Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10632

Managing technical debt at Ubisoft IT: interfaces and change propagation in engineering systems interventions

Jose Bonet Faus[1,2,✉], Pascal Le Masson[1], Benoît Weil[1] and Antoine Bordas[1]

[1] Mines Paris - PSL, France, [2] Ubisoft, France

✉ pbonetfaus@gmail.com

ABSTRACT: Technical Debt (TD) is a buzzword that has gained traction among IT practitioners and researchers, as a financial metaphor explaining how existing elements can hinder future changes. In engineering design terminology, managing TD can be equated to managing the redesign of existing systems, a task in which change propagation plays a key role. Via a partnership with Ubisoft, a game development firm that owns and operates a global IT network, we present a case study showcasing a novel way to explore interfaces and propagations in engineering systems interventions, analyzed using Axiomatic Design.

KEYWORDS: technical debt, engineering systems interventions, change propagation, interface management, design theory

1. Introduction: technical debt as a design challenge in IT

Interconnected engineering systems form the fabric of our world (Maier et al., 2022). In the current context of digitalisation, increasing complexity and increasing reliance on software, IT systems and infrastructure have become a foundational part of said fabric.

In IT and Software Engineering circles, a buzzword has been gaining traction among practitioners and researchers: Technical Debt (TD). TD is a financial metaphor that describes existing system elements that can negatively influence future design efforts as a form of debt (Li et al., 2015). This metaphor has the virtue of framing design challenges in business terms, enabling engineers to express their concerns in a way that is accessible to non-technical decision-makers. In doing so, it helps managers recognize that certain design decisions and implementation constructs create conditions where modifying a system demands greater effort. These elements must therefore be identified, budgeted, prioritized, and redesigned in due course (Avgeriou et al., 2016; McConnell, 2013).

The challenges of conceptualizing and managing TD were initially approached through the lens of Software Engineering. Consequently, work on source code using incremental and iterative approaches is prevalent in existing literature (Ernst et al., 2015; Li et al., 2015). The concept of TD has since evolved beyond "pure" software systems to include broader engineering contexts. This transition shifts attention from code-level implementation to systemic aspects of design, encompassing the influence of existing architectures, processes, and requirements on future change. These forms of debt, referred to as "high-level" TD, are widely acknowledged as highly impactful and cannot be addressed through traditional code-centric approaches. Facing these difficulties, emerging TD research promotes the development of systemic models and methods capable of supporting sound design over complex existing structures (Ciolkowski et al., 2021). This profoundly resonates with existing literature on engineering systems design, thus framing the challenge of managing TD in complex systems as the following engineering design research question: ***What models of action are suited to effectively manage the redesign of an existing engineering system?*** This challenge remains unresolved, particularly when the system to be redesigned is highly complex and evolves amid uncertain contexts (Eckert & Clarkson, 2023).

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

Section 2 reviews how existing literature addresses the redesign of engineering systems. It emphasizes the role of change propagation in engineering systems interventions and highlights limitations in current approaches. A research question is formulated accordingly. Section 3 outlines how data was collected at Ubisoft IT, and proposes an analytical framework to extract results from said data. Section 4 presents Ubisoft IT's context in detail and describes the case study's empirical material using Axiomatic Design. Section 5 reports the results extracted from the empirical material and addresses the research question. Lastly, Section 6 concludes with a discussion of the findings and perspectives for future research.

2. Interfaces and change propagation in engineering systems design

As previously mentioned, the challenge of redesigning existing systems is heightened by complexity and uncertainty. This is particularly relevant for engineering systems, which *"by a strict understanding [. . .] are large scale global systems [. . .] comprised of many constituent systems, products, and services"* (Maier et al., 2022). The term is used to describe global infrastructure systems such as transportation, energy and telecommunication networks, but also applies to "local" examples like autonomous vehicles, production lines or datacenters. These systems are characterised by long and uncertain life cycles, and dynamic interactions with their environments. As such they are *"partially designed and partially evolved"* (de Weck et al., 2011), resulting in a great deal of technical and organisational complexity.

2.1. Engineering systems interventions

These attributes call for approaching their redesign through a specific lens, that of engineering systems interventions (Maier et al., 2022). Figure 1 presents a model of such interventions, depicted as local actions intended to influence a broader existing system: an artefact is implemented within a target system that is part of a broader system of interest, to improve the performance of a target process (Bots, 2022).

System of interest	Larger system to which we want to make an improvement or in which we want to reduce harm
Measure of performance	Specific metric that the "client" seeks to improve
Target process	Particular process related to the measure of performance that the "client" seeks to improve
Target system	Subsystem within the system of interest that is selected by "the engineer" after an analysis aimed at diagnosing which subsystem most constrains the targeted measure of performance.
Artefact	Artificial structure to be implemented with the goal of bringing the desired improvement to the target process
Contingent processes	Processes outside the target system whose performance is affected by the intervention
Externalities	Impacts, positive or negative, on processes outside the scope of the client's system of interest

Figure 1. Engineering systems interventions model, extracted from (Bots, 2022)

Other concepts from Figure 1, such as contingent processes and externalities, highlight the systemic nature of interventions. This perspective echoes a long line of research, dating back to the 1990s, which established that design inherently brings about change, regardless of the motivation behind it (correcting errors, adapting to new constraints, introducing new functionalities . . .). Such changes inevitably propagate throughout a system, which can lead to unexpected outcomes that compromise overall system performance. In such situations cost, time, and effort can quickly spiral out of control, thus calling for design approaches that are attentive to propagation effects (Brahma & Wynn, 2023). The use of models that represent objects by emphasizing connections, interfaces, and interdependencies, such as Design Structure Matrices (DSMs) (Eppinger & Browning, 2012) and Axiomatic Design (AD) (Suh, 2001), help understand propagations. The descriptions generated through such models provide a foundation for building sophisticated Change Propagation Analysis (CPA) methods that aim to understand and predict how change spreads along the elements of a system (Clarkson et al., 2004). If an interface is defined as *"a functional or physical relation between two mating system elements across which interaction may occur"* (Parslov & Mortensen, 2015), the aforementioned methods are based on the designer's ability to describe the behaviour of interfaces when changes are introduced. However, the inherent complexity of engineering systems implies that the behaviour of all interfaces under change cannot be known in its entirety. Therefore, from the designer's perspective, an engineering system inevitably contains two kinds of interfaces. On one hand, known interfaces that can be explicitly described and formally modelled.

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

On the other hand, unknown interfaces, whose existence is not known a priori. In some cases where the designer already possesses an advanced level of understanding, they may be aware of such unknown interfaces but remain uncertain about their properties.

2.2. Approaches to manage change propagation

Engineering systems design brings forward two approaches to manage change propagation along the interfaces described above. A *"predict and control"* approach, focusing on rigor and standardization, and a *"prepare and commit"* approach, embracing contextual variability and emphasizing flexibility (Bots, 2022). These approaches originated in project management literature regarding engineering projects with notable complexity and uncertainty (Koppenjan et al., 2011). This body of work acknowledges that both represent one-sided extremes that fail to capture real management challenges, and instead advocates for a balanced integration of the two overarching philosophies.

In practice, however, the principles of systems engineering remain deeply rooted in front-end analyses conducted during the initial design phases. Thus, they are firmly grounded in the "predict and control" approach. For instance, methods and models such as DSMs, AD, or CPAs rely on the assumption that the design process progressively reveals all interface characteristics, allowing the designer to model them in detail. This entails a strong underlying hypothesis: that all interfaces are, in principle, knowable and can ultimately be controlled.

However, if this assumption were true, and design standards were applied correctly, common issues observed in large engineering projects, such as budget overruns and schedule delays, would not occur. Yet these issues are observed, indicating that established approaches fail to prevent difficulties they should theoretically address. This suggests the existence of additional phenomena that are not accounted for. The hypothesis brought forward in this paper is that these difficulties arise from the existence of unknown interfaces. If such interfaces are present, propagations along them should also be labelled as "unknown". Given the extreme relevance of change propagation for the redesign of engineering systems, the next natural step is to analyse how experimentation on unknown interfaces and their associated propagations is conducted within the context of an engineering systems intervention.

2.3. Two archetypes of experimentation in engineering systems interventions

The model depicted in Figure 1 does not make an explicit distinction between known and unknown interfaces. However, if one attempts to move beyond the assumption that "all interfaces are known" and to establish such a distinction, two major moments of propagation experimentation can be identified.

A first stage focused on the front-end diagnosis of known interfaces, conducted early in the intervention process, roughly corresponding to the first stages of the V-model's downward slope (Forsberg & Mooz, 1992). This exploration of knowable propagations directly appears in Figure 1 through the engineer's diagnosis aimed at identifying the "target system", defining requirements, metrics, tolerances and designing the "artefact" accordingly.

A second stage concerning the discovery through action of unknown interfaces, that often takes place late in the intervention process, when the new artefact is constructed and deployed within the system of interest (roughly corresponding to the later stages of the V-model's upward slope). This phase is almost absent from the model presented in Figure 1, though it can be vaguely discerned through the notion of "contingent processes". The adjective "contingent" suggests that the designer may suspect potential propagations, but the behaviour of corresponding interfaces between the target system and the rest of the existing environment, at the time of the engineer's diagnosis, is unknown. Unanticipated propagations related to these interfaces only become apparent once the artefact is introduced in real-life conditions, and must then be handled urgently.

These two moments thus reveal an underlying sequential approach: a front-end diagnosis and treatment of known interfaces, followed by a painful discovery of unknown interfaces, which correspond to two distinct models of action: two archetypes of experimentation.

The front-end diagnosis phase follows an archetype carrying the logic of "scientific proof": hypotheses about interface behaviour and propagations are formulated and systematically validated, with the primary goal of learning. Such experiments are conducted early in interventions, typically within spaces separated from the main system, and carry relatively low risk. This kind of experimentation of known interfaces aligns closely with the "predict and control" mindset. Engineering design literature explores it extensively, as illustrated by work on prototypes (D'Amelio et al., 2011), proofs of concept

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

(Jobin, 2022), Technology Readiness Level assessments (Olechowski et al., 2020), CPAs (Brahma & Wynn, 2023), or digital twins (Mercat et al., 2025) among many others.

The second phase instead follows an "urgent repair" archetype: unknown interfaces are uncovered through experimentation in real-world conditions, resulting in unanticipated propagations that emerge in real time and must be handled urgently. The primary aim becomes incident resolution, which involves a great deal of risk, thus relegating learning about newly uncovered propagations and critical interdependencies to a secondary spot. This kind of "firefighting" approach is much harder to find in engineering systems literature and is weakly aligned with the "predict and control" philosophy, but is commonly discussed in manufacturing literature in the context of commissioning and production ramp-up (Terwiesch & E. Bohn, 2001), as well as in software engineering debugging and defect resolution (McCauley et al., 2008).

It then becomes apparent that the experimental regimes corresponding to these two archetypes are far from equivalent; they differ greatly in their level of codification. On one hand, known interfaces are addressed within a well-established regime, highly codified through scientific and engineering epistemology. On the other hand, unknown interfaces are addressed within a second, much less recognized regime that is poorly codified and therefore not yet a true "archetype".

2.4. Research gap and research question

In this literature review, we have shown that engineering systems design addresses the redesign of complex existing systems through interventions. State-of-the-art intervention models are based on an underlying logic for managing change propagation along system interfaces in two phases: (1) An upstream phase, dealing with propagations along known interfaces and rooted in scientific validation, which is extensively explored in engineering design, followed by (2) a downstream phase, addressing propagations along unknown interfaces through an urgent repair approach, which remains marginally treated in engineering design. The first mode is well established and clearly defined, whereas the second remains poorly understood. This reveals an underlying gap concerning the effective management of unknown interfaces in real-world engineering systems interventions. This leads to the following research question: ***To what extent and in what ways can the existing two-phased model of action be enriched to better account for the management of unknown interfaces in systems engineering interventions?***

3. Methodology: Ubisoft IT as a single case study

Existing literature provides limited guidance on this exploratory research topic, which requires close examination of the technical and managerial complexities of a systems engineering intervention. To capture the multifaceted nature of discovering and managing unknown interfaces within interventions, a single case study in a research intervention setting (Aggeri, 2016) was deemed suitable. Case selection was informed by a preexisting technical audit of Ubisoft IT's systems portfolio, which identified the system presented in Section 4.2 as being in the most critical condition. Its recovery required addressing substantial architectural TD, making it a prime candidate for this study.

Data, including presentations, workshop minutes, meeting notes, specification documents, architecture decision records, and business cases (64 documents, 431 pages), were collected by the authors between October 2024 and November 2025. As a member of the project team, the first author participated in weekly working sessions (96 meetings, 120 hours), combining empirical data collection with project-related cost modelling tasks.

Axiomatic Design theory (AD) was used to code the system-related data, taking the form of a set of Functional Requirements (what is to be achieved, FRs) and Design Parameters (how it is achieved, DPs) for its subsequent analysis. Interdependencies between FRs and DPs are systematically examined and visualized using a matrix representation, shown in Section 4.2 (Suh, 2001). Data on the intervention process itself is coded via the engineering systems interventions model presented in Figure 1, which is expanded to operate on two levels. A first level that addresses the overall intervention process and a second level focusing specifically on change propagation and its practical experimentation, as depicted in Figure 2. For pedagogical purposes, Figure 2 also maps the elements of our expanded intervention model onto the terminology of AD. To consolidate findings, telemetry data from the affected IT systems were analysed over 170 days, alongside additional formal and informal interviews with project stakeholders, including IT architects, DevOps engineers, system administrators, business analysts, service owners, and

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

IT directors (16 participants, 45 interviews, 58 hours). To mitigate potential organizational bias, preliminary findings were reviewed bi-monthly by steering committees comprising senior academic supervisors and Ubisoft directors throughout the intervention period.

Level 1: Macro-level intervention		
System of interest	Larger system to which we want to make an improvement or in which we want to reduce harm	Full FR/DP Matrix
Target process	Particular process related to the measure of performance that the "client" seeks to improve	FR that we want to improve
Measure of performance	Specific metric that the "client" seeks to improve	Metric associated to the FR we want to improve
Target system	Subsystem within the system of interest that is selected by "the engineer" after an analysis aimed at diagnosing which subsystem most constrains the targeted measure of performance.	Set of DPs that exert the greatest influence on the metric associated with the FR we aim to improve (target system DPs)
Artefact	Artificial structure to be implemented within the target system with the goal of bringing the desired improvement to the target process	Change to be introduced to target system DPs.
Contingent processes	Processes outside the target system whose performance is affected by the intervention	Coupled cells along the columns related to target system DPs
Level 2-A: Propagation experimentation of known interfaces		
Target system known interfaces	Interfaces between the target system and other neighboring subsystems whose behaviour is known or knowable	Known cells in the target system DP(s)' column(s)
Propagation control strategies	Set of mechanisms put in place during the intervention to mitigate and contain propagations along known interfaces	Redesign of existing DPs to them from target system DPs
Level 2-B: Propagation experimentation of unknown interfaces		
Target system unknown interfaces	Interfaces between the target system and other neighboring subsystems whose behaviour is a known unknown	Unknown cells in the target system DP(s)' column(s)
Propagation response strategies	Set of mechanisms put in place during the intervention to build the capacity to respond to propagations along unknown interfaces, while minimally disrupting the target process and contingent processes	Use of transient DPs, over-engineering of new DPs, redirection of load to DPs with known behaviours…
Propagation learning strategies	Set of mechanisms implemented within the intervention to enhance the discovery and learning about propagations along unknown interfaces	Genericity, observability and measurability of FRs and DPs

Figure 2. Expanded model of engineering systems interventions. Intended as an analytical framework to describe data collected at Ubisoft IT

4. Exploring propagations in a real-world intervention at Ubisoft IT

Ubisoft is a world-class game development firm. Ubisoft's IT branch serves as the technological backbone, responsible for designing, maintaining, and operating a wide range of IT solutions for diverse users and purposes. These solutions support game production (including game engines, asset libraries, and version control tools used by developers and collaborators), online services (such as infrastructure for online gaming), corporate tools (including ERP systems, cybersecurity and threat management, and data governance), as well as datacenter hosting services for external clients.

4.1. Ubisoft IT technical heritage

Ubisoft IT is thus tasked with managing a substantial technical heritage encompassing all previously developed and acquired infrastructures, systems, and applications (Magnusson & Bygstad, 2014). This technical legacy exhibits several defining traits. It is complex, with a high number of interconnected components. It is multi-layered, ranging from infrastructure that sustains core business activities to end-user applications. It is distributed across 50+ Ubisoft sites in 20+ countries. It is inherited, built on legacy systems dating back to the early 21st century, and perennial, forming a long-lasting network where individual components are replaced but the whole is never truly decommissioned. It is perpetually online, supporting critical business functions that require near 24/7 availability, and dynamic, evolving to meet changing requirements. It is partially embodied, combining tangible physical elements such as datacenters, servers, and switches with intangible digital elements like virtual servers, APIs, and identity management systems. Taken together, these characteristics qualify it as an engineering system.

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

4.2. The Version Control System

The Version Control System (VCS) is an integral part of this technical heritage and constitutes the focus of our case study. Version control is a service provided by Ubisoft IT for Ubisoft production teams, meaning artists, designers and programmers. This service fulfils a key function: it stores and organizes all versions of files (code, textures, 3D models, audio . . .) produced during the years-long development of Ubisoft's marquee games like Assassin's Creed™. To give a sense of scale, the VCS stores millions of file versions every day, representing dozens of terabytes of data and requiring bandwidths of several gigabytes per second.

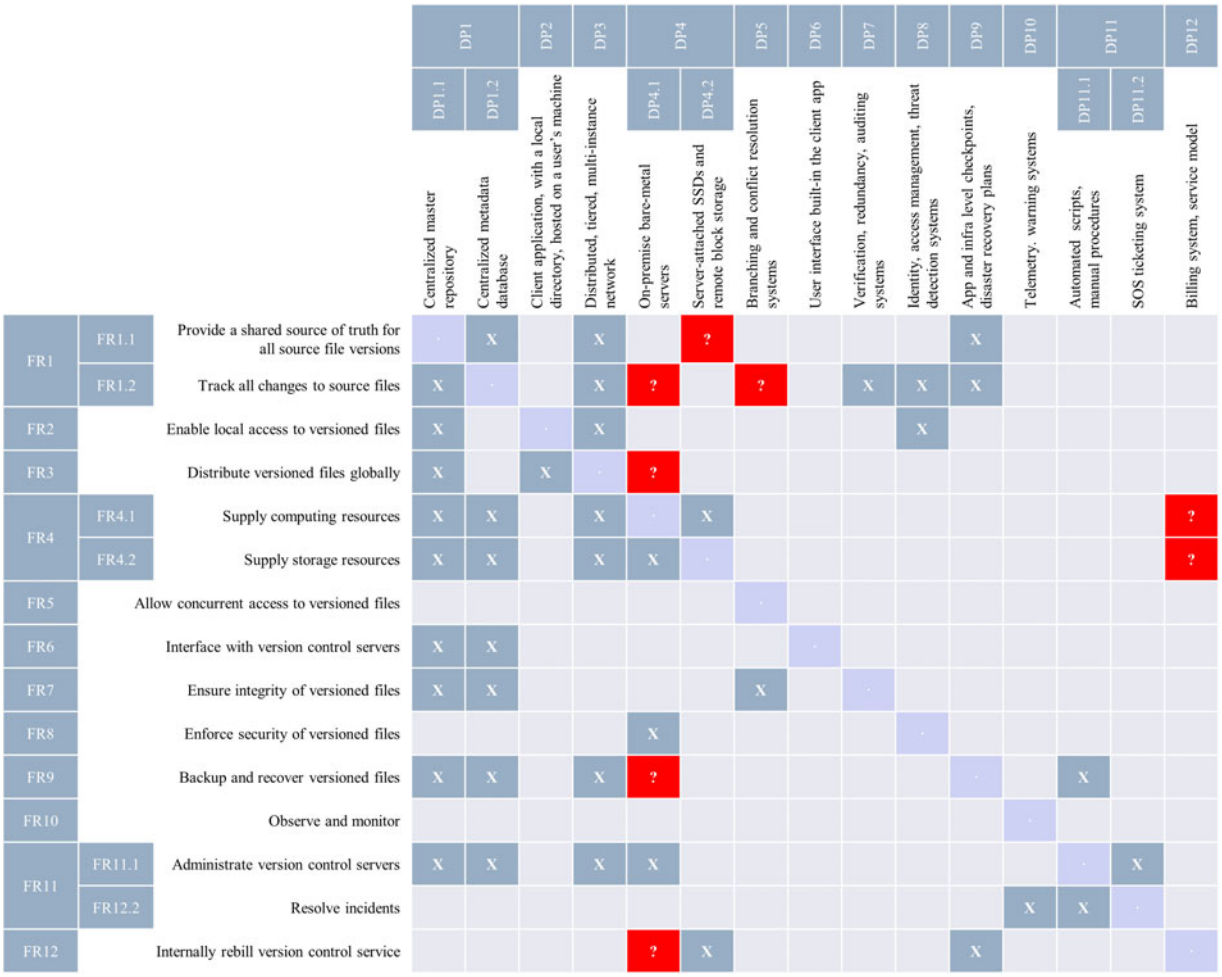


Figure 3. VCS axiomatic design matrix. Blue cells represent known couplings and red cells represent couplings which could not be evaluated by Ubisoft IT teams prior to intervening

These versioning needs are met by using a commercially licensed software around which a complex operationalization has been built. The chosen commercial solution is Perforce's P4, an industry standard for large scale development environments with two essential components, a central database and a master repository of file versions, packaged in a server application: the P4 server. This whole, comprising the commercial solution and the operationalization built by Ubisoft IT, is what we will refer to as the Version Control System (VCS).

In Axiomatic Design terms the VCS **manages versioning of all files involved in all of Ubisoft's game development projects** (FR0) via a **centralized system built around a commercial versioning software** (DP0). To get a better understanding of the different components and subfunctions within the VCS an Axiomatic Design matrix is provided in Figure 3. The matrix's structure reflects the previously described combination of a commercial solution, designed and maintained by an external stakeholder, over which an operationalization developed and maintained by various Ubisoft IT teams is built. FR/DP 1.1 and 1.2 correspond to the core commercial solution and are therefore highly coupled with the rest of the matrix, whereas the other FR/DP pairs exhibit a more decoupled design.

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

This system has been operational since 2004 and has undergone numerous evolutions, most notably in response to changes in how Ubisoft produces games. Over the past 20 years game sizes and the number of employees involved in a single game have increased by a factor of ten. Moreover, co-development across multiple sites has become common practice. Over time, upgrades have been gradually introduced to the VCS to adapt to these radical changes, with relative success. However, in recent years, having been identified by IT architects and management as a TD focal point, addressing the system's shortcomings has become a priority for Ubisoft IT.

4.3. TD management intervention

An initial diagnosis attributed TD in the VCS to architectural complexity, dependence on outdated hardware, and a lack of operational standardization, factors that generate numerous manual fixes and exceptions. IT architecture experts determined that remediation required a new infrastructure architecture enabling operational automation and reducing workload. However, as noted in Section 4.1, the VCS forms part of Ubisoft's broader Technical Heritage, which entails major constraints. Its complexity makes it only partially knowable, and it cannot be shut down or rebuilt from scratch without dismantling Ubisoft's datacenters, a "greenfield" illusion. Any modification risks far-reaching knock-on effects due to unpredictable interactions across system layers. Thus, redesign must regenerate a partially understood system while keeping it operational. To address this, Ubisoft IT teams designed a specific protocol for introducing the new architecture. A single server instance was migrated to an external cloud platform supporting automation, while the rest of the VCS remained intact. This instance, hosting files from a live production game, maintained active links to the broader VCS. The protocol effectively acted as a graft, introducing a donor architecture (virtualized hosting) into the host system (Ubisoft's VCS). A simplified timeline of the process is shown in Figure 4. For confidentiality purposes, the game is referred to as "**Project X**" and the external cloud platform as "**Platform Y**".

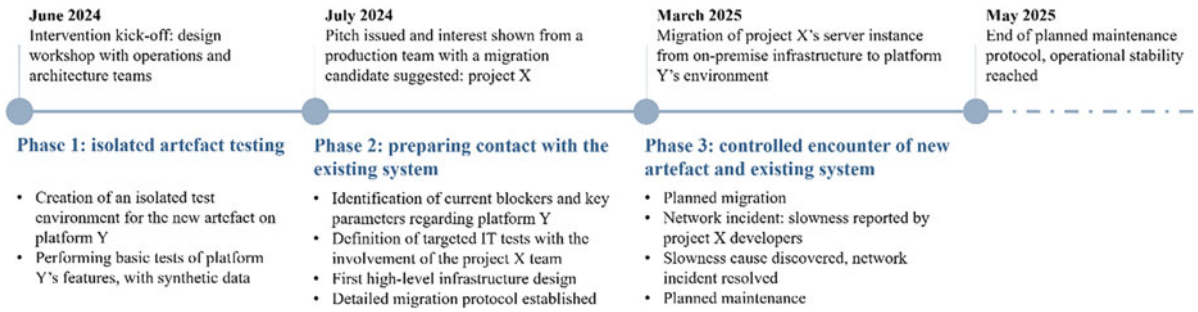


Figure 4. Summary timeline highlighting the main phases and milestones of the VCS's intervention

5. Results: effective experimentation of unknown interfaces

As outlined in Section 2.3, literature on engineering systems interventions distinguishes two stages of propagation experimentation, which are guided by two different logics of action: "scientific proof" and "urgent repair". The following discussion refers to them as Archetype 1 and Archetype 2, respectively.

Figure 5 applies the analytical framework introduced in Section 3 to the VCS intervention. Certain elements, such as the redesign of DP4.2 and DP8 in response to upstream couplings, illustrate a logic of predicting and controlling propagations, characteristic of Archetype 1. However, the VCS intervention moves beyond Archetype 1. It acknowledges, by design, the existence of unknown interfaces, as reducing uncertainty around them had been a key focal point since the project's inception. Moreover, it also recognizes the need to "jump into the deep end" as such interfaces can only be revealed through experimentation, and accepts that despite containment efforts, said experimentation will occur within an uncertain perimeter. This marks a stark departure from established TD management approaches, that advocate for precise identification and measurement of TD before envisioning its eventual repayment. Furthermore, experimentation of unknown interfaces was initiated early in the project cycle, which creates a need to anticipate unexpected events resulting from change propagation and build the capacity to respond to them and learn about them. This is illustrated by the decision to over-engineer DP4.1 and DP4.2, the parallelization of new and legacy versions of DP9 to create an artificial form of reversibility

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

and the redesign of DP12 to funnel risk toward the stakeholder best suited to absorb it. When an incident arose, such as DP3's malfunction caused by an unanticipated replication dependency, teams were able to resolve it swiftly and shed light on the nature of the coupling between FR3 and DP4.1. The process also revealed completely unexpected interfaces. For instance, FR9 performed better than expected following the introduction of the new artefact (DP4.1 and DP11.1). This positive knock-on effect was not expected and enabled an optimization of DP9, previously considered unfeasible.

Level 1: Macro-level intervention		
System of interest	Version Control System	Full FR/DP Matrix
Target process	Operation of P4 instances	FR11.1
Measure of performance	Reduced workload and improved flexibility for P4 operation experts	Disk resize time, deployment time, scale up & down time…
Target system	Infrastructural elements (storage, compute, network) and operations system	The diagnosis reveals that the DPs exerting the greatest influence on FR11.1's performance are DP4 and DP11.1.
Artefact	A virtualized hosting architecture on Platform Y's infrastructure, which enables basic automation of operations through "Infrastructure as Code".	The artefact is a modified version of DP4.1 and DP11.1.
Contingent processes	Observability and monitoring, identity management, billing, replication, backup and Disaster Recovery Plans (DRPs)	Cells along the DP4.1 column and along the DP11.1 column.
Level 2-A: Propagation experimentation of known interfaces		
Target system known interfaces	Identity management, observability and monitoring	According to the matrix, modifying DP4.1 and DP11.1 allows us to determine precisely that: • FR4.1, FR4.2, FR8, and FR11.1 will be affected • FR1.1, FR2, FR5, FR6, FR7, FR8, FR10, and FR11.2 will remain unaffected
Propagation control strategies	Definition of a "sphere of influence" within which the interaction between the new artefact and the existing system is organized. Redesign of known interfaces between the target system and neighboring systems related to contingent processes.	Partial introduction of the new DP4.1 and new DP11.1: • New DPs were deployed in a small part of the overall network, the rest remained unchanged Redesign of existing DPs to decouple the target system from the systems related to contingent processes: • Redesign of DP4.2 • Redesign of DP8
Level 2-B: Propagation experimentation of known interfaces		
Target system unknown interfaces	Billing, replication, backup and DRP of server instances	According to the matrix, modifying DP4.1 and DP11.1 indicates that FR3, FR9, and FR12 will be affected, although the nature of these impacts remains unknown
Propagation response strategies	Tripartite agreement between IT, Financial Planning, and Project X's team, stipulating that cost discrepancies will be covered by IT to avoid penalizing Project X. Limitation of bandwidth consumption through partial redirection of requests to a neighboring server instance. Creation of an artificial reversibility through partial parallelization of infrastructure over a short time frame, ensuring the possibility of rollback. Initial oversizing of server specifications followed by a controlled reduction of server specification to identify the minimal operational configuration.	Redirection of load to existing DPs with known tolerances: • Redesign of DP3 • Redesign of DP12 Parallelization of an old DP and its newer counterpart: • DP9 was redesigned, with the new DP9 coexisting alongside the legacy DP9 for a defined period. Over-engineering and built-in scalability of the new DPs: • DP4.1 and DP4.2 were designed to accommodate high loads, incorporating adjustability features that allow for post-deployment resizing.
Propagation learning strategies	Project X shares a game engine and a version control infrastructure with many other game development projects. A controlled permeability of the "sphere of influence" ensures propagations introduced by the new artefact affect contingent processes.	The characteristics of the network's portion chosen for deploying DP4.1 and DP11.1 ensure that the intervention's results and insights are reproducible and generalizable. Said portion remains connected to the broader VCS environment, allowing changes to propagate within the matrix and reveal interfaces.

Figure 5. Application of the extended analytical framework to the VCS intervention

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

These characteristics reflect a level of codification and preparedness exceeding the "urgent repair" mindset of Archetype 2 and clearly illustrate that the expertise required to manage TD goes well beyond fixing defects and suboptimal designs. Based on this singular case, Figure 6 proposes a preliminary five-step model to begin enriching Archetype 2 for effective experimentation on unknown interfaces. This model is also intended to guide practitioners tasked with navigating complex TD Management projects.

1. Filtering to isolate unknown interfaces: clearing the design space of issues that can be managed through established methods, refining subsequent experimentation processes, leaving only truly unknown interfaces that could not be handled through conventional approaches.
2. Implementing firewalls: establishing containment mechanisms, transient structures, and absorption poles, that aim to mitigate the effects of unknown propagations and build a capability to respond to unexpected interactions without destabilizing the broader system.
3. Conducting the experiment with a crisis management mindset: anticipating potential incidents and organizing rapid response mechanisms by establishing hotlines, pre-positioning response teams to take action as soon as unexpected propagations occur.
4. Ensuring a transition to steady state operation: once initial contact between the artefact and existing system proceeds successfully, attention must shift to effectively managing the transition from crisis response to normal operations and ensuring a proper handover. Experimentation in a production environment requires ensuring that the system continues to operate reliably over an extended period.
5. Leveraging learnings to enable meaningful follow-ups: capitalizing on the insights gained from the unknown propagation experiment to organize future developments.

Figure 6. Five-step model for effective experimentation of unknown interfaces

6. Discussion

Through an analysis of a TD management intervention in Ubisoft's VCS, this paper illustrates how change propagation is managed within an already existing, poorly understood system. The intervention protocol described in the case study reveals previously unknown interfaces early in the design process, triggering targeted redesign activities while retaining the benefits typically associated with experiments on known interfaces notably risk control and learning. By explicitly acknowledging and codifying the existence of such unknown interfaces, and bringing their exploration to the forefront rather than reacting when they manifest as problems, this paper highlights the fact that engineering systems design is a non-linear dialogue between isolated experimentation and experimentation in action.

While elements of the proposed model resemble established practices, such as the progressive reduction of uncertainty in the traditional V-cycle, its originality lies in making this "filtering" effort explicit and acknowledging that a residual "unknown" will subsist said filtering. Moreover, unlike classical models culminating in release and retirement, design activity does not end with the intervention. Instead, deliberate consolidation of lessons learned ensues, assessing whether unknown interfaces have been sufficiently de-risked or whether new ones have emerged. This is the VCS' case, where subsequent improvements, such as server containerization, were pursued within a conventional design framework. The complexity of the research subject motivated a single case-study approach, which enabled a detailed depiction of practical intervention and its corresponding theoretical implications. Nonetheless, ensuring transferability and generalizability of proposed results beyond Ubisoft IT, would require replication across additional systems and organizations facing similar redesign challenges, opening fruitful directions for future research. Furthermore, intervention success depended on both the expertise of the designers involved and organizational support for a systemic renovation of technical heritage, underscoring the organizational dimensions of complex system redesign. This raises important questions regarding how teams are trained, managed, and supported to operate effectively under conditions of partial knowledge, which remain promising directions for future research.

References

Aggeri, F. (2016). *La recherche-intervention: Fondements et pratiques* (p. 79). Dunod. https://minesparis-psl.hal.science/hal-01230457

Avgeriou, P., Kruchten, P., Ozkaya, I., & Seaman, C. (2016). Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). *Dagstuhl Reports*, 6(4), 110–138. https://doi.org/10.4230/DagRep.6.4.110

Bots, P. W. G. (2022). Engineering Systems in Flux: Designing and Evaluating Interventions in Dynamic Systems. In *Handbook of Engineering Systems Design* (pp. 1–27). Springer, Cham. https://doi.org/10.1007/978-3-030-46054-9_19-3

Brahma, A., & Wynn, D. C. (2023). Concepts of change propagation analysis in engineering design. *Research in Engineering Design*, 34(1), 117–151. https://doi.org/10.1007/s00163-022-00395-y

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press

Ciolkowski, M., Lenarduzzi, V., & Martini, A. (2021). 10 Years of Technical Debt Research and Practice: Past, Present, and Future. *IEEE Software*, 38(6), 24–29. IEEE Software. https://doi.org/10.1109/MS.2021.3105625
Clarkson, J. P., Simons, C., & Eckert, C. (2004). Predicting change propagation in complex design. *Journal of Mechanical Design*, 126(5), 788–797. https://doi.org/10.1115/1.1765117
D'Amelio, V., Chmarra, M. K., & Tomiyama, T. (2011). Early design interference detection based on qualitative physics. *Research in Engineering Design*, 22(4), 223–243. https://doi.org/10.1007/s00163-011-0108-7
de Weck, O. L., Roos, D., Magee, C. L., & Cooper, S. (2011). *Engineering Systems: Meeting Human Needs in a Complex Technological World*. The MIT Press. https://doi.org/10.7551/mitpress/8799.001.0001
Eckert, C., & Clarkson, J. (2023). The Evolution of Complex Engineering Systems. In Handbook of Engineering Systems *Design* (pp. 1–39). Springer, Cham. https://doi.org/10.1007/978-3-030-46054-9_6-2
Eppinger, S. D., & Browning, T. R. (2012). *Design Structure Matrix Methods and Applications*. MIT Press.
Ernst, N. A., Bellomo, S., Ozkaya, I., Nord, R. L., & Gorton, I. (2015). Measure it? Manage it? Ignore it? software practitioners and technical debt. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015*, 50–60. https://doi.org/10.1145/2786805.2786848
Forsberg, K., & Mooz, H. (1992). The Relationship of Systems Engineering to the Project Cycle. *Engineering Management Journal*, 4(3), 36–43. https://doi.org/10.1080/10429247.1992.11414684
Jobin, C. (2022). La preuve de concept comme outil de développement des capacités de générativité collective: Modélisation, expérimentation et conditions de performance [Phdthesis, Université Paris sciences et lettres]. https://pastel.hal.science/tel-03738429
Koppenjan, J., Veeneman, W., van der Voort, H., ten Heuvelhof, E., & Leijten, M. (2011). Competing management approaches in large engineering projects: The Dutch RandstadRail project. *International Journal of Project Management*, 29(6), 740–750. https://doi.org/10.1016/j.ijproman.2010.07.003
Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101, 193–220. https://doi.org/10.1016/j.jss.2014.12.027
Magnusson, J., & Bygstad, B. (2014, June 7). Technology debt: Toward a new theory of technology heritage. Proceedings of the European Conference on Information Systems (ECIS). https://aisel.aisnet.org/ecis2014/proceedings/track15/2
Maier, A., Oehmen, J., & Vermaas, P. E. (2022). *Introducing Engineering Systems Design: A New Engineering Perspective on the Challenges of Our Times. In Handbook of Engineering Systems Design*. Springer Nature. https://link.springer.com/rwe/10.1007/978-3-030-46054-9_1-3
McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, 18(2), 67–92. https://doi.org/10.1080/08993400802114581
McConnell, S. (2013). Managing Technical Debt. The Proceedings of ICSE 2013. International Conference on Software Engineering 2013.
Mercat, F., Masson, P. L., & Weil, B. (2025). The unique feature of designing with digital twin uncovered by design theory. *Proceedings of the Design Society*, 5, 3101–3110. https://doi.org/10.1017/pds.2025.10324
Olechowski, A. L., Eppinger, S. D., Joglekar, N., & Tomaschek, K. (2020). Technology readiness levels: Shortcomings and improvement opportunities. *Systems Engineering*, 23(4), 395–408. https://doi.org/10.1002/sys.21533
Parslov, J. F., & Mortensen, N. H. (2015). Interface definitions in literature: A reality check. *Concurrent Engineering*, 23(3), 183–198. https://doi.org/10.1177/1063293X15580136
Suh, N. P. (2001). *Axiomatic design: Advances and applications* (Vol. 4). Oxford University Press. https://www.axiomaticdesign.com/technology/axiomatic-design-advances-and-applications/
Terwiesch, C., & E. Bohn, R. (2001). Learning and process improvement during production ramp-up. *International Journal of Production Economics*, 70(1), 1–19. https://doi.org/10.1016/S0925-5273(00)00045-1

https://doi.org/10.1017/pds.2026.10632 Published online by Cambridge University Press