TNO innovation for life

Archipel Study Report

The interplay between Systems- and Software-Engineering

TNO Intern ⟩ TNO 2024 R11856

18 February 2025

ICT, Strategy & Policy
www.tno.nl
+31 88 866 50 00
info@tno.nl

TNO 2024 R11856 – 18 February 2025

Archipel Study Report

The interplay between Systems- and Software-Engineering

Author(s)	Richard Doornbos, Pieter Goosen, Johan Lukkien, Joana Teixeira, Alexandr Vasenev
Classification report	TNO Intern
Title	TNO Public
Report text	TNO Public
Number of pages	17 (excl. front and back cover)
Number of appendices	0
Project name	Archipel 2024
Project number	060.61213

Disclaimer
In the creation of this document, Microsoft 365 Copilot was used to improve the use of English. In all instances where Copilot was used, the authors thoroughly reviewed the results to ensure the end result is accurate and free from errors, thus allowing the authors to take responsibility for the content.

Acknowledgement
The research is supported by the Netherlands Organisation for Applied Scientific Research (TNO). The participation of several TNO-ESI partners (see https://esi.tno.nl/about-us/partners/) is sincerely appreciated.

Contents

Summary

The high-tech equipment industry in the Netherlands develops cyber-physical systems that are increasingly enabled by software. The interplay between Systems Engineering (SE) and Software Engineering (SWE) remains persistently problematic. Organisations report recurring symptoms such as integration issues, emergent-behaviour problems, rework, delays, and budget overruns. This study, conducted with researchers from four companies and four universities, investigated the underlying generative mechanisms driving these symptoms. Using causal modelling and structured workshops, the research reveals that the SE–SWE gap is systemic. It arises from complex reinforcing feedback loops involving organisational structures, misaligned development cadences, limited shared system knowledge, historical undervaluation of software and software engineering, siloed processes, etc.

The report identifies that many current industry initiatives address symptoms rather than causes, resulting in limited impact. To meaningfully improve the interplay, organisations must adopt a holistic portfolio of interventions that target structural, behavioural, and perception-level factors simultaneously. Key recommendations include involving software engineering in strategic decision-making, aligning architectural decompositions, synchronising workflows and cadences, investing in cross-disciplinary training, redesigning organisational structures to reduce distance, enabling safe-to-fail learning environments, and co-creating system requirements.

Finally, the study provides guidance on researching sociotechnical systems through co-creation, causal reasoning, and multi-stakeholder workshops. It concludes with directions for future research aimed at strengthening engineering systems and improving long-term organisational learning.

1 Introduction

1.1 Motivation

In the high-tech industry, the interplay between Systems Engineering (SE) and Software Engineering (SWE) is often perceived as problematic and persistent. Problems are mismatches between SE and SWE in a broad sense, with symptoms such as: issues with emergent behaviour and qualities of (integrated) products, errors, rework, performance problems, delays, budget overruns and undesired results in general. The problem is also recognized in literature as persistent for some time [1].

In 2023, TNO-ESI and Thermo Fisher Scientific conducted a research project on the SE–SWE interplay. Because universities play a key role in competency development (and in shaping mental models), and because the interplay challenge is shared across multiple companies, it became clear that more companies and universities needed to be involved to enable structural improvements. In 2024, the Archipel study was conducted by researchers from four companies and four universities, covering the relevant engineering backgrounds.

1.2 Research Questions

The following Research Questions (RQs) are addressed in this report:
RQ1: Why is the SE–SWE interplay perceived as problematic in industry? What are the observable symptoms and underlying causes? This is discussed in Chapter 3 (Problems and challenges), particularly Section 3.1.
RQ2: What are current industry initiatives to address the situation? In Chapter 3.4, current industry initiatives are addressed.
RQ3: How can the interplay be improved? This is the central section of the study and Chapter 4 describes how it can be improved.

Conclusions, answers to the research questions and proposed future research directions are given in Chapter 5.

2 Methodology

To answer the research questions, the following world view and approach was used.
The SE–SWE interplay is a complex socio-technical system, and therefore must be studied using methods that reveal generative mechanisms, interdependencies, and system-level behaviour, rather than focusing on symptoms or isolated activities. This study consisted of four workshops with structured reflection and causal reasoning [2].

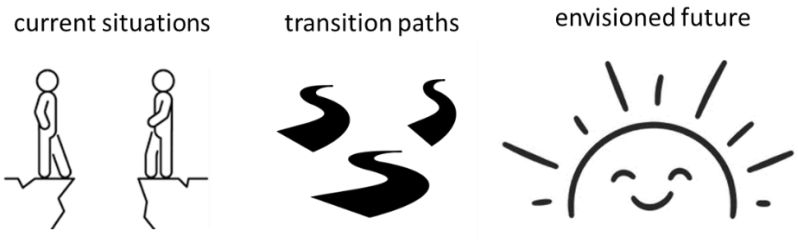


Figure 1 The main concepts to researching a sociotechnical problem situation.

Workshop 1.
In workshop 1, causal models were explicitly chosen as the “reasoning frameworks” to understand current situations, ideal situations, and interventions, because they expose generative mechanisms and potential unintended consequences of actions.

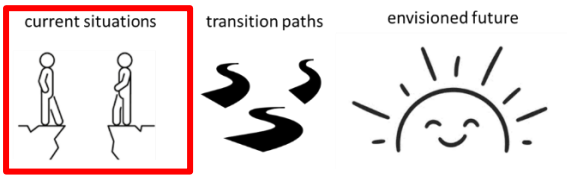
Workshop 2.
Workshop 2 used the Appreciative Inquiry method [3] to envision “what might be,” to broaden the solution space and encourage systemic rather than incremental thinking.

Workshop 3.
Develop and compare interventions using an intervention framework [4] that accounts for actors, actions, leverage points, potential unintended effects, required resources and feedback mechanisms.

Workshop 4.
As a final workshop, two industry cases were selected, analysed and developed into plans that integrate the results from the previous workshops.

3 Problems and challenges

This chapter describes the current situation, i.e. the problem description, and why it is a complex problem.



3.1 Problem description (symptoms)

In the high-tech industry, the interplay between Systems Engineering (SE) and Software Engineering (SWE) is often perceived as problematic. SE is here defined as the "transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems using systems principles and concepts and scientific, technological and management methods" [5], SWE is "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" [6]. Problems are mismatches between SE and SWE in a broad sense, with symptoms such as: issues with emergent behaviour and qualities of (integrated) products, errors, rework, performance problems, delays, budget overruns and undesired results in general. The problem has been studied for decades [1] [7] [8] [9] [10] [11] [12], ever since software (SW) became a dominant factor in the realisation of high-tech products.

The interplay is sometimes called the *chasm* or the *gap* between those disciplines [9]. Studies, like Pyster et al. [1] describing aspects of the problem and their possible causes and solutions are published regularly. Typical mentioned problem sources include ineffective communication, lack of understanding of each other's domains, and unsynchronized development processes.

In 2023, a collaborative research initiative was undertaken by TNO-ESI and Thermo Fisher Scientific to examine the interplay within Cyber-Physical System (CPS) development. Subsequent discussions with ESI partners confirmed that the observed challenges are widely acknowledged and commonly encountered across the industry. The 2023 study with Thermo Fisher focused on the unique complexities inherent to CPS development, as also highlighted in prior research listed above [13]. The severity of these challenges varies significantly among participating organisations. A primary differentiating factor is the stage of organisational transition—from a predominantly hardware-oriented approach within physical domains to a more balanced integration of hardware and software characteristic of cyber-physical systems. In early transition stages, the imbalance was exemplified by the absence of software-driven value drivers in certain R&D roadmaps, while in late stage of transition the imbalance was considered insignificant.

3.2 The interplay as a complex problem

It seems not all organizations experience the problem, and not in the same way. The problem is by no means just technical, but technology is often the driver and defines the goal of the interplay, which is, to deliver (software-intensive) technical products (the engineered system). The engineering system, that produces this engineered system, is a complex system (as

defined by [15]). Martin [14] refers to these as the realization system (S3) and intervention system (S2). The engineering system is complex because it needs to holistically integrate individuals into teams that need to integrate multiple disciplines, each with different world views, processes and heartbeats (rhythms in production, i.e. SWE favours shorter agile sprints than SE). In complex systems, observed symptoms are generated by hidden mechanisms.

3.3 The interplay during product development projects

Causal modelling was used to reason about these generative mechanisms and the results are described next.

A detailed study in 2023 [16] at one company experiencing successful and less successful projects resulted in the causal model shown in figure 2. This model could be generalized to all the study partners. For guidance on interpretation of the graph elements, please refer to [2]

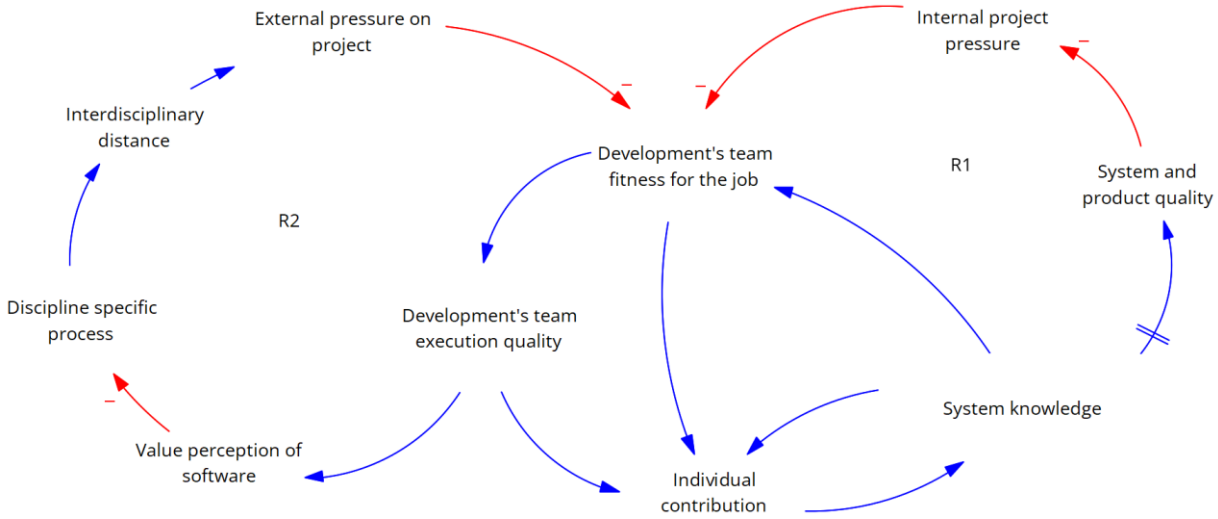


Figure 2 A causal model showing the factors in the interplay and their connections

The model can be divided into development team related factors (R1) on the right and organisational/context related factors (R2) on the left.

- R1: Development Team related causes
  The development team's fitness for the job is often reduced due to SWE not being included or not enabled to start working, reducing the individual's contribution. When excluded, the individual also does not build up the required system knowledge and the system knowledge reduces/excludes the individual's contribution. Once the individual disciplines' contributions don't come together in the product, the product quality (observable consequences) is lacking and a lot of pressure is put on the team to improve the quality, resulting in delays referred to as the teams' execution quality (again, an observable consequence).

- R2: Context related causes
  Low team execution quality results in negative perceptions of SWE and organizational interventions to improve that specific discipline. This can lead to more silo behaviour or even organisational silos/isolation, and an increased distance between disciplines instead of better collaboration. This external pressure reduces the team's fitness for the job even further.

The consequences are represented as two main observable effects:

- System/Product quality: integration problems, costly rework, reduced product functionality or performance.
- Development team's execution quality: delays, poor cost and effort estimations, not achieving contracted deadlines, and penalties.

3.4 Current industry initiatives

In this study, it was observed that the participating companies primarily address symptoms rather than underlying causes, and that the interplay therefore remains problematic.

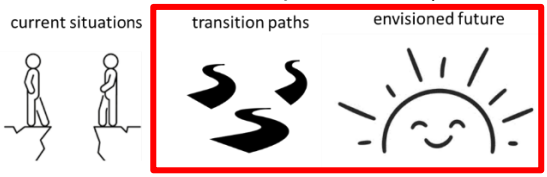
Some of the observed initiatives focussed on:
Communication and coordination- In successful project cases, the role of a core team consisting of system architect, product owner and project manager, was identified as key to the project success. This team formed a bridge between disciplines, communicating market and system-level views, synchronising milestones and integration points and adjusting accordingly.

SWE in management positions- Companies that did not recognize the symptoms were further down the transition path and earlier identified the role of SW in CPSs, as reflected in management positions and processes.

SWE processes and practices- Interventions included efforts to improve SWE practices and quality and introduction of Scaled Agile Framework (SAFE) across the enterprise.

4 Improving the interplay towards an ideal future

This section discusses approaches and opportunities to improve the interplay towards an ideal future and lists nine possibilities (this is not an exhaustive list)



Improving the interplay between Systems Engineering (SE) and Software Engineering (SWE) requires more than isolated adjustments or discipline-specific optimisations. The gap between SE and SWE is **systemic** in nature: it is produced, reinforced, and sustained by interacting feedback loops across organisational structures, mindsets, workflows, and historical practices. Because of this, a **holistic portfolio of interventions** is essential—only coordinated, multi-leverage actions can influence the generative mechanisms driving the persistent symptoms.

At the heart of the problem lies the fact that the challenges observed in industry are not caused by single faults but by **interacting feedback loops**. The causal model developed in this study shows that delays, integration issues, rework, and performance shortfalls all emerge from reinforcing dynamics between organisational perceptions, team fitness, system knowledge, and development cadences. Addressing one visible symptom—such as fixing an integration delay—does little to influence the underlying structure that produces repeated occurrences of the same issue. As systemic theorists note, modifying individual process steps provides only limited leverage, whereas altering the feedback structure of the system has far greater impact [4].

Another central insight from the Archipel study is that the interplay is fundamentally a **sociotechnical system**, not a simple coordination problem between two technical disciplines. It spans technology, business models, educational backgrounds, organisational roles, power structures, and psychological safety. Sociotechnical systems develop **path dependencies**, where past decisions shape current behaviour; **delays** between cause and effect make outcomes hard to predict; and **local optimisations** often obscure global improvements. Moreover, behaviour emerges from deeply held **mental models**, not from written procedures or tools. This means the highest leverage lies in influencing mindsets, shared goals, and the rules of the organisational system, rather than adding further process artefacts or refining local practices [4].

The risks of narrow interventions become clear when single-point changes are attempted. Focusing on SWE as “the problem,” for example, can unintentionally increase organisational

distance and reinforce silo behaviour, which then reduces team fitness and degrades product outcomes even further. Similarly, introducing new process templates or enforcing strict role boundaries can inadvertently freeze collaboration rather than improve it. Complex systems often produce counterintuitive results: well-intentioned fixes can shift the burden elsewhere, create unintended resistance, or strengthen the very problem they sought to solve. Only a coordinated portfolio of interventions can minimise such risks by balancing structural, behavioural, and perception-level factors at the same time.

The study also revealed that the core issues manifest across **multiple system layers simultaneously**. At the highest level, academia and industry operate on different cadences and feedback loops, shaping engineers' mindsets long before they enter organisations. At the organisational level, incentive structures, governance, and budgeting practices influence how SE and SWE are valued and when they are involved. At the project level, day-to-day practices around knowledge sharing, workflow synchronisation, and system decomposition shape how effectively teams collaborate. A quick fix applied to only one layer cannot resolve the structural misalignments across the others. A holistic portfolio, by contrast, operates across layers—aligning education with industry needs, synchronising organisational incentives with project realities, and improving architectural practices alongside team-level collaboration.

Finally, the SE–SWE interplay is shaped by **historical forces and organisational inertia**. Decades-old perceptions—such as the belief that software is "free" or secondary to hardware—continue to influence pricing, staffing, and roadmapping decisions, even as modern products have become software-intensive. Patterns of underrepresentation of SWE in early decision-making stages persist, reinforcing unequal influence across the lifecycle. Complex systems rarely "reset" on their own; changing their trajectory requires sustained pressure at several leverage points, applied consistently over time. A holistic intervention portfolio enables precisely that kind of coordinated, sustained change.

In summary, **systemic change requires consistency, alignment, and reinforcement across multiple interventions**. Meadows [4] refers to this as a "portfolio of congruent actions," where improvements in structure, information flows, mindsets, and goals all support one another. Only by addressing the SE–SWE interplay as a multi-layered, sociotechnical system—rather than as a sequence of isolated coordination problems—can organisations strengthen the resilience and long-term effectiveness of their engineering systems.

For the final workshop, two use cases were selected and each use case developed some interventions valuable for that company's context. The following interventions were identified as being feasible and promising. This is not an exhaustive or scientifically validated list of interventions or recommendations.

4.1 Steer on holistic systems level causes, not local symptoms

A recurring insight is that many symptoms—delays, rework, quality issues—are just the *visible iceberg (above the water)* of deeper, generative mechanisms as depicted in the causal model in **Error! Reference source not found.**.

Intervention portfolio element:

- Use causal modelling as an *ongoing diagnostic and management tool*, not a one-off exercise, to reveal non-obvious leverage points and avoid treating symptoms. Continuously update the model with the new insights gleaned from the emerging effects of interventions.

This focusses on application of causal modelling for continuous improvement in complex systems.

4.2 Embed SW structurally into strategic business decision making

The roots of the SE–SWE problem are partially **business-structural**: SW is undervalued, underpriced, considered “free”, or not included early enough in strategic planning.

Intervention portfolio element:

- Treat SW as a *value creator*, not a cost centre.
- Link pricing and product strategy to lifecycle cost, maintainability, and upgradeability towards an ideal future where SW features and non-quality cost is recognised, valued and planned.
- Make SW leadership structurally part of portfolio decisions, solution initiatives, and sales phases.

This is focussed on improving the causal model factor “Value perception of software” in R2.

4.3 Align architectures and decomposition

A major generative mechanism is the mismatch between SE decomposition (functions, architectures) and SWE decomposition (features, iterative refinement).

Intervention portfolio element:

- Joint architectural decomposition sessions led by multidisciplinary architects (SW + SE).
- Establish common architectural languages and modelling approaches (e.g., SysML-based shared views) to allow high-level alignment without forcing uniform detail levels.
- Introduce SW system architects operating at “high decomposition levels” to reduce the abstraction gap.

This is focussed on improving the closely connected R1 factors in the causal model “Development team’s fitness for the job, individual contribution and system knowledge”

4.4 Synchronise workflows and cadences across disciplines

Desynchronised cadences create recurring systemic issues such as SW being “late,” missing hardware inputs/test platforms required for SW development, or incompatible expectations.

Intervention portfolio element:

- Co-located, small interdisciplinary teams with shared ownership and joint priority-setting.
- Joint sprinting or quarterly alignment cycles that respect differences in HW/SW iteration speeds.
- Shared backlog ownership and transparent decision-making.

This reduces the causal model factor in R2 called “interdisciplinary distance”.

4.5 Invest in shared system knowledge and cross-disciplinary training

Lack of mutual understanding is a core systemic cause highlighted repeatedly.

Intervention portfolio element:

- Cross-disciplinary training that considers discipline specific mindsets and preferences: “SE for SWE” and “SWE for SE”
- On-the-job coaching using real product examples
- Long-term investment to grow multidisciplinary architects

This is focussed on improving the closely connected R1 factors in the causal model “Development team’s fitness for the job, individual contribution and system knowledge”**.**

This reduces the causal model factor in R2 called “interdisciplinary distance”.

4.6 Redesign organisational structures to reduce distance

Organisational distance is a major factor identified as leading to sustained negative outcomes.

Intervention portfolio element:

- Introduce bridging roles or core teams explicitly responsible for cross-disciplinary communication (ideal future).
- Reduce hierarchy-based separation by embedding SW earlier in system design and reducing “throwing over the wall” behaviour
- Promote SW representation at all organisational levels

This is focussed on improving the “interdisciplinary distance” in R2 as well as the closely “Development team’s fitness for the job” in R1.

4.7 Create safe-to-fail innovation environments

Innovation pressure typically obstructs integration-level learning; safe-to-fail “sandbox” environments support cross-disciplinary experimentation and mutual learning.

Intervention portfolio element:

- Establish controlled spaces where SE, SWE, and HW engineers prototype together.
- Use sandbox learnings to improve processes outside the sandbox (e.g., tooling trials, new WoW).

This acts on the “external pressure on the project” factor in R2.

4.8 Develop a long term learning organisation

Several participants explicitly surfaced the need to transform into a learning organisation.

Intervention portfolio element:

- Make causal modelling and reflective reasoning a part of normal operations (not just project-specific).
- Document successes and failures of interventions.
- Periodically revisit intervention impact.

This focusses on application of causal modelling for continuous improvement in complex systems.

4.9 Co-create requirements and system definitions

One of the most actionable interventions is joint SE–SWE creation of “just enough” high-level requirements and specifications.

Intervention portfolio element:

- Replace sequential requirements with iterative SE+SWE discovery cycles.
- Define system-level uncertainties early and convert them into risk and cost structures

This positively reinforces the R1 factors related to individual’s contributions to system knowledge, the individual’s knowledge of the system and the team’s fitness for the job and prevents disengagement while waiting for requirements. Ultimately the development team’s execution quality improves.

5 Conclusions

In conclusion, the research questions or study objectives can be answered as follows:

5.1 Research Questions

RQ1: Why is the SE–SWE interplay perceived as problematic in industry? What are the observable symptoms and underlying causes?
The interplay is problematic (see Chapter 3.1) because organisations consistently observe symptoms such as integration issues, emergent behaviour problems, rework, delays, performance shortfalls, and budget overruns. These symptoms stem from deeper underlying causes: misaligned development cadences, limited shared system knowledge, organisational distance between disciplines, reinforcing feedback loops that reduce team fitness, and siloed processes that prevent early, effective SWE involvement. The causal model describes these causes and how they are connected to form (reinforcing) loops.

RQ2: What are current industry initiatives to address the situation?
Companies attempt to address the interplay through communication and coordination improvements, the formation of cross-disciplinary core teams and process-level interventions such as improving SWE quality practices and adopting scaled-agile frameworks (as discussed in Chapter 3.4). However, these initiatives often target symptoms rather than systemic generative mechanisms.

RQ3: How can the interplay be improved?
Improvement requires holistic, multi-leverage interventions because the SE–SWE gap is produced by interacting feedback loops across technical, organisational, and human-factor dimensions (Chapter 4). Single fixes cannot address misaligned architectures, incentive structures, workflows, mindsets, and organisational separation—each of which reinforces the gap. A coherent intervention portfolio must act across multiple system layers, embedding software into strategic decisions, aligning architectural decomposition, synchronising cadences, reducing organisational distance, investing in cross-training, enabling safe-to-fail learning environments, and fostering long-term learning behaviour.

