LUND UNIVERSITY
School of Economics and Management
Department of Informatics

Sociotechnical Factors of Post-Adoption Software Upgrade Instability in Enterprise IT Ecosystems: A mixed-method study

Master thesis 15 HEC, course INFM12, Master Thesis in Information Systems

Authors: Gunnþóra Mist Björnsdóttir
Sharon Mpofu

Supervisor: Nam Aghaee

Grading Teachers: Blerim Emruli Daniel Lindvall

Sociotechnical Factors of Post-Adoption Software Upgrade Instability in Enterprise IT Ecosystems: A mixed-method study

AUTHORS: Gunnþóra Mist Björnsdóttir and Sharon Mpofu

PUBLISHER: Department of Informatics, Lund School of Economics and Management, Lund University

PRESENTED: May, 2026

DOCUMENT TYPE: Master Thesis

FORMAL EXAMINER: Osama Mansour, Associate professor

NUMBER OF PAGES: 81

KEY WORDS: Software upgrade instability, Post-adoption, Enterprise ecosystems, Sociotechnical systems

ABSTRACT (MAX. 200 WORDS):
Software upgrades are a substantial source of instability in enterprise IT ecosystems. Instabilities are most common during the post-adoption phase, when systems must remain operational while evolving. This thesis examines post-adoption software upgrade instability from a sociotechnical perspective. A sequential explanatory mixed-methods design was adopted. The quantitative analysis identified recurring patterns in issue categories, operational impact, escalation behaviour, and resolution characteristics. Qualitative findings explain how support practices, coordination breakdowns across roles, governance constraints, and decision-making under uncertainty shape these patterns. The study finds that upgrade instability is not a temporary post-adoption irregularity. Instead, it is a sustained operational condition reinforced over time through sociotechnical mechanisms. Configuration and compatibility issues formed the largest category of upgrade-related incidents. Support practices shaped how instability became visible, detected and understood. Coordination and unclear ownership limit the resolution authority, contributing to degraded system states. Decision-making under uncertainty is shown to be structurally embedded in enterprise IT ecosystems. The study contributes a process-based understanding of upgrade instability, shifting focus from discrete failures to ongoing post-adoption dynamics. The study is based on a single enterprise IT ecosystem, and with that in mind, the findings support analytical generalisation to similar contexts.

Content

Figures

Tables

1 Introduction

This chapter provides an overview of the study by introducing the context of enterprise IT ecosystems and post-adoption support for software upgrades. It establishes the research problem by identifying the gap in managing software upgrade instability, outlines the study's motivation, and presents the research question. Finally, the thesis's specific delimitations define the research's focus and scope.

1.1 Background

Enterprise IT environments now operate as interconnected ecosystems rather than isolated systems (Costa et al. 2024; Tiwana et al. 2010). Therefore, software vendors are required to integrate heterogeneous components, such as third-party hardware, standards, configurations, and diverse user roles. Components that each of which affects operational outcomes (Jakobsen & Vanhaverbeke, 2025; Kapoor et al. 2021).

A persistent challenge within these ecosystems is the upgrade process. The process entails updating core software, connected components, or configurations to implement security patches, introduce new features, or improve compatibility (Petersen & Seymour, 2021; Zhang et al. 2021). Although upgrades are necessary, they also introduce operational risks (Feldman et al. 2017; Zhang et al. 2021). In large-scale enterprise environments, inadequate coordination of upgrades can lead to cascading operational disruptions, increased support demands, diminished organisational credibility, and financial losses (Dumitraş & Narasimhan, 2009; Kotlarsky et al. 2019; Retana et al. 2018; Zhang et al. 2021).

Currently, stability within these platform ecosystems depends not only on the underlying code but also on intricate coordination, the allocation of responsibility, and decision-making across organisations and roles (Collopy et al. 2020; Jacobides et al. 2024). These aspects illustrate the broader sociotechnical conditions that outline system outcomes in enterprise environments (Techakriengkrai et al. 2021).

In this thesis, upgrade instability is defined as unintended operational disruptions, failures, or performance degradation (Dumitraş & Narasimhan, 2009). Where increasing support incidents are emerging from system version changes within an enterprise ecosystem that compromise operational continuity and reduce stakeholder confidence (Feldman et al. 2017; Vert & Sharpanskykh, 2026). These disruptions are fundamentally sociotechnical. Notably, the technical elements, such as version dependencies, compatibility mismatches, and integration-level faults, introduce structural complexity and potential failure points. In contrast, the organisational elements, such as governance bottlenecks, misaligned responsibilities, and inefficient communication between teams, shape how these technical conditions are interpreted and managed in practice (Dumitraş & Narasimhan, 2009; Zhang et al. 2021).

Prior research addresses many of the technical and organisational aspects of platform ecosystems and product configuration systems (Rydén Sonesson et al. 2021). Retrospectively, existing IS studies primarily focus on system adoption, user acceptance, or the technical and organisational causes of upgrade failures. Thereby, explaining how system defects,

dependencies, and contingencies influence upgrade outcomes (Khoo & Robey, 2007; Techakriengkrai et al. 2021; Zhang et al. 2021).

Prior studies have progressively recognised the significance of the post-adoption phase in enterprise systems (Techakriengkrai et al. 2021). To clarify, post-adoption research demonstrates that system value and long-term outcomes depend on sustained use and effectiveness of support structures following deployment (Jasperson et al. 2005; Retana et al. 2018). However, most existing studies have concentrated on initial adoption and implementation rather than the ongoing, dynamic changes. Furthermore, prior research does not expand on how responsibility for cross-cutting upgrade risks is distributed, negotiated, and operationalised among actors in the ecosystem context (Rydén Sonesson et al. 2021). While existing literature acknowledges the importance of support conditions, it offers insufficient explanation of how these conditions are structured and enacted in practice during upgrades (Feldman et al. 2017; Kapoor et al. 2021).

For example, Techakriengkrai et al. (2021) identify socio-technical disruptions resulting from new system implementations. Moves on to explicitly recommend that future research apply this socio-technical perspective to other Enterprise Systems (ES) contexts within the post-adoption stage. As a result, a substantial gap persists in understanding how socio-technical mechanisms. Particularly, coordination practices and decision-making under uncertainty influence operating stability during recurring software upgrades in complex IT ecosystems (Dissanayake et al. 2022; Dumitraş & Narasimhan, 2009; Figalist et al. 2019; Jasperson et al. 2005).

Taken together, the lack of detailed critique suggests that existing literature underestimates the complexity of sociotechnical influences on upgrade stability. Accordingly, this thesis explicitly addresses this gap by focusing on coordination and decision-making as central sociotechnical factors through which upgrade instability is manifested and managed in practice, while also recognising compatibility mismatches and detailed technical conditions, what broke, aligning with underlying drivers that interact with these processes. This motivates the need for a structured analysis of these factors underlying upgrade instability, which is further elaborated in the following section (Techakriengkrai et al. 2021).

1.2 Problem Area

Enterprise ecosystems often experience upgrade instability due to variations in interoperability across heterogeneous hardware and software configurations. Also, fragmented support knowledge across actors and limited transparency into upgrