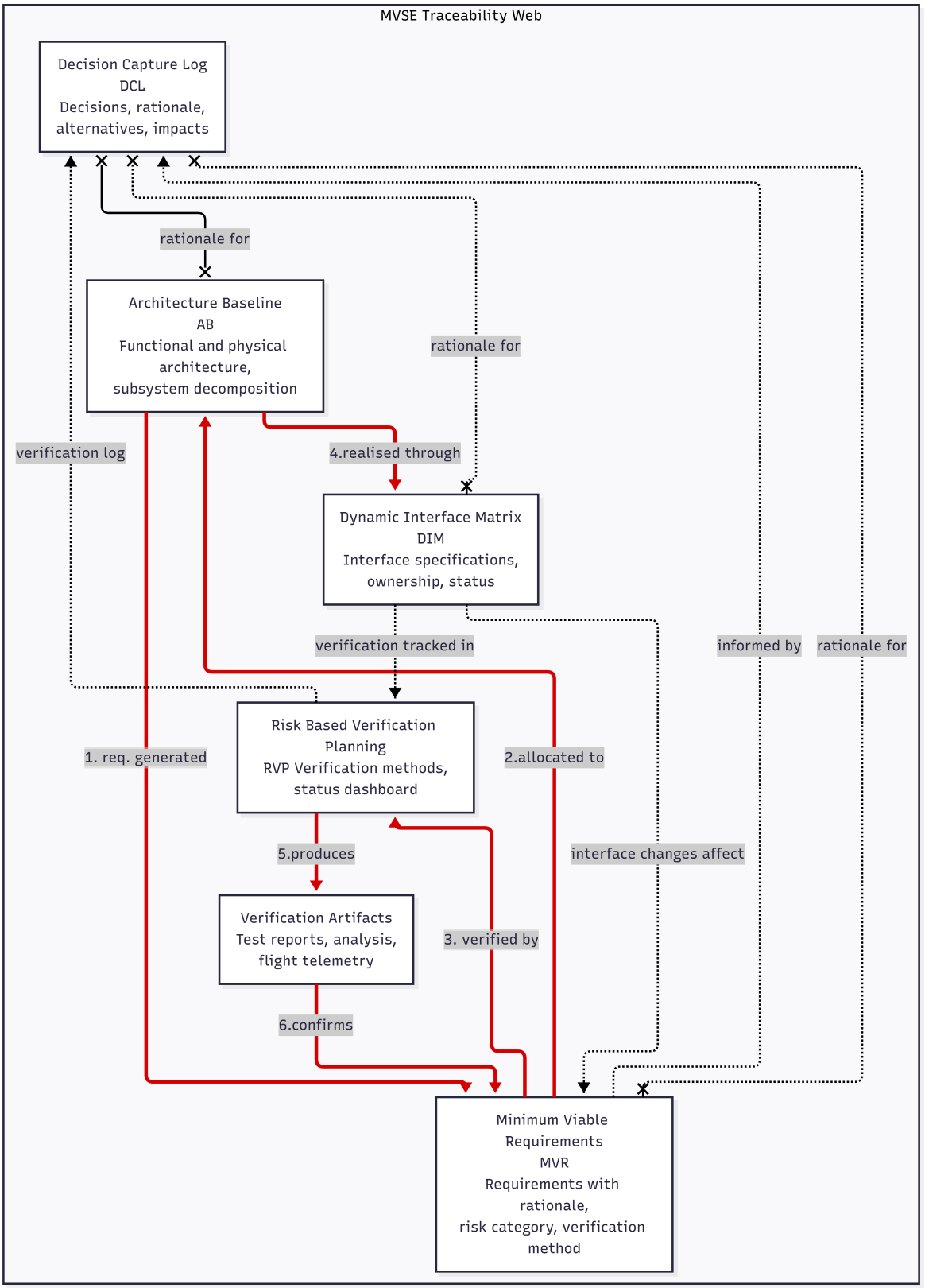

Figure 5.2: Traceability web architecture

Table 5.17: Artifact Interconnections and Traceability Queries

From Artifact	To Artifact	Link Type	Example Query	Benefit
MVR (Requirement)	AB (Function)	Allocation	"Which functions implement this requirement?"	Impact analysis of requirement change
MVR (Requirement)	AB (Subsystem)	Allocation	"Which subsystem is responsible for this requirement?"	Responsibility assignment
MVR (Requirement)	DIM (Interface)	Satisfaction	"Which interfaces must satisfy this requirement?"	Interface verification planning
MVR (Requirement)	RVP (Verification)	Verification	"How is this requirement verified?"	Test traceability
MVR (Requirement)	DCL (Decision)	Rationale	"Why was this requirement value chosen?"	Prevent requirement churn
AB (Subsystem)	DIM (Interface)	Realisation	"Which interfaces belong to this subsystem?"	Change impact analysis
AB (Function)	DCL (Decision)	Rationale	"Why was this function allocated to this subsystem?"	Architecture rationale
DIM (Interface)	MVR (Requirement)	Satisfaction	"Which requirements constrain this interface?"	Interface change impact
DIM (Interface)	DCL (Decision)	Rationale	"Why was this interface specification chosen?"	Interface design rationale
RVP (Verification)	MVR (Requirement)	Coverage	"Which requirements remain unverified?"	Verification gap analysis
RVP (Verification)	VER (Artifacts)	Evidence	"What test report verifies this requirement?"	Audit trail
DCL (Decision)	MVR (Requirement)	Affects	"Which requirements are affected by this decision?"	Decision impact analysis
DCL (Decision)	AB (Subsystem)	Affects	"Which subsystems are affected by this decision?"	Cross subsystem impact
DCL (Decision)	DIM (Interface)	Affects	"Which interfaces are affected by this decision?"	Interface change impact

3. Follow links from AB to DIM to identify interfaces involved in distance maintenance.
4. Follow links from DIM to DCL to find design decisions affecting those interfaces.
5. Result: Root cause identified through traceability, not guesswork.

Query 3: Interface Change Impact Analysis

Scenario: The Power Lead proposes changing IF-P-003 specification from 28 V to 12 V.

1. Query DIM for all interfaces from Power PDU (IF-P-003, IF-P-004, IF-P-005).
2. Follow links from DIM to MVR to find requirements affected (L1-DE-0050 mass, L2-PWR-0020 fault protection).
3. Follow links from DIM to AB to find subsystems affected (GNC, OBDH, Communications).
4. Follow links from MVR to RVP to identify verification activities that must be updated.
5. Follow links from DIM to DCL to find previous decisions about power bus voltage.
6. Result: Change approved or rejected with full awareness of consequences.

5.4.7.4 Artifact Update Cadence

Table 5.18 specifies how frequently each artifact should be updated during different project phases.

Table 5.18: Artifact Update Cadence by Project Phase

Artifact	**Concept Phase**	**Architecture Phase**	**Design Phase**	**Integration Phase**	**Launch Phase**
MVR - L0 Requirements	Weekly (create)	Frozen	Frozen	Frozen	Frozen
MVR - L1 Requirements	Not yet	Weekly (create)	Monthly (refine)	Frozen	Frozen
MVR - L2 Requirements	Not yet	Not yet	Weekly (capture)	As needed	As needed
AB - Functional	Weekly	Weekly	Monthly	Frozen	Frozen
AB - Physical	Not yet	Weekly	Weekly	Monthly	Frozen
DIM	Not yet	Weekly	Weekly	Weekly (critical)	Frozen
RVP - Planning	Weekly	Weekly	Monthly	As needed	As needed
RVP - Status	Not yet	Not yet	Weekly	Weekly	Daily
DCL	As decisions made	As decisions made	As decisions made	As decisions made	Lessons learned

5.4.7.5 Artifact Dependencies and Workflow

Figure 5.3 shows the workflow for creating and maintaining MVSE artifacts, including dependencies and handoffs between artifacts.

5.4.7.6 Development Timeline with MVSE Artifacts

Figure 5.4 shows how the five MVSE artifacts are created, maintained, and baselined throughout a typical 18 month NewSpace project.

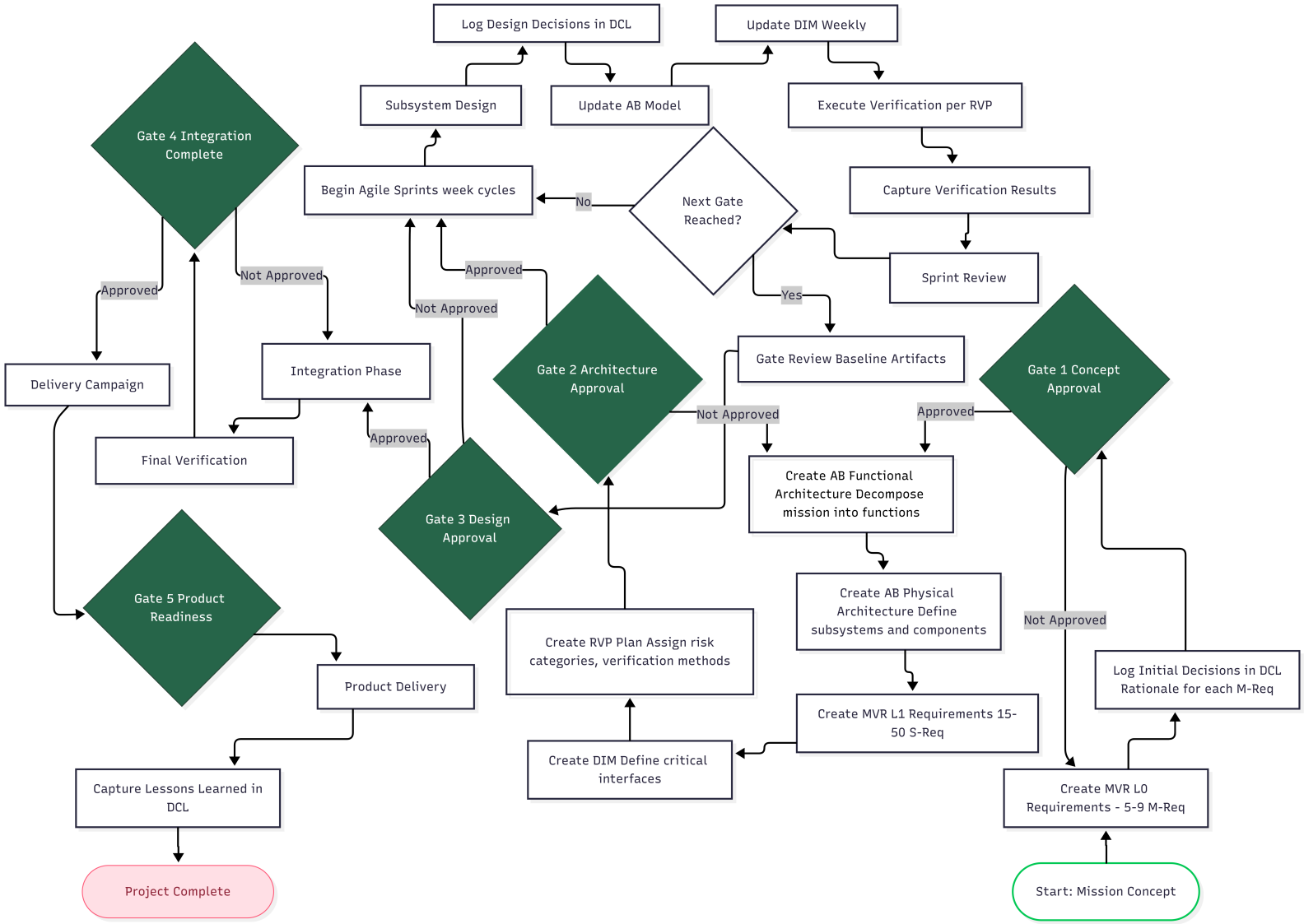

Figure 5.3: MVSE Process Flow

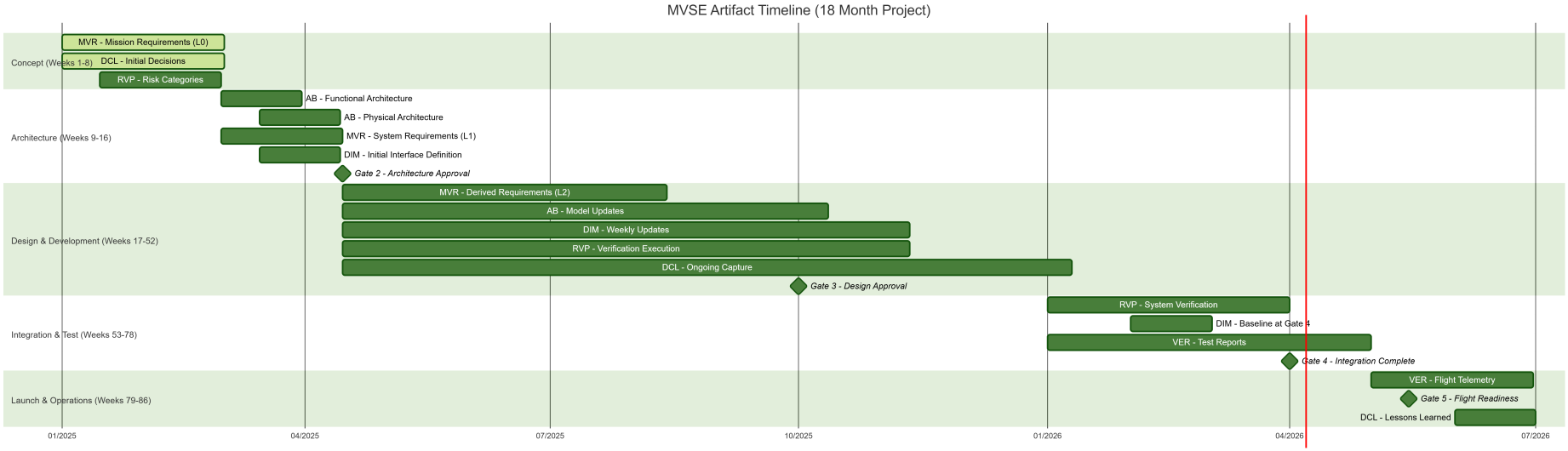

Figure 5.4: An example of a MVSE Project Timeline

5.4.7.7 Traceability Dashboard

A traceability dashboard provides real time visibility into the health of the traceability web. Table 5.19 shows key metrics to track.

Table 5.19: Traceability Dashboard Metrics

Metric	Target	Red Flag	Action if Red Flag
Requirements linked to AB	100%	< 90%	Review unlinked requirements, assign to subsystem
Requirements with verification method	100% for Critical/High	< 95%	Complete RVP planning for missing items
DIM interfaces with owner assigned	100%	< 80%	Assign owners to unowned interfaces
DIM interfaces with verification method	100% for Critical	< 90%	Define verification for each interface
Critical requirements verified	100% before Gate 4	< 80% before Gate 3	Accelerate verification activities
DCL entries linked to requirements	> 80%	< 50%	Review unlinked decisions, add links
Requirements with no DCL rationale	< 10%	> 30%	Create DCL entries for missing rationale

5.4.7.8 Summary of Integration Benefits

The integrated traceability web provides the following benefits to NewSpace teams:

- **End to End Traceability:** Every requirement traces to architecture, interfaces, verification, and decisions. No orphaned requirements or untested features.
- **Rapid Impact Analysis:** A change to any artifact immediately shows affected artifacts through linked queries. Impact analysis that takes weeks in document centric environments takes hours.
- **Continuous Visibility:** The dashboard shows verification progress, requirement coverage, and interface status in real time. No waiting for monthly reports.
- **Knowledge Persistence:** Every decision is captured and linked. When an engineer leaves, their rationale remains accessible to the team.
- **Audit Readiness:** The traceability web provides a complete chain of evidence from mission objectives to verification results. Regulatory audits are supported without

additional documentation effort.

- **Reduced Rework:** Impact analysis catches ripple effects before they cause integration failures. Decisions are revisited only when context changes, not because rationale was lost.

The SHERLOCK mission example throughout this chapter demonstrates how these five components work together in practice. Chapter 6 presents a detailed account of the MVSE workshop conducted using the SHERLOCK mission, including participant feedback and lessons learned.

5.4.8 Other Features of MVSE

In addition to the five core components (MVR, AB, DIM, RVP, DCL), the MVSE framework includes three supporting features that enable effective execution of NewSpace programmes: proportional review gates, lightweight budget tracking, and integration with agile practices.

5.4.8.1 Proportional Review Gates

Traditional aerospace programmes use formal gates such as System Requirements Review, Preliminary Design Review, Critical Design Review, and System Verification Review. Each gate involves 50 to 100 participants, 50 to 200 page review packages, and 2 to 5 days of formal proceedings [6]. For NewSpace teams, this overhead is excessive.

MVSE defines five lightweight review gates proportioned to programme scale and risk.

Table 5.20: MVSE Proportional Review Gates

Gate	**Focus**	**Duration**	**Key Approval Criteria**
Gate 1	Concept	2 hours	Mission requirements approved; critical requirements identified; risk assessment complete
Gate 2	Architecture	4 hours	System architecture complete; interface matrix reviewed; verification plan documented
Gate 3	Design	4 to 8 hours	Design 100% complete; verification plan detailed; manufacturing ready
Gate 4	Integration	4 hours	Integration testing complete; critical and high requirements verified
Gate 5	Flight Readiness	8 hours	All requirements verified; regulatory approval obtained; launch authorized

Gate 1 (Concept) Approval Criteria:

- Mission requirements complete and approved by customer
- Critical system requirements identified and rationale captured
- Top level functional architecture defined
- Preliminary risk assessment completed
- High risk items identified and mitigation strategy outlined
- Schedule and budget baseline established
- Team assembled and roles assigned

Gate 2 (Architecture) Approval Criteria:

- System architecture complete (functional and physical)
- All critical and high risk requirements allocated to subsystems
- Interface matrix complete and reviewed
- Verification plan for critical requirements documented
- Subsystem budgets (mass, power, cost) allocated and verified feasible
- Design rationale captured in DCL for major decisions
- Integration risks identified

Gate 3 (Design) Approval Criteria:

- Design 100 percent complete (CAD, schematics, code, procedures)
- Design reviews completed for all subsystems
- Verification plan detailed (test procedures, schedule, responsibility)
- Integration activities planned and scheduled
- Manufacturing readiness confirmed (suppliers can deliver on schedule)
- Environmental test plan defined
- Outstanding design risks resolved or mitigated

Gate 4 (Integration) Approval Criteria:

- System integration testing complete

- All critical and high risk requirements verified (tested or analysed)
- Flight data from early vehicles validates design (if applicable)
- Medium risk requirements verified or risk accepted
- Outstanding issues documented with closure plans
- Flight procedures and operations training ready
- Customer final approval obtained

Gate 5 (Flight Readiness) Approval Criteria:

- All requirements verified
- Regulatory approval obtained (FAA, FCC, ITAR, etc.)
- Flight permit issued
- Final flight readiness review completed
- Launch authorization granted

Verification Timeline Across Phases:

Table 5.21: Verification Activities by Project Phase

Phase	**Verification Activities**
Concept Phase	Identify critical requirements. Define rough verification approach for critical requirements (bench test, flight validation, etc.). Estimate verification effort.
Architecture Phase	Develop detailed verification plan for critical and high risk requirements. Establish design review schedule for each subsystem. Draft test procedures for critical tests.
Design Phase	Execute verification activities per plan. Update verification status. Capture verification results (data, photos, reports).
Integration Phase	Conduct subsystem integration testing. Perform full system testing (SIL and HIL). Execute environmental testing (thermal vacuum, vibration, EMC).
Pre Flight	Complete final verification review confirming all critical requirements verified. Close any verification gaps.

Gate Execution:

One week before a gate, the Systems Engineering lead compiles a gate package including requirements status, architecture diagrams, DIM snapshot, DCL summary, and verification

dashboard. Subsystem leads review their sections and raise open issues.

The gate meeting lasts 2 to 8 hours depending on the gate. It includes a programme status presentation, architecture review, risk review, verification review, and decision discussion. The outcome is recorded as Approved, Approved with Conditions, Conditional Approval, or Not Approved.

Between Gate Agile Operations:

- Between gates, the programme operates in agile mode with two week sprints, daily standups, continuous verification, and lightweight change control.
- Sprints include sprint planning, daily standups, sprint reviews, and retrospectives.
- Daily communication focuses on integration risks and blocker resolution.
- Tests run as subsystems are ready, and early results inform design refinement.

5.4.8.2 Budget Tracking

Traditional space programmes manage budgets through centralised spreadsheets that are updated infrequently. When changes occur, margin erosion goes unnoticed until integration. MVSE introduces four practices for lightweight budget tracking.

Distributed Budget Ownership:

Each subsystem team owns its own local budget envelope, defined, updated, and tracked throughout design sprints. For example, the avionics team owns the power draw budget. The propulsion team owns thrust margin, valve mass, and thermal envelope.

Budget Snapshots per Sprint:

A shared system budget board aggregates the latest values at the beginning of each sprint. For example, a mass budget spreadsheet contains the latest dry mass values from structure, propulsion, and avionics. The board is stored in a collaborative tool such as Notion, Confluence, or Excel with version control.

Cross-Team Constraint Traceability:

Each local budget entry links to its requirement ID, test source, and affected interfaces. This enables propagation: a change in avionics power draw updates both the power budget and informs propulsion thermal load.

Triggered Budget Audits:

When cumulative margin loss exceeds a threshold, for example a 15 percent power draw

increase, a focused audit is scheduled. The audit reviews design changes, test results, and failure modes. This prevents margin stack up and last minute integration shocks.

Implementation Steps:

1. Define high level system budgets such as mass, power, and thermal limits.
2. Allocate budgets per subsystem. For example, GNC receives 60 watts continuous and 2.5 kilograms. Avionics receives 120 watts and 4.0 kilograms.
3. Link each budget to design and verification. Each team tracks usage against allocation and verifies performance during test.
4. Propagate updates when a team exceeds margin. Changes are discussed in system review and impacts on other subsystems are assessed collaboratively.

5.4.8.3 Integration with Agile Practices

MVSE is designed to work seamlessly with agile development methods, not against them.

Sprint Planning:

MVSE requirements from MVR are decomposed into deliverable user stories. A Mission Requirement becomes an epic. System Requirements become stories. Derived Requirements become tasks. For example, an epic for mission duration breaks down into stories for battery design, algorithm specification, ground testing, and flight validation.

A typical two week sprint allocates 50 to 60 percent of capacity to design work, 20 to 30 percent to implementation, 10 to 20 percent to verification, and 5 to 10 percent to technical debt such as DCL updates and documentation.

Daily Standup:

The daily standup lasts 15 minutes and covers four topics:

- **Blockers:** What is preventing progress? For example, "Waiting for interface data from supplier."
- **Integration Risks:** Are any design decisions affecting other subsystems? Escalate to DIM review when needed.
- **Verification Progress:** Are tests on schedule?
- **Sprint Progress:** Will the team hit the sprint goal?

During the standup, the team updates task status, logs key technical decisions in the DCL, and flags emerging interface constraints immediately to prevent rework.

Sprint Review:

The sprint review lasts 60 minutes and includes:

- **Demonstration:** Show completed user stories. Present architecture progress, design work, and verification results.
- **Metrics:** Review burndown chart, velocity, and requirement coverage.
- **Risk and Issue Review:** Discuss new technical risks identified during the sprint. Review interface conflicts and how they were resolved. Assess if completed stories suggest requirement changes.
- **Stakeholder Feedback:** Confirm progress aligns with customer expectations. Clarify any requirement changes.
- **Next Sprint Preview:** Outline focus for next sprint and resolve blockers between sprints.

Sprint Retrospective:

The retrospective lasts 60 minutes and asks four questions:

- What went well? Which stories were completed smoothly? What tools or processes helped?
- What did not go well? Which stories slipped? Were there unexpected challenges? Did DCL miss any decisions?
- What process improvements should be made? Should sprint length, standup timing, or review format be adjusted?
- What action items will be implemented? Assign improvements as tasks for the next sprint.

Continuous Verification:

Unlike traditional programmes that defer verification to a dedicated test phase, MVSE runs tests as subsystems are ready. Early results inform design refinement. This test early, test often approach prevents late discovery of integration failures.

5.4.8.4 Request for Change Process

MVSE uses a lightweight Request for Change (RFC) process for managing changes to requirements, architecture, interfaces, or budgets. The process is documented in the DCL to ensure rationale is never lost.

When to Create an RFC:

An RFC is required for any change that affects:

- A baselined requirement (especially Mission Requirements frozen at Gate 1)
- A baselined interface in the DIM
- A subsystem budget allocation (mass, power, cost)
- The system architecture or functional allocation
- The schedule or cost baseline

Routine design decisions that do not affect baselined artifacts do not require an RFC, but should still be logged in the DCL using the light format.

RFC Workflow:

1. **Initiation:** The proposer creates an RFC entry in the DCL with the following fields:
    - RFC ID (unique identifier)
    - Date of proposal
    - Proposer name and subsystem
    - Change description (what is being changed)
    - Rationale (why the change is needed)
    - Alternatives considered (what other options were evaluated)
2. **Impact Analysis:** The Systems Engineering lead performs impact analysis using the traceability web:
    - Query the DIM to identify all interfaces affected by the change
    - Query MVR to identify requirements affected
    - Query AB to identify subsystems and functions affected
    - Estimate schedule and cost impact
    - Document impact analysis in the DCL entry
3. **Review and Decision:**
    - Minor changes (impact less than 5 percent of budget or 1 week of schedule): Approved by programme manager within 3 days.

- Major changes (impact greater than 5 percent of budget or 1 week of schedule): Reviewed at the next gate or emergency review. Decision made by programme manager with subsystem leads.
- Critical changes (affects mission safety or success): Requires customer approval.

4. **Implementation:** Once approved:
   - Update affected artifacts (MVR, AB, DIM, RVP) with the change
   - Update the DCL entry with approval date and decision maker
   - Communicate the change to all affected subsystem teams
   - Adjust sprint plans if schedule impact exists
5. **Closure:** The RFC is marked as Closed in the DCL. The decision rationale remains searchable for future reference.

RFC Example for SHERLOCK Mission:

Table 5.22: RFC Example for SHERLOCK Mission

Field	**Content**
RFC ID	RFC-SHERLOCK-001
Date	2025-04-15
Proposer	Power Lead
Change Description	Increase OBDH power allocation from 10 W to 15 W
Rationale	Detailed design analysis showed 10 W insufficient for peak processing loads
Alternatives Considered	Optimise code to reduce power (insufficient margin); Upgrade to more efficient processor (longer lead time)
Impact Analysis	Affects IF-P-004 (Power to OBDH interface). Affects IF-T-002 (OBDH thermal dissipation). Affects L1-DE-0050 (mass may increase with larger battery). Affects L2-PWR-0020 (fault protection margins).
Schedule Impact	No impact
Cost Impact	1,000 Euros for larger battery
Decision	Approved by programme manager on 2025-04-17
Closure	DIM updated. DCL entry linked to affected requirements. Team notified.

5.4.8.5 Summary of Supporting Features

Table 5.23 summarises how each supporting feature addresses the pain points identified in Chapter 2.

Table 5.23: MVSE Supporting Features and Addressed Pain Points

Feature	**Purpose**	**Pain Points Addressed**
Proportional Review Gates	Replace heavy phase gate reviews with lightweight, risk appropriate gates	PP2 (phase gate rigidity), PP1 (excessive overhead)
Budget Tracking	Distributed ownership with sprint level snapshots and triggered audits	PP5 (informal margins), PP8 (tool fragmentation)
Agile Integration	Seamless work with sprints, standups, and continuous verification	PP2 (rigidity), PP4 (late verification), PP7 (milestone distortion)
Request for Change	Lightweight change control with full traceability and rationale capture	PP3 (requirements churn), PP6 (knowledge loss)

5.4.8.6 Positioning MVSE Against Classical Frameworks

Having detailed each MVSE component, Table 5.24 below summarises how the framework differs from the classical SE approaches analysed in Chapter 2. This comparison highlights the specific design choices that make MVSE suitable for NewSpace startups.

Table 5.24: Comparison of Classical SE Frameworks and MVSE

Aspect	**Classical SE (INCOSE, NASA, ECSS)**	**MVSE**
Target audience	Large programmes (100+ engineers, 5-10 years)	NewSpace startups (20-150 engineers, 12-36 months)
Requirements	Hundreds to thousands, frozen at baseline	20-65 active, 7 ± 2 per level, mission requirements frozen only
Reviews	Phase-gate with 50-200 participants, days-long	5 proportional gates, 2-8 hours, 8-30 participants
Documentation	Document-centric, extensive formal documents	Model-centric, minimal mandatory artefacts
Verification	Comprehensive plan for every requirement	Risk-based, proportional to risk category
Traceability	Manual matrices, labour-intensive	Digital-first, automated links, queryable
Change control	Formal CCB, weeks to approve	Lightweight RFC, days to approve, logged in DCL
Tailoring	Requires active tailoring effort, no pre-configured profile	Built-in risk-proportional governance
Knowledge management	Relies on documentation, often lost	Decision Capture Log (DCL) with rationale and alternatives
Tooling	Often expensive specialised tools	Tool-agnostic, works with affordable tools
Intended use	Crewed missions, flagship science, high regulatory oversight	Uncrewed missions, technology demonstration, rapid iteration

5.5 Summary of MVSE Framework

Minimum Viable Systems Engineering (MVSE) is proposed as a hybrid SE methodology that reconciles the tension between speed and rigour. MVSE retains only the essential artefacts required for mission assurance while systematically eliminating superfluous processes. The framework defines a minimal artefact set comprising five core elements.

- **Minimum Viable Requirements (MVR)** are concise, testable, criterion-based statements that capture functional and non-functional needs without the elaborate requirement specifications typical of classical SE.
- **Lightweight Architecture Baseline (AB)** is established using a simplified notation that captures functional decomposition and key subsystem interfaces, without exhaustive architectural documentation.
- **Dynamic Interface Mapping (DIM)** provides a continuously updated, collaborative map of subsystem boundaries and interaction points, enabling rapid traceability and change management.
- **Risk-Based Verification Plan (RVP)** links verification activities explicitly to quantified risk thresholds, ensuring that verification effort is proportional to risk exposure rather than driven by compliance checklists.
- **Decision-Capture Log (DCL)** serves as a structured repository of design decisions, lessons learned, failure modes, and change rationales, thereby enabling organisational learning and reducing rework in subsequent projects.

5.5.1 MVSE Substitution Mapping: What Replaces What

Table 5.25 maps classical SE artefacts and practices to their MVSE equivalents. It clarifies what is retained (in simplified form), what is discarded, and what is newly introduced.

Table 5.25: Classical SE Artefacts and Their MVSE Equivalents

Classical SE Artefact / Practice	**MVSE Equivalent**	**Change Type**	**Description of Change**
Requirements Specification (500+ requirements)	Minimum Viable Requirements (MVR, 20-65 requirements)	**Reduced**	Only essential requirements captured; 7 ± 2 per level; deferred detailed requirements
Formal Phase Reviews (SRR, PDR, CDR, SVR; 50-200 participants, days)	Proportional Review Gates (5 gates; 8-30 participants, 2-8 hours)	**Streamlined**	Risk-appropriate formality; event-driven rather than phase-driven
Interface Control Document (ICD; static, 100+ pages)	Dynamic Interface Matrix (DIM; living spreadsheet)	**Simplified**	Updated weekly; enables rapid impact analysis; explicit ownership
Verification Plan + Test Procedures + Test Reports (uniform for all requirements)	Risk-Based Verification Plan (RVP; proportional to risk category)	**Restructured**	Critical requirements: full verification; Low risk: lightweight verification
Architecture Description (lengthy documents)	Architecture Baseline (AB; model-centric)	**Transformed**	Single source of truth; queryable; version-controlled
Design Decision Records (rarely used, unstructured)	Decision Capture Log (DCL; mandatory, structured)	**Added**	Every decision captured with rationale, alternatives, impacts; searchable
Traceability Matrices (manual)	Traceability Web (automated between MVR, AB, DIM, RVP, DCL)	**Automated**	End-to-end traceability via artifact links; queryable
Configuration Control Board (CCB; weeks to approve changes)	Request for Change (RFC; lightweight, documented in DCL)	**Simplified**	Days to approve; impact analysis via DIM query
Documentation-heavy tailoring process	Built-in risk-proportional governance	**Eliminated**	Tailoring is inherent to the framework, not a separate activity
Manual margin tracking (scattered spreadsheets)	Budget tracking (sprint-level snapshots, triggered audits)	**Added**	Distributed ownership; real-time visibility; margin erosion prevention

The table illustrates three types of changes:

- **Reduced / Simplified:** Requirements, reviews, interfaces, and change control are scaled down to what is essential for small teams. The ceremony is removed while the essential purpose is retained.
- **Transformed / Restructured:** Architecture and verification shift from document-centric to model-centric and risk-proportional. The fundamental purpose remains, but the form changes dramatically.
- **Added / Automated:** Decision capture (DCL), budget tracking, and automated traceability are new or significantly enhanced relative to typical NewSpace practice. These address the knowledge loss and tool fragmentation pain points identified in Chapter 2.
- **Eliminated:** The separate tailoring process is eliminated because MVSE embeds risk-proportional governance directly into its structure. Teams do not need to tailor the framework; they simply apply its built-in rules.

6 Implementation and Future Scope

This chapter describes the practical application of the Minimum Viable Systems Engineering (MVSE) framework through a pilot workshop based on the SHERLOCK mission. It presents the implementation setup, the workshop execution, a full worked example from the Notion workspace, participant feedback, and recommendations for future adoption and research.

6.1 Pilot Testing: SHERLOCK Mission Workshop

To validate the MVSE framework, a pilot workshop was conducted using the SHERLOCK mission as a test case. SHERLOCK is a student designed 3U CubeSat mission with the primary objective of inspecting the EU:CROPIS spacecraft in low Earth orbit. The mission has a compressed development timeline of 18 months and a team of approximately 25 engineers, making it representative of the NewSpace context for which MVSE is designed.

6.1.1 Workshop Setup

The workshop was designed to simulate a real NewSpace development environment. Participants were assigned specific roles based on the work instructions provided in advance. The Participants were Systems Engineers from the CEF facility of DLR Bremen.

Workshop Environment:

- **Platform:** A dedicated Notion workspace was created to host all MVSE artifacts. Notion was selected for its collaborative features, searchability, and ease of use without specialised training.
- **Duration:** The workshop ran for approximately 3.5 hours, following the agenda outlined in Appendix A.
- **Participants:** The workshop included Systems Engineers, subsystem leads (GNC, Power, Structures, OBC, Communications, Payload), and a Risk Manager.
- **Input Materials:** Participants were provided with the SHERLOCK mission requirements (L0, L1, L2), the risk matrix, and the MVSE work instructions.

Workshop Agenda Summary:

Table 6.1: Workshop Agenda for SHERLOCK MVSE Pilot

Time	**Activity**
09:30 - 09:45	Introduction, purpose, and workshop instructions
09:45 - 10:00	Mission scenario presentation (SHERLOCK)
10:00 - 10:15	Set up and tool familiarisation
10:15 - 10:45	Round 1: MVR and DCL
10:45 - 11:00	Break
11:00 - 12:00	Round 2: DIM for domain roles; RVP for SE and Risk roles
12:00 - 12:30	Group reflection on usability, clarity, and effectiveness
12:30 - 12:50	Feedback collection through questionnaire
12:50 - 13:00	Wrap up

6.1.2 Workshop Execution

Participants worked through the MVSE components sequentially, populating the Notion workspace with artifacts for the SHERLOCK mission.

Round 1: Minimum Viable Requirements (MVR) and Decision Capture Log (DCL)

Participants reviewed the existing requirement set in the provided requirements spreadsheet. Systems Engineers identified duplicate requirements, ambiguous wording, and non testable statements. Subsystem leads analysed requirements affecting their subsystems, identifying missing constraints and unrealistic assumptions.

Key outcomes from this round:

- The initial requirement set was reduced from approximately 35 requirements to 18 core requirements following the 7 ± 2 principle per level.
- Each retained requirement was assigned a risk category (Critical, High, Medium, Low).
- Design decisions made during requirement analysis were logged in the DCL captured with rationale, alternatives, and impact.

Round 2: Dynamic Interface Matrix (DIM) and Risk Based Verification Planning (RVP)

In the second round, subsystem leads defined interfaces for their domains using the DIM

template. They identified inputs, outputs, and operational dependencies, then mapped requirements with full traceability.

Systems Engineers and the Risk Manager focused on RVP. They linked risks from the risk matrix to relevant requirements, selected appropriate risk classifications, and ensured high risk items received verification coverage.

Key outcomes from this round:

- A DIM containing 12 critical interfaces was created, covering electrical, data, mechanical, and thermal interfaces.
- Each interface was assigned an owner, specification, status, and verification method.
- A verification dashboard was established showing status by risk category.

6.1.3 Notion Workspace and References

The complete MVSE implementation for the SHERLOCK mission is available in the public Notion workspace created for this thesis.

Access Link:

The Notion workspace can be accessed at: `https://www.notion.so/Minimum-Viable-Systems-Engineering-MVSE-Workshop-32c7b1ecb42d80fdad8ec2d026fa20f3?source=copy_link`

Workshop Materials in Appendix:

The following workshop materials are included in Appendix C:

1. Workshop Agenda
2. Work Instructions for each role (Systems Engineer, Subsystem Leads, Risk Manager)

6.2 Future Scope

The pilot testing of MVSE on the SHERLOCK mission has demonstrated the framework's potential, but several areas remain for future research and development.

6.2.1 Potential Use Cases:

Paotential applications beyond the SHERLOCK mission:

- Other CubeSat and small satellite programmes

- Technology demonstration missions
- Student and educational space projects
- Internal research and development projects in larger aerospace organisations

6.2.2 Longitudinal Validation

The current validation is based on a single workshop. Future work should conduct longitudinal studies tracking NewSpace projects from concept to launch using MVSE. Such studies would provide quantitative data on:

- Actual reduction in non value added effort compared to traditional SE
- Impact on integration failure rates and rework cycles
- Team onboarding time and knowledge retention metrics
- Correlation between MVSE adoption and mission success rates

6.2.3 Tool Development

While the Notion based implementation proved effective for the pilot, dedicated tool support could further reduce overhead. I intend to explore this area in the near future. Potential development areas include:

- A lightweight MVSE specific tool with native traceability between MVR, DIM, DCL, and RVP
- Automated traceability checking and impact analysis
- Integration with existing engineering tools (CAD, simulation, test stands)
- Open source reference implementation to lower adoption barriers

6.2.4 Extension to Larger Programmes

MVSE was designed for teams of 20 to 150 people. Future research should investigate how the framework scales to larger programmes, including:

- Multi team coordination with shared artifacts
- Supplier integration and interface management across organisational boundaries
- Integration with formal standards (INCOSE, NASA, ECSS) for regulatory compliance
- Transition pathways from MVSE to full SE as programmes grow

6.2.5 Industry Adoption and Standardisation

To support broader industry adoption, future work should develop:

- Training materials and certification programmes for MVSE practitioners
- Case studies from multiple NewSpace organisations across different mission types
- Guidance for tailoring MVSE to specific mission classes (launch vehicles, satellites, in space infrastructure)
- Potential incorporation of MVSE principles into future INCOSE or ECSS updates

6.2.6 Integration with Emerging Technologies

MVSE can potentially integrate with emerging technologies to further reduce overhead:

- **AI Assisted Requirements:** Large language models could help identify ambiguous requirements, suggest risk categories, and generate verification methods.
- **Automated Interface Detection:** Tools that parse CAD models or schematics to automatically populate the DIM.
- **Natural Language to Traceability:** Voice or text interfaces for logging DCL entries during design meetings.

7 Conclusion

This thesis seeks to balance two opposing trends in the contemporary era of space development: ensuring the rigor, traceability, and safety of systems engineering (SE) activities, and working under the fast pace and limited resource constraints unique to NewSpace organizations. The introductory chapter has clearly stated the research aim: developing and assessing an approach called Minimum Viable Systems Engineering (MVSE) which allows for maintaining the necessary level of technical assurance and traceability despite the need for frequent iterations and changes characteristic for NewSpace ventures. The purpose of the concluding chapter is to confirm that the research aim was achieved, as well as summarize the fulfillment of research objectives and reflect on contributions and limitations.

7.1 Achievement of Research Aim and Objectives

The objectives defined in Chapter 1 were met throughout this work in the following sequence.

Objective 1:

Evaluate the shortcomings of existing SE approaches in fast-paced, hardware-oriented projects.
Chapter 2 provided a thorough evaluation of the four leading SE standards – INCOSE, NASA, ECSS, and ISO guidelines – which identified eight core pain points: excessive overhead, inflexible phase-gate process, instability, late verification, informal margins, knowledge loss, milestone-focused schedule, and fragmented tools. Chapter 4 empirically confirmed those pain points.

Objective 2:

Gather empirical data from surveys of SE practitioners involved in space projects.
Chapter 3 described a mixed methods survey instrument, targeted at European NewSpace practitioners (total number of respondents = 44). Chapter 4 contains the detailed description of the results obtained with that survey: process overhead, tailoring practices, toolchain fragmentation, verification problems, as well as the evaluation of MVSE principles by SE practitioners.

Objective 3:

Identify which SE artefacts produce maximum value relative to effort.
In order to meet this objective, chapter 4 (section 4.7) specifically asked the respondents to

rank the principles of MVSE approach. Digital-first traceability (mean 4.1/5) and structured tailoring (mean 3.9/5) were evaluated most positively among others. Chapter 5 formulated five core artefacts in MVSE: Minimum Viable Requirements (MVR), Architecture Baseline (AB), Dynamic Interface Matrix (DIM), Risk-Based Verification Planning (RVP), and Decision Capture Log (DCL). Justification of these artefacts is provided for each core pain point.

Objective 4:

Design an MVSE approach which is efficient, risk-oriented, and lightweight.
Chapter 5 introduces the complete framework which represents the key features of MVSE: the 7 ± 2 cardinality requirement on minimum viable set of requirements (MVR); a model-centric architecture baseline as the single source of truth; a dynamic interface matrix updated weekly with impact analysis; a verification plan proportional to risk level (RVP); and a lightweight mechanism for documenting decisions (Decision Capture Log). Supportive artefacts include proportional gates and reviews, budget tracking, agile integration, and a change request process.

Objective 5:

Conduct a theoretical application of the framework with SE practitioners with feedback sessions.
Chapter 6 describes a 3.5-hour pilot workshop applying MVSE to the SHERLOCK 3U CubeSat mission. This exercise, conducted with systems engineers from DLR Bremen, successfully created MVR, DIM, RVP and DCL artefacts using a Notion workspace. The workshop proved the usability of MVSE and resulted in generation of the key artefacts in the timeframe of less than four hours.

7.2 Satisfaction of Research Questions

As outlined in chapter 1, this study posed eight specific research questions. All of them have been answered in light of empirical findings (chapter 4) and MVSE design (chapter 5):

1. **Level of non-value-added effort:** 26-50% (median value, section 4.3.1).
2. **Degree of bureaucracy of current SE process:** mean score of 2.54/5 (section 4.3).
3. **Approach to traceability and tool integration:** 45% of SE practitioners use spreadsheets for traceability; average satisfaction with integrated tools = 2.3/5 (section 4.5).
4. **Review policy and improvements provided:** 50% carry out verification reviews only

at major milestones; MVSE introduces proportional review gates and weekly status reviews (sections 4.6.2, 5.4.8.1)..

5. **Tailoring:** 55% apply ad-hoc tailoring; MVSE defines risk-driven tailoring rules (section 4.4).

6. **Major bottleneck challenges:** cultural resistance (45%) and lack of tools (41%) are the key issues; MVSE addresses these by means of lightweight artefacts.

7.2.1 Reflection on Limitations and Intended Use

As noted in Section 1.6, this thesis has several limitations that must be acknowledged when interpreting the MVSE framework and its applicability.

7.2.1.1 Empirical Limitations

The survey data presented in Chapter 4 is based on 44 responses used for core analysis. While this sample achieved thematic saturation for qualitative analysis, it does not support statistical generalisation to the entire European NewSpace sector. The findings should be interpreted as indicative rather than definitive. Additionally, all data are self-reported. Responses to questions about non-value-added effort and process bureaucracy reflect subjective perceptions, not objective measurements. While anonymity was maintained to encourage candour, perception bias cannot be eliminated.

7.2.1.2 Validation Limitations

A controlled longitudinal study comparing MVSE projects against traditional SE projects under identical conditions is not feasible in real-world NewSpace settings. Each project operates under unique constraints: different team compositions, funding profiles, technical maturities, and mission objectives. No two projects are directly comparable. Validation must therefore rely on cumulative evidence from multiple case studies, practitioner feedback, and iterative refinement. This thesis provides the first such case study. Future work should apply MVSE to a range of NewSpace missions – varying in team size, timeline, and mission type – to build a body of evidence for its effectiveness and to identify boundary conditions for its applicability.

7.2.1.3 Scope Limitations

Consistent with Section 1.6.2, MVSE was designed for uncrewed NewSpace missions with teams of 20 to 150 engineers and development timelines of 12 to 36 months. The framework is not applicable on:

- Crewed missions or systems involving human safety risk

- Programmes longer than 36 months or with teams larger than 150 engineers
- Highly regulated domains requiring formal certification artefacts (e.g., medical devices, nuclear systems)

Organisations operating in these contexts should augment MVSE with additional rigor or revert to full classical SE frameworks.

7.2.1.4 Assumptions and Prerequisites

The framework assumes a baseline level of engineering maturity, as noted in Section 1.6.3:

- Version control for code and configuration items
- Basic configuration management practices
- Test discipline and verification infrastructure

Teams without these foundations will not benefit from MVSE until those capabilities are established. The framework does not prescribe specific software tools or platforms; it remains tool-agnostic. Adopting organisations may implement MVSE with any toolchain that supports the required artifact types and traceability links.

7.2.1.5 Intended Use Revisited

As stated in Section 1.6.4, MVSE is offered as a foundation, not a final prescription. Organisations should tailor the framework to their specific risk profile, team size, and programme constraints. The framework is not intended to replace INCOSE, NASA, ECSS, or ISO standards for all applications. Rather, it provides a lightweight alternative for contexts where full standard compliance would create disproportionate overhead.

The MVSE framework is intended to be a starting point for NewSpace teams seeking to balance engineering velocity with mission assurance. It is not presented as a universal solution, but as a validated approach that has demonstrated usability in a pilot workshop and empirical support from survey data. Future work, including longitudinal validation and dedicated tool development, will determine the framework's broader applicability.

7.3 Summary of Contributions

There are three principal contributions that this thesis makes.

For SE practitioners and NewSpace organizations:

The MVSE approach provides an easy-to-implement methodology which includes artefact templates, their update schedules, criteria for review gates, and integration with agile methodologies. Both the workshop described in Chapter 6 and publicly available Notion workspace show that MVSE could be implemented within a couple of hours.

For SE researchers and academics:

Data from chapter 4 constitutes one of the first empirical studies of SE practices among European NewSpace organizations. In particular, the level of non-value-added effort (26-50%) provide important quantitative benchmark for further research.

For standards organizations (INCOSE, ECSS, NASA):

The data suggests that SE standards should be amended to introduce lightweight guidance for tailoring processes in smaller, fast-moving projects. Proportional governance structure and MVSE verification approach appear to be good candidates for inclusion in future updates to INCOSE or ECSS standard.

7.4 Closing Reflection

The NewSpace revolution goes beyond reusable launch vehicles, small satellites, and innovative business models – it demands reconsideration of the very engineering process. Standards in systems engineering were initially designed for billion-euro projects that take decades and involve hundreds of engineers. They cannot simply be scaled down to 18-month projects employing 20 people.

Minimum Viable Systems Engineering is not about abandoning the systems engineering process; it is about returning to its fundamentals: evidence of correctness, traceability, and preservation of knowledge. MVSE cuts ceremonious documentation, rigid phase gates, and false assumption of initial stability , while retaining essential rigor. Empirical evidence gathered throughout this research proves not just the necessity but urgency of the MVSE approach. Its pilot exercise shows that this goal is feasible.

Thus, the research aim stated in chapter 1 (design and evaluation of MVSE approach) has been successfully achieved. Whether longitudinal validation, development of dedicated tools, and expansion to larger projects will prove MVSE applicable even there remains to be seen. Regardless, MVSE appears to be an urgent need for small NewSpace start-ups, small satellite missions, and fast-moving organizations – the target audience of this thesis.

Appendix A: Survey Questions

Evaluation of Systems Engineering in NewSpace Development Lifecycle

Welcome, and thank you for participating in this research.

This survey is part of a Master thesis in Space Engineering at the University of Bremen. It investigates how Systems Engineering (SE) is applied in NewSpace environments, teams, and organisations operating with rapid iteration cycles, agile development models, and hardware-intensive workflows.

The goal is to assess current SE capabilities, identify practical challenges and inefficiencies, and understand how digital tools, tailoring, and verification processes are managed across the NewSpace development lifecycle. The results will support the creation of a Minimum Viable Systems Engineering (MVSE) framework optimized for fast-paced space projects while maintaining mission assurance.

The survey is anonymous, and no personal or sensitive data is required. It takes approximately 15-20 minutes and focuses solely on professional engineering practices.

At the end of the survey, you may optionally provide your email address if you are open to being contacted for brief follow-up questions or clarification at a later stage. This is entirely voluntary and will not affect the anonymity of your response dataset.

Thank you for your contribution. Your insights are highly valuable to this research.

Evaluation of Systems Engineering in New Space Development Lifecycle

Section A: Background

Sec	Q#	Question Text	Response Type & Options
A	A1	What type of organization do you work for?	Single Choice: Early-stage Newspace Startup, established newspace, Mature commercial, Heritage aerospace - government/prime, other
A	A2	How many years has your organization been active in space-system development?	Single Choice: < 1 year, 1–3 years, 3–5 years, > 5 years
A	A3	What is your primary role in Systems Engineering?	Single Choice: Lead SE / Chief Engineer, SE (mid-level), Project / Program Manager with SE responsibility, Technical Lead / Subsystem Engineer, AIV / PA, Other [Text Input]
A	A4	What is the typical duration of a development cycle for your current project?	Single Choice: < 6 months, 6 months – 1 year, 1 – 2 years, > 2 years
A	A5	What is the approximate size of the SE team (including any part-time or support staff) on your current project?	Numeric input

Section B: Process & Tailoring

Sec	Q#	Question Text	Response Type & Options
B	B1	Which SE standards or frameworks does your organization formally follow or try to follow?	Multiple Choice: INCOSE, ISO/IEC 15288, NASA NPR 7123.1, ECSS-E-ST-10C, Custom/hybrid, Informal/ad-hoc
B	B2	Rate your agreement (1=Strongly Disagree, 5=Strongly Agree): (a) Our SE process is too heavy/bureaucratic for our team size and timeline. (b) We spend significant time on SE documentation that adds little mission value. (c) SE artefacts (requirements, ICDs, etc.) are frequently outdated or inconsistent. (d) Balancing formal verification rigor with rapid iteration is a major difficulty. (e) Our SE process is proportional to the actual mission criticality and risk profile.	5-Point Likert Scale (Matrix)
B	B3	How many formal, mandatory SE Review Gates (e.g., MDR, SRR, PDR, CDR, QR, AR, FRR, LRR, ORR, MCR, etc.) are typically included in a project of standard duration?	Single Choice (Binned): 1. Minimal (1–3 reviews); 2. Standard Cycle (4–7 reviews); 3. Multiple Layers (8–12 reviews); 4. Excessive (> 12 reviews)
B	B4	Do you have a documented method for deciding which SE activities can be omitted or simplified?	Single Choice: Yes (formal), Yes (informal/ad-hoc), No (case-by-case), Not sure
B	B5	Approximately what percentage of your SE team's time is spent on non-value-added activities (manual documentation, manual traceability, data synchronization, re-verification due to inconsistency)?	Numeric input (%)

Section C: Agility & Governance

Sec	Q#	Question Text	Response Type & Options
C	C1	How does your organization currently approach SE tailoring?	Single Choice: Strictly follows standard, Single "fit-for-all" process, Tailors proportional to risk, Ad-hoc tailoring, Agile/Lean SE, Other [Text Input]
C	C2	Rate your agreement (1=Strongly Disagree, 5=Strongly Agree): (a) Our SE process is proportional to mission risk and system criticality. (b) We can rapidly adjust SE rigor (e.g., verification depth) when priorities shift. (c) Design and verification are performed in short, iterative cycles rather than only at phase gates. (d) The team clearly knows which SE activities are mandatory versus optional. (e) Decision-making about SE scope is driven by quantitative risk metrics rather than intuition.	5-Point Likert Scale (Matrix)
C	C3	Rank the following factors (1=Most Important, 5=Least Important): (a) Speed to first flight / delivery (b) Regulatory / contractual compliance (c) Team learning & continuous improvement (d) Verification rigor & safety assurance (e) Cost efficiency & resource optimization	Rank Order (1-5)
C	C4	How often does your team hold retrospective or process-improvement meetings focused on SE practices?	Single Choice: After every sprint/iteration, Monthly, Quarterly, Only at major milestones, Never

Section D: Tools & Digital Integration

Sec	Q#	Question Text	Response Type & Options
D	D1	Which of the following tools are used on your current project for SE activities?	Multiple Choice: 000 Dedicated MBSE tool, Spreadsheets, Document-centric platforms, Git, Issue-tracking, Simulation, CAD, Multiple disconnected tools, Other [Text Input]
D	D2	What is the approximate number of distinct software tools (e.g., dedicated MBSE tool, Jira, Confluence, Excel, CAD, Git, etc.) used to manage SE artefacts on a typical project?	Single Choice (Binned): 1. Few Tools (1–4); 2. Moderate (5–9); 3. High (10–15); 4. Extreme (> 15)
D	D3	Rate your agreement (1=Strongly Disagree, 5=Strongly Agree): (a) Our SE tools are well-integrated (live data sync, no manual re-entry). (b) We have automated traceability across requirements $\rightarrow$ design $\rightarrow$ verification. (c) Our team would benefit from a unified, lightweight MBSE platform. (d) We can export or import data between our SE tools without manual re-formatting.	5-Point Likert Scale (Matrix)
D	D4	Does your organization maintain a single source of truth (e.g., a database or MBSE model) for all SE artefacts?	Single Choice: Yes (all), Partially, No (scattered)

Section E: Verification & Validation Pain Points

Sec	Q#	Question Text	Response Type & Options
E	E1	Rate the severity (1=No problem, 10=Critical bottleneck): (a) Traceability latency – time to trace a requirement change to affected tests. (b) Data fragmentation – spread of requirements/design/verification data across tools. (c) Verification closure effort – person-hours per cycle to close V&V deviations. (d) Requirements stability – frequency of post-baseline requirement changes that cause rework. (e) Test-environment setup time – effort to provision a repeatable test configuration.	10-Point Rating Scale (Matrix)
E	E2	How often does your team conduct a formal verification status review?	Single Choice: Weekly, Bi-weekly, Monthly, Only at major milestones (e.g., CDR, QR), Rarely / never
E	E3	What proportion of verification results are captured automatically (e.g., via scripts, data logs) versus entered manually?	Single Choice: 80% automatic, 50–80% automatic, 20–50% automatic, < 20% automatic, None automatic
E	E4	When a verification anomaly is discovered, what is the typical time required to update the related requirement(s) and the Verification Matrix / VCD?	Single Choice (Binned): 1. Fast (< 8 hours); 2. Agile (8 hours to 3 days); 3. Traditional (4 to 10 days); 4. Slow (> 10 days)

Section F: MVSE Future State

Sec	Q#	Question Text	Response Type & Options
F	F1	If you could redesign your SE process, what is the single most important change would you make?	Text Area (max 3 sentences)
F	F2	Rate how valuable each MVSE principle would be (1=Not valuable, 5=Highly valuable): (a) structured tailoring – proportional SE rigor based on criticality. (b) Digital-first traceability – automated, live requirement-to-test linkage. (c) Sprint-level verification – continuous V&V integrated into short cycles. (d) Minimal mandatory artefacts – smaller set of "must-have" SE outputs. (e) Evidence-centric, not document-centric – focus on audit-ready data. (f) Integrated risk-tracking dashboard that updates automatically with requirement changes.	5-Point Likert Scale (Matrix)
F	F3	Which of the following would be the biggest barrier to adopting an MVSE-style framework?	Cultural resistance, Lack of management support/funding, Insufficient tooling, Perceived loss of compliane, Limited expertise, Other [Text Input]

Appendix B: Consent Form

Informed Consent of Study Participation

You are invited to participate in the online study "Minimum Viable Systems Engineering". The study is conducted by Satya Sree Rupa Rallabandi and supervised by Dr. Caroline Lange , Dominik Quantius from the DLR & University of Bremen. The study with an estimated 50 participants takes place in the period from 2025-11-17 to 2025-12-15.

Please note:

- Your participation is entirely voluntary and can be discontinued or withdrawn at any time
- One session of the online study will last ca. 25 minutes
- You have no direct benefit from participating in the study, but your support our work and helps to advance research in this area
- For the evaluation, we collect some contact data, only be used for feedback or further information about the study and will not be passed on to any third parties
- During the session, we will log your input and manually record notes
- Recordings and personal data are subject to the guidelines of the General Data Protection Regulation (GDPR) and will be pseudo-anonymised (with a coded number) stored, evaluated, and potentially published so that without information from the researchers no conclusions can be drawn about individual persons.

The alternative to participation in this study is to choose not to participate. If you have any questions, concerns, or complaints about the informed consent process of this research study or your rights as a human research subject, please contact Dr. Caroline Lange , Dominik Quantius.

Please read the following information carefully and take the time you need.

1. **Purpose and Goal of this Research :** Identifying the pain points in the current SE processes. Using the pain points as the requirements for deriving the MVSE framework. Your participation will help us achieve this research goal. The results of this research may be presented at scientific or professional meetings or published in scientific proceedings and journals.

2. **Study Participation :** Your participation in this online study is entirely voluntary and can be discontinued or withdrawn at any time. You can refuse to answer any questions or continue with the study at any time if you feel uncomfortable in any way. You can discontinue or withdraw your participation at any time without giving a reason. However, we reserve the right to exclude you from the study (e.g., with invalid trials or if the data is irrelevant). Repeated participation in the study is not permitted.

3. **Risks and Benefits :** In the online study, you will not be exposed to any immediate risk or danger. As with all computer systems that process data, despite security measures, there is a small risk of data leakage and the loss of confidential or personal information. You have no direct benefit from participating in the study, but your support helps advance research in this area.

4. **Data Protection and Confidentiality :** In this study, subject-related data are collected for our research. The use of personal or subject-related information is governed by the European Union (EU) General Data Protection Regulation (GDPR) and will be treated in accordance with the GDPR. This means that you can view, correct, restrict processing, and delete the data collected in this study. Only with your agreement, we will log your input and manually record notes in the study. We plan to publish the results of this and other research studies in academic articles or other media. Your data will not be retained for longer than necessary or until you contact researchers to have your data destroyed or deleted.

   Access to the raw data, transcribed interviews, and observation protocols of the study is encrypted, password-protected and only accessible to the authors, colleagues and researchers collaborating on this research. Other members and administrators of our institution do not have access to your data. When publishing, the data will be anonymised using code numbers and published in aggregated form, so that without the researchers' information, no conclusions can be drawn about individual persons. Any interview content or direct quotations from the interview, that are made available through academic publications or other academic outlets will also be anonymized using code numbers. Contact details (e.g.e-mails) will not be passed on to third parties, but may be used by the researchers to contact participants, trace infection chains, or to send you further details of the study. According to the GDPR, the researchers will inform the participants using their contact details if a confidential data breach has been detected.

5. Identification of Investigators If you have any questions or concerns about the research, please feel free to contact:

Researcher: Satya Sree Rupa Rallabandi

Principal investigators: Dr.-Ing Caroline Lange Dipl. -Ing. Dominik Quantius

Appendix C: Workshop Data

C.1 MVSE Validation Workshop - Invitation

C.1.1 Topic

Evaluation of selected artefacts of the Minimum Viable Systems Engineering (MVSE) framework through a structured systems engineering workshop using a predefined mission scenario and toolchain.

C.1.2 Goal

The goal of the workshop is to validate the practical applicability and usefulness of the key MVSE artefacts that represent the novel contribution of the framework. Participants will be provided with a predefined mission scenario together with the first two artefacts of the framework: the Minimum Viable Requirements (MVR) and the Lightweight Architecture Baseline (AB). Based on this starting point, participants will apply the remaining MVSE artefacts in order to assess whether they effectively support rapid systems engineering activities while maintaining sufficient engineering rigor.

The workshop also aims to collect qualitative feedback regarding the clarity, usability, and perceived value of these artefacts in a fast-paced NewSpace development context.

C.1.3 Agenda

1. **Introduction and Workshop Objectives (20 minutes)**
   Introduction to the MVSE framework, explanation of the workshop structure, and overview of the artefacts to be evaluated.

2. **Mission Scenario and Baseline Presentation (20 minutes)**
   Presentation of the example mission scenario together with the predefined Minimum Viable Requirements (MVR) and Lightweight Architecture Baseline (AB).

3. **Hands-on Application of MVSE Artefacts (2–2.5 hours)**
   Participants apply the remaining MVSE artefacts using the provided baseline information:

   - Dynamic Interface Matrix (DIM)
   - Risk-Based Verification Plan (RVP)

- Knowledge-Capture Log (KCL)

4. **Group Reflection and Discussion (30 minutes)**
   Participants reflect on the process and discuss usability, clarity, and effectiveness of the artefacts in supporting systems engineering activities.

5. **Feedback Collection (15–20 minutes)**
   Structured feedback is collected through questionnaires and open discussion to evaluate the strengths and limitations of the MVSE approach.

C.2 Workshop Handout

C.2.1 Validation of the MVSE Framework

C.2.1.1 Introduction

In the following, the study scope of the planned MVSE Validation workshop on 1st April 2026 at DLR Institute of Space Systems is defined.

The topics listed below are described within this document:

- Purpose
- Problem Statement
- Mission Objectives
- Study Scenario
- Study Objectives
- Participant Role
- Preliminary Schedule

C.2.1.2 Purpose

The SHERLOCK mission has completed an initial concurrent engineering study phase in which mission objectives, operational concepts, and a preliminary system architecture were established. The study confirmed overall mission feasibility and produced an initial system definition.

As the project progresses toward subsequent development phases, the current system description requires further consolidation to support integration planning, verification preparation, and informed engineering decision-making.

C.2.1.3 Problem Statement

At this stage, several typical challenges emerge:

- subsystem interfaces are only partially formalized,
- verification activities are not yet prioritized according to system risk,
- engineering decisions and assumptions lack structured traceability.

Applying traditional systems engineering processes would introduce significant documentation overhead, while purely agile approaches may compromise consistency and traceability. The project, therefore, requires a balanced methodology that preserves engineering rigour while remaining lightweight and adaptable.

C.2.1.4 Mission Objectives

A student-designed 3U CubeSat shall be launched into low Earth orbit to inspect Eu:CROPIS spacecraft to analyze its attitude progression and structural conditions.

- Eu:CROPIS possibly tumbling
- verify tumbling with visual inspection
- inspect structure for possible damage
- Impact of micro debris
- tank/experiment explosion

C.2.1.5 Study Scenario

This workshop is conducted as part of a master's thesis to evaluate the Minimum Viable Systems Engineering (MVSE) Framework. The session focuses exclusively on the practical implementation and testing of the framework developed within the thesis. The workshop builds upon a Concurrent Engineering Facility (CEF) study conducted in February 2025 by students of the Space Engineering programme as part of their academic curriculum, during which a mission concept was developed using a Scrum-based approach.

Selected artefacts and system elements generated during the original study are reused solely as a working scenario to enable the application of the MVSE framework. The mission itself is not part of the evaluation process and serves only as contextual background for the exercise. The workshop therefore does not assess mission design quality, technical performance, or engineering outcomes.

The evaluation is strictly focused on the MVSE methodology, including its artefacts, workflows, and supporting toolchain. The success of the workshop is measured exclusively by the insights gained into usability, collaboration efficiency, and the effectiveness of the MVSE framework in a realistic engineering environment.

C.2.1.6 Study Objectives

- Produce a complete DIM, RVP, and DCL for the SHERLOCK mission using Dalus.
- Validate that each artefact satisfies the "minimum-viable" definition

- Capture feedback on clarity, effort, perceived rigor, and adoption intent.
- Record all decisions, versioned changes, and traceability links within Dalus and DCL to preserve institutional knowledge.
- Assess overall participant satisfaction with the MVSE workflow.

C.2.1.7 Participant Role

Participants act as members of the SHERLOCK systems engineering team responsible for preparing the mission for the next stage of development. The objective is not to redesign the mission architecture, but to strengthen the existing baseline through structured analysis and documentation.

A. Refine Requirements (MVR)

- Review the provided Phase A requirements.
- Identify and retain only essential, clear, and testable requirements.
- Reformulate requirements where necessary.
- Document the refined set as Minimum Viable Requirements (MVR) in the provided tool.

B. Define System Interfaces (DIM)

- Identify interfaces between study elements (Payload, Platform, Communications, Ground).
- Capture exchanged data, dependencies, and interface ownership.
- Record uncertainties or missing information.
- Document interfaces in the Dynamic Interface Matrix (DIM).

C. Define Risk-Based Verification (RVP)

- Identify risks related to requirements and interfaces.
- Verify with the Risk Assessment report
- Define verification activities only where justified by risk.
- Select suitable verification methods (Analysis, Inspection, Test, Demonstration).
- Record results in the Risk-Based Verification Plan (RVP).

D. Capture Decisions and Knowledge (DCL)

- Continuously record engineering decisions during discussions.
- Capture rationale, assumptions, alternatives, and open issues.
- Use the provided Decision Capture Log (DCL) tool.

Table C.1: Roles and Responsibilities

Role	**Responsibility (Workshop Context)**	**Assigned Participant**
Systems Engineer (SE-1,2)	Coordination of MVR refinement and overall system consistency; Interface coordination and DIM consolidation	
GNC	Identify guidance, navigation, and control related requirements and interfaces	
Power	Define power-related constraints, interfaces, and verification inputs	
Structures	Assess structural interfaces, loads, and platform constraints	
Onboard Computer (OBC)	Define data handling requirements and subsystem interactions	
Communications	Define communication interfaces and operational data flow	
Payload	Represent payload needs and operational requirements	
Risk	Lead risk identification and support Risk-Based Verification Planning (RVP)	

C.2.1.8 Schedule Overview (Reference)

Table C.2: Schedule Overview

Time	**Activity**
09:30-09:45	Introduction, purpose, and charter review
09:45-10:00	Mission scenario presentation (SHERLOCK)
10:00-10:15	Set up
10:15-10:45	Round 1 - MVR, DCL
10:45-11:00	Break
11:15-12:00	Round 2 - DIM for Interfacing roles; RVP for SE & Risk Roles
12:00-12:30	Group Reflection - discussion on the usability, clarity of the framework, effectiveness and potential use cases of the artefacts in the Systems engineering activities
12:30-12:50	Feedback collection through a questionnaire and open for comments
12:50-13:00	Wrap up

C.3 Work Instructions

C.3.0.1 Role: Systems Engineer

Systems Engineers are responsible for maintaining overall system coherence and ensuring correct application of MVSE principles throughout the workshop.

ENSURE THAT ALL DECISIONS ARE NOTED IN THE DCL NOTION FORM

Minimum Viable Requirements (MVR)

Responsibilities

- Review the existing requirement set in Dalus.io.
- Identify requirements essential for system operability and mission feasibility.
- Detect:
  - duplicate requirements
  - ambiguous wording
  - non-testable statements
  - unnecessary specification depth.

Actions

- Simplify requirements where possible.
- Ensure each requirement:
  - expresses a single intent
  - is measurable or verifiable
  - aligns with architecture elements.
- Coordinate subsystem inputs before modifying shared requirements.

Dynamic Interface Matrix (DIM)

Responsibilities

- Lead cross-discipline interface identification.
- Ensure subsystem engineers define interactions consistently.

Actions

- Verify all critical subsystem interactions are captured:
  - data exchange
  - command flow
  - power dependencies
  - operational interactions.
- Assign interface ownership.

Risk-Based Verification Planning (RVP)

Responsibilities

- Ensure verification effort follows risk prioritisation.

Actions

- Confirm verification exists for high-risk items.
- Challenge unnecessary verification proposals.
- Maintain alignment between: requirements <–> risks <–> verification activities.

C.3.0.2 Role: GNC, Power, Structures, OBC, Communications, Payload

Provide subsystem expertise required to validate requirements, define interfaces, and enable risk-based verification planning.

ENSURE THAT ALL DECISIONS ARE NOTED IN THE DCL NOTION FORM

Requirement Review (MVR Support)

Tasks

- Analyse requirements affecting your subsystem.
- Simplify requirements where possible.
- Ensure each requirement:
  - expresses a single intent
  - is measurable or verifiable
  - aligns with architecture elements.
- Identify:
  - missing constraints
  - unclear performance expectations
  - unrealistic assumptions.
- Recommend simplifications consistent with MVSE minimalism.

Interface Definition (DIM)

Tasks

- Review architecture links involving your subsystem.
- Define:
  - inputs received
  - outputs generated
  - operational dependencies.
- Clarify interface ownership.
- Map 3-4 requirements with full traceability as shown in the example.

- Mark all edges with the same ID for each requirement. I.e 1 requirement = 1 Edge ID
- Label each flow with the respective Requirement ID and follow the colour coding.

Examples

- GNC <–> OBC command flow
- Payload <–> Power consumption profile
- Comms <–> Ground data transfer

Verification Contribution (RVP) Tasks

- Propose verification only where risk justifies effort.
- Select an appropriate verification method based on the risk category.

C.3.0.3 Role: Risk Management

Ensure engineering effort is guided by risk awareness rather than completeness.

ENSURE THAT ALL DECISIONS ARE NOTED IN THE DCL NOTION FORM

Tasks

- Go through the risks provided Risk Assessment report.
- Link risks to relevant requirements.
- Take 3-4 risks per category and enter them in the Dalus environment.
- Select appropriate risk classification.
- Ensure high-risk items receive verification coverage.
- Consult with respective stakeholders/teams to check the level of risk.
- Discuss and settle the verification type based on risk level.
- Record reasoning behind risk decisions in DCL.

Appendix D: Documentation of AI-Based Tools Usage

During the course of this thesis, several AI-based applications and tools were used to support literature review, data analysis, coding, survey validation, and language refinement. The following table documents the purpose, input, and output of each tool.

Remarks on Responsible Use

- All AI-generated content, including literature summaries and code snippets, was manually verified against original sources or tested for correctness.
- The AI-generated survey mock data (Section 3.3.3) was used exclusively for instrument validation and expectation calibration. It was not included in the empirical findings presented in Chapter 4.
- Grammarly was used only for grammar and spelling corrections, not for content generation or structural changes.
- The final thesis content, including all arguments, interpretations, and conclusions, is solely the work of the author.

Table D.1: Documentation of AI-Based Tools Used in This Thesis

AI Tool	**Use Case**	**Thesis Section**	**Description of Input (Prompt)**	**Remark**
Anara	Literature review assistance	Chapter 2 (Literature Review)	Topic-specific queries on NewSpace SE, INCOSE limitations, agile hardware development	Helped identify relevant papers and synthesize key arguments; all AI-generated content was verified against original sources
ChatGPT	Survey mock data generation	Section 3.3.3 (Pilot Testing)	Simulated 50 practitioner responses across three organisational archetypes (startup, established, student) using the survey structure from Appendix A	Used exclusively for instrument validation; not included in final empirical findings
DeepSeek	Coding assistance for data analysis	Section 3.7 (Data Analysis Methodology)	Assistance with Python code for data cleaning, visualisation (matplotlib, seaborn), and statistical calculations	Generated code snippets were reviewed, tested, and modified where necessary
Grammarly	Language and grammar correction	Entire thesis	Automated grammar, spelling, and style suggestions throughout all chapters	Used for proofreading only; no content generation
ChatGPT	Framework design refinement	Chapter 5 (MVSE Framework)	Iterative refinement of artefact definitions, guidelines, and working examples	Used as a brainstorming and structuring tool; all framework content was developed and validated by the author

Bibliography

[1] P. Breda, V. Di Pietrantonio, and S. Schneider, "Spaceport selection, licensing complexity and launch operator constraints in NewSpace", *Journal of Space Safety Engineering*, 2026. DOI: https://doi.org/10.1016/j.jsse.2026.02.012

[2] W. Peeters, "The Paradigm Shift of NewSpace: New Business Models and Growth of the Space Economy", *New Space*, 2024. DOI: 10.1089/space.2023.0060

[3] I. Chechile, *NewSpace Systems Engineering*. Springer, 2021. DOI: 10.1007/978-3-030-66898-3

[4] Aerospace.org, "Keeping Pace with a Rapidly Evolving Launch Landscape", 2025. Accessed: Apr. 6, 2026. [Online]. Available: https://aerospace.org/article/keeping-pace-rapidly-evolving-launch-landscape

[5] N. S. Economy, "The Fragile Architecture of the Space Economy", *Market Segment*, 2026. Accessed: Apr. 6, 2026. [Online]. Available: https://newspaceeconomy.ca/2026/03/23/the-fragile-architecture-of-the-space-economy/

[6] INCOSE SE, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 5th ed. Hoboken, NJ: John Wiley & Sons, 2023.

[7] NASA NPR, "NASA Systems Engineering Handbook", National Aeronautics and Space Administration, Tech. Rep. NASA/SP-2016-6105 Rev2, 2016.

[8] *ISO/IEC/IEEE 15288:2015 Systems and Software Engineering – System life cycle processes*, ISO/IEC/IEEE, 2015.

[9] *ECSS-E-ST-10C Rev.1: Space Engineering – System Engineering General Requirements*, European Cooperation for Space Standardization (ECSS), 2014.

[10] ECSS, *ECSS-E-ST-10C Rev.1: Space engineering – System engineering general requirements*, European Cooperation for Space Standardization, 2017.

[11] ECSS, *ECSS-S-ST-00-02C: ECSS system – Tailoring (DRAFT)*, European Cooperation for Space Standardization, 2020.

[12] M. Group, "It's coming home: The return of agile hardware product development", *McKinsey's Operations Practice*, 2023.

[13] Beck,Nick K. Taylor.,et al. "Manifesto for agile software developments", Accessed: Apr. 6, 2026. [Online]. Available: http://agilemanifesto.org/

[14] The Standish Group International, Inc., "CHAOS REPORT 2015", *Journal of Computer and Communications*, 2015.

[15] K. Schwaber and J. Sutherland, “The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.”, *Journal of Software Engineering and Applications, Vol.18 No.4, April 21, 2025*, 2025.

[16] A. Hajou and M. Voropaeva, “Engineering Agility: A Pragmatic Approach to Adapting Agile Practices for Hardware Product Development”, 2026.

[17] N. Garzaniti and A. Golkar, “Performance Assessment of Agile Hardware Co-development Process”, in *ISSE 2020 - 6th IEEE International Symposium on Systems Engineering*, IEEE, 2020.

[18] Better Devices, “White Paper: Hardware Development in Cross-Functional Agile Teams”, 2025.

[19] C. Grimm and J. Hendrikse, “Concurrent AIV as a Method to Hard Tailor Test- and Model Philosophies in Times of Need”, *MethodsX*, vol. 6, pp. 2148–2155, 2019. DOI: 10.1016/j.mex.2019.09.001

[20] J. A. D. Macedo et al., “CertiA360: Enhance Compliance Agility in Aerospace Software Development”, *arXiv*, 2025.

[21] C. D. Grimm, J.-T. Grundmann, J. Hendrikse, C. Lange, C. Ziach, and T.-M. Ho, “From idea to flight - A review of the Mobile Asteroid Surface Scout (MASCOT) development and a comparison to historical fast-paced space programs”, *Progress in Aerospace Sciences*, vol. 104, pp. 20–39, 2019. DOI: https://doi.org/10.1016/j.paerosci.2018.11.001

[22] M. Group, “Lean management or agile? The right answer may be both”, *McKinsey’s Operations Practice*, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/lean-management-or-agile-the-right-answer-may-be-both

[23] B. W. Oppenheim, *Lean for systems engineering with lean enablers for systems engineering*. Wiley, 2011.

[24] D. Systèmes, “Reducing Non-Value-Added Work in Engineering”, *Advanced Manufacturing*, 2015.

[25] R. S. Friedenthal S Alan Moore, *A Practical Guide to SysML: The Systems Modeling Language.* Morgan Kaufmann, 2014.

[26] O. Sysmlv2. “Sysmlv2 the next generation systems modeling language”. [Online]. Available: https://www.omg.org/sysml/sysmlv2/

[27] S. Technologies, *What is New Space?*, 2021. Accessed: Apr. 12, 2026. [Online]. Available: https://solar-mems.com/blog-news/what-is-new-space/

[28] Baber, W.W., Ojala, A., “New Space Era: Characteristics of the New Space Industry Landscape”, *Space Business*, 2024. DOI: 10.1007/978-981-97-3430-6_1

[29] O. de Weck, A. Siddiqi, M. Moraguez, A. Trujillo, G. Lordos, and M. Sarang, "Commercial Space Technology Roadmap", Massachusetts Institute of Technology, Tech. Rep. 80NSSC17K0330, 2019.

[30] Summit Ventures Partners, *Company Overview: SpaceX*, 2025. Accessed: Apr. 6, 2026. [Online]. Available: https://www.summit-ventures.net/company/space-x/

[31] Axiom Space, *In-Space Infrastructure and Logistics*, 2024. Accessed: Apr. 6, 2026. [Online]. Available: https://axiomspace.squarespace.com/inspace-infrastructure-and-logistics

[32] Factories in Space, *Relativity Space*, 2025. Accessed: Apr. 6, 2026. [Online]. Available: https://www.factoriesinspace.com/relativity-space

[33] Firefly Aerospace. "Mission updates – alpha flight 7 path forward", Accessed: Apr. 6, 2026. [Online]. Available: https://fireflyspace.com/missions/alpha-flta007/

[34] ASTRA e.V., *Phoenix*, https://www.astra-bremen.de/projects, 2024. Accessed: Apr. 12, 2026.

[35] E. Herbert, R. Sega, J. McGraw, and T. Bradley, "Risk Management for Commercial-Off-The-Shelf Parts Based Space Hardware", *Systems Engineering*, vol. 28, pp. 374–392, 2025. DOI: 10.1002/sys.21797

[36] MEWS PARTNERS, *Aerospace start-ups: Between ambitions and challenges*, MEWS News Release, 2024. Accessed: Apr. 12, 2026. [Online]. Available: https://www.mews-partners.com/en/aerospace-start-ups-between-ambitions-and-challenges/

[37] B. C. Yalcin, V. Muralidharan, and Kremer, *Ground-Based Testing of In-Orbit Interactions Under Scale and Time-Delay Limitations*, TechRxiv, 2025. DOI: 10.36227/techrxiv.176160107.71718266/v1

[38] L. Harris, "A Probabilistic Approach for Margin Allocation Tradeoffs Pertaining to Early Two-Stage-To-Orbit Launch Vehicle Design", Ph.D. dissertation, Georgia Institute of Technology, 2023. Accessed: Apr. 12, 2026. [Online]. Available: https://hdl.handle.net/1853/72766

[39] M. D. Guenov, X. Chen, A. Molina-Cristobal, A. Riaz, A. S. J. van Heerden, and M. Padulo, "Margin allocation and tradeoff in complex systems design and optimization", *AIAA Journal*, vol. 56, no. 7, pp. 2887–2902, 2018. DOI: 10.2514/1.J056357

[40] K. Paliwoda, "Managing people for innovation: leveraging dynamic capabilities in space startups", *JBIS, Journal of the British Interplanetary Society*, vol. 78, 2025. DOI: 10.59332/jbis-078-01-0027

[41] P. A. Pavlou and O. A. El Sawy, "The 'Third Hand': IT-Enabled Competitive Advantage in Turbulence Through Improvisational Capabilities", *Information Systems Research*, vol. 21, no. 3, pp. 443–471, 2010. DOI: 10.1287/isre.1100.0280

[42] E. Bjarnason, D. Šmite, N. Johansson, and K. Wnuk, "The Role of Responsiveness to Change in Large Onboarding Campaigns", in *Agile Processes in Software Engineering and Extreme Programming*, ser. Lecture Notes in Business Information Processing, vol. 475, Springer, 2023, pp. 131–147. DOI: 10.1007/978-3-031-33976-9_9

[43] A. Tiwana and H. Safadi, "Synchronizing Development Teams with Drifting Software Systems: A Decomposability-Based Theory of Evolutionary Mechanisms", *Journal of Management Information Systems*, vol. 43, 2026. DOI: 10.1080/07421222.2025.2602394

[44] J. Austin-Breneman and J. Bayley, *Tough Tech Fundamentals: Product Development*, The Engine Learning Concepts, 2025. Accessed: Apr. 12, 2026. [Online]. Available: https://engine.xyz/resources/fundamentals-concepts-product-development

[45] B. Sauser, J. E. Ramirez-Marquez, R. Magnaye, and W. Tan, "External Change in Large Engineering Design Projects: The Role of the Client", *IEEE Transactions on Engineering Management*, vol. 53, no. 3, pp. 426–439, 2006. DOI: 10.1109/TEM.2006.878100

[46] DARPA, *DARPA Launch Challenge Closes With No Winner*, DARPA News Release, 2020. Accessed: Apr. 12, 2026. [Online]. Available: https://www.darpa.mil/news-events/2020-03-02

[47] Hackaday, *NASA Mission Off To Rough Start After Astra Failure*, Hackaday, 2022. Accessed: Apr. 12, 2026. [Online]. Available: https://hackaday.com/2022/06/13/nasa-mission-off-to-rough-start-after-astra-failure/

[48] D. L. Morgan, *Integrating Qualitative and Quantitative Methods: A Pragmatic Approach.* Thousand Oaks, CA: Sage Publications., 2014.

[49] J. W. Creswell, "Designing and Conducting Mixed Methods Research", *Psychology, Vol.11 No.3*, 2018.

[50] R. K. Yin, *Case Study Research and Applications: Design and Methods (6th ed.)* Thousand Oaks, 2018.

[51] Greene, *Toward a Conceptual Framework for Mixed-Method Evaluation Designs Volume 11, Issue 3*. Educational Evaluation and Policy Analysis, 1989.

[52] https://www.limesurvey.org/. Accessed: Apr. 6, 2026.

[53] https://www.spacetechexpo-europe.com/. Accessed: Apr. 6, 2026.

[54] G. A. Miller, “The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information”, *Psychological Review, 63, 81-97.*, 1956.

Universität Bremen

Hinweise zu den offiziellen Erklärungen

1. Die folgende Seite mit den offiziellen Erklärungen

   A) Eigenständigkeitserklärung

   B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

   C) Einverständniserklärung über die Bereitstellung und Nutzung der Bachelorarbeit / Masterarbeit in elektronischer Form zur Überprüfung durch eine Plagiatssoftware

   ist entweder direkt in jedes Exemplar der Bachelor- oder Masterarbeit fest mit einzubinden oder unverändert im Wortlaut in jedes Exemplar der Bachelor- oder Masterarbeit zu übernehmen.

   Bitte achten Sie darauf, jede Erklärung in allen drei Exemplaren der Arbeit zu unterschreiben.

2. In der digitalen Fassung kann auf die Unterschrift verzichtet werden. Die Angaben und Entscheidungen müssen jedoch enthalten sein.

Zu B)
Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden.

Zu C)
Das Einverständnis der dauerhaften Speicherung des Textes ist freiwillig.
Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden.
Weitere Informationen zur Überprüfung von schriftlichen Arbeiten durch die Plagiatsoftware sind im Nutzungs- und Datenschutzkonzept enthalten. Diese finden Sie auf der Internetseite der Universität Bremen.

Notes on the official declarations

1. The following pages with the official declarations
   A) Declaration of Authorship
   B) Declaration on the Publication of Bachelor's or Master's Thesis
   C) Declaration of Consent for the Provision and Use of the Bachelor's Thesis / Master's Thesis in Electronic Form for Review by Plagiarism Software

   is to be either integrated directly into each copy of the bachelor's or master's thesis or adopted unchanged in the wording of each copy of the bachelor's or master's thesis.

   Please be sure to sign each declaration in all three copies of the thesis.

2. The signature can be omitted from the digital version. However, the information and decisions must be included.

Regarding B)
The consent can be revoked at any time with future effect by notifying the University of Bremen.

Regarding C)
Consent for the permanent storage of the text is voluntary.The consent can be revoked at any time with future effect by notifying the University of Bremen.

Further information on the checking of written work using plagiarism software can be found in the data protection and usage concept. This can be found on the University of Bremen website.

Universität Bremen

Eigenständigkeits- und Einverständniserklärung zur Überprüfung mit Plagiatssoftware sowie die Erklärung zur Veröffentlichung bei Bachelor- und Masterarbeiten

Declarations of Authorship and Consent for Checking with Plagiarism Software and the Declaration of Publication for Bachelor's and Master's Thesis

Studierenden-Angaben / Student Information:

Matrikelnr./ Student ID	**6256157**
Nachname / Surname	**Rallabandi**
Vorname / First Name	**Satya Sree Rupa**

Titel der Arbeit / Title of Thesis

Minimum Viable Systems Engineering (MVSE) for Rapid Space-System Development

A) Eigenständigkeitserklärung / Declaration of Authorship

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet, dazu zählen auch KI-basierte Anwendungen oder Werkzeuge.
Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht.

I hereby affirm that I have written the present work independently and have used no sources or aids other than those indicated. All parts of my work that have been taken from other works, either verbatim or in terms of meaning, have been marked as such, indicating the source. The same applies to drawings, sketches, pictorial representations and sources from the Internet, including AI-based applications or tools. The work has not yet been submitted in the same or a similar form as a final examination paper.

☑ Ich habe KI-basierte Anwendungen und/oder Werkzeuge genutzt und diese im Anhang "Nutzung KI basierte Anwendungen" dokumentiert.

I have used AI-based applications and/or tools and documented them in the appendix "Use of AI-based applications".

B) **Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten**
Declaration regarding the publication of bachelor's or master's thesis

Die Abschlussarbeit wird zwei Jahre nach Studienabschluss dem Archiv der Universität Bremen zur dauerhaften Archivierung angeboten. Archiviert werden:

Two years after graduation, the thesis is offered to the archive of the University of Bremen for permanent archiving. The following are archived:

1) Masterarbeiten mit lokalem oder regionalem Bezug sowie pro Studienfach und Studienjahr 10 % aller Masterarbeiten
Master's theses with a local or regional focus, as well as per subject and academic year 10% of all Master's thesis

2) Bachelorarbeiten des jeweils ersten und letzten Bachelorabschlusses pro Studienfach und Jahr.
Bachelor's thesis for the first and last bachelor's degrees per subject and year.

☑ Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.
I agree that my thesis may be viewed by third parties in the university archive for academic purposes.

☑ Ich bin damit einverstanden, dass meine Abschlussarbeit nach 30 Jahren (gem. §7 Abs. 2 BremArchivG) im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.
I agree that my thesis may be viewed by third parties for academic purposes in the university archive after 30 years (in accordance with §7 para. 2 BremArchivG).

☐ Ich bin **nicht** damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.
I do not consent to my thesis being made available in the university archive for third parties to view for academic purposes.

C) Einverständniserklärung zur elektronischen Überprüfung der Arbeit auf Plagiate
Declaration of consent for electronic checking of the work for plagiarism

Eingereichte Arbeiten können nach § 18 des Allgemeinen Teil der Bachelor- bzw. der Masterprüfungsordnungen der Universität Bremen mit qualifizierter Software auf Plagiatsvorwürfe untersucht werden.
Zum Zweck der Überprüfung auf Plagiate erfolgt das Hochladen auf den Server der von der Universität Bremen aktuell genutzten Plagiatssoftware.

Submitted papers can be checked for plagiarism using qualified software in accordance with § 18 of the General Section of the Bachelor's or Master's Degree Examination Regulations of the University of Bremen. For the purpose of checking for plagiarism, the upload to the server is done using the plagiarism software currently used by the University of Bremen.

☑ Ich bin damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum oben genannten Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

I agree that the work I have submitted and written will be stored permanently on the external server of the plagiarism software currently used by the University of Bremen, in a library belonging to the institution (accessed only by the University of Bremen), for the above-mentioned purpose.

☐ Ich bin **nicht** damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

I do not consent to the work I submitted and wrote being permanently stored on the external server of the plagiarism software currently used by the University of Bremen, in a library belonging to the institution (accessed only by the University of Bremen), for the above-mentioned purpose.

Das Einverständnis der dauerhaften Speicherung des Textes ist freiwillig. Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden. Weitere Informationen zur Überprüfung von schriftlichen Arbeiten durch die Plagiatsoftware sind im Nutzungs- und Datenschutzkonzept enthalten. Diese finden Sie auf der Internetseite der Universität Bremen.

Consent to the permanent storage of the text is voluntary. Consent can be withdrawn at any time by making a declaration to this effect to the University of Bremen, with effect for the future. Further information on the checking of written work using plagiarism software can be found in the data protection and usage concept. This can be found on the University of Bremen website.

Mit meiner Unterschrift versichere ich, dass ich die obenstehenden Erklärungen gelesen und verstanden habe und bestätige die Richtigkeit der gemachten Angaben.

With my signature, I confirm that I have read and understood the above explanations and confirm the accuracy of the information provided.

Datum / Date Unterschrit/ Signature

Universität Bremen

Schriftliche Arbeiten

Eigenständigkeitserklärung und Einverständniserklärung zur Überprüfung mit Plagiatssoftware

Studierenden-Angaben

Matrikelnr. **6256157**

Nachname **Rallabandi**

Vorname **Satya Sree Rupa**

Titel der Arbeit:

Minimum Viable Systems Engineering (MVSE) for Rapid Space-System Development

A) Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet, dazu zählen auch KI-basierte Anwendungen oder Werkzeuge.
Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht.

☑ Ich habe KI-basierte Anwendungen und/oder Werkzeuge genutzt und diese im Anhang "Nutzung KI basierte Anwendungen" dokumentiert.

B) Einverständniserklärung zur elektronischen Überprüfung der Arbeit auf Plagiate

Eingereichte Arbeiten können nach § 18 des Allgemeinen Teil der Bachelor- bzw. der Masterprüfungsordnungen der Universität Bremen mit qualifizierter Software auf Plagiatsvorwürfe untersucht werden.
Zum Zweck der Überprüfung auf Plagiate erfolgt das Hochladen auf den Server der von der Universität Bremen aktuell genutzten Plagiatssoftware.

☑ Ich bin damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum oben genannten Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

☐ Ich bin nicht damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

Das Einverständnis der dauerhaften Speicherung des Textes ist freiwillig.
Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden.
Weitere Informationen zur Überprüfung von schriftlichen Arbeiten durch die Plagiatsoftware sind im Nutzungs- und Datenschutzkonzept enthalten. Diese finden Sie auf der Internetseite der Universität Bremen.

Mit meiner Unterschrift versichere ich, dass ich die obenstehenden Erklärungen gelesen und verstanden habe und bestätige die Richtigkeit der gemachten Angaben.

Datum 15.04.2026 Unterschrift