5.2 Future research

The following research directions should be explored further:

- We now understand the socio-technical aspects for the interplay for this sample of participants, how do we incorporate that into engineering systems (called the realization system (S3) in [14]) improvements in terms of methodologies, processes and tools?
- What are KPIs relevant to monitoring improvements in complex socio-technical systems? What are key indicators of tipping points/changes towards the ideal future?
- How to go from a small group to the larger organization (beyond SE and SWE disciplines)?

- The causal model for product development identifies several aspects of the interplay that each could form a research topic. Similarly, the relationships between aspects could be researched.
- Explore how differences outside of new product development projects (e.g. during bug fixing or platform creation), between the engineering disciplines affect the interplay, e.g. software technical debt is reported to be between 23 to 42% of development time [18].
- Outside of companies, academia is playing an important role in shaping mindsets and interdisciplinary competencies that could improve the interplay at mental model level. How can academia contribute to improving the interplay?
- How can the research methodology be improved to make it more effective and efficient, especially with regards to causal modelling?

6 References

[1] Pyster, A., et al. (2015). Exploring the relationship between systems engineering and software engineering. *Procedia Computer Science*, *44*, 708 6717.

[2] van Gerwen, E., & Lukkien, J. (2025). *Causal modelling: Tools for high-tech industry*. TNO.

[3] Dash Bhatt, A., & Singh, S. (2025). Appreciative inquiry: A systematic review and future research agenda. *Journal of Organizational Change Management*, *38*(3), 66483.

[4] Meadows, D. (1999). *Leverage points: Places to intervene in a system*. The Sustainability Institute.

[5] International Organization for Standardization. (2023). *Systems and software engineeringSystem life cycle processes* (ISO/IEC/IEEE 15288). Author.

[6] IEEE Computer Society. (2025). *Guide to the software engineering body of knowledge (SWEBOK)*.

[7] Aurum, A., Biffl, S., Boehm, B., Erdogmus, H., & Grfnbacher, P. (2005). *Value-based software engineering*. Springer.

[8] Baldwin, K. (2007). *DoD software engineering and system assurance: New organization, new vision*. https://apps.dtic.mil/sti/citations/tr/AD1014788

[9] Boehm, W., Henkler, S., Houdek, F., Vogelsang, A., & Weyer, T. (2014). Bridging the gap between systems and software engineering by using the SPES modeling framework as a general systems engineering philosophy. *Procedia Computer Science*, *28*, 187194.

[10] Maier, M. W. (2006). System and software architecture reconciliation. *Systems Engineering*, *9*(2), 146159.

[11] Muscarella, S., Osaisai, M., & S. S. (2020). Systems and software interface survey. In *INCOSE International Symposium* (*30*(1), 13051323).

[12] Sheard, S. A., Cadigan, J., Marvin, J., Chim, L., & Pafford, M. E. (2018). INCOSE working group addresses system and software interfaces. *INSIGHT*, *21*(3), 6271.

[13] Vasenev, A., Lukkien, J., van Veen, L., Goosen, P., Doornbos, R., & Mooij, A. (2023). Obtaining insights into the interplay between systems and software engineering. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 8788).

[14] Martin, J. N. (2004, June). The seven samurai of systems engineering: Dealing with the complexity of 7 interrelated systems. In *INCOSE International Symposium*.

[15] Snowden, D. J., & Boone, M. E. (2007). A leader's framework for decision making. *Harvard Business Review*.

[16] R. Doornbos, A. Vasenev, J. Lukkien, L. van Veen and P. Goosen, “The interplay between systems engineering and software engineering (doc no 2024 R10291),” TNO, 2024.

[17] D. Kim and C. Lannon, “A pocket guide to using the archetypes”.

[18] CodeScene, “Business costs of technical debt,” 2023.

ICT, Strategy & Policy

High Tech Campus 25
5656 AE Eindhoven
www.tno.nl

TNO innovation for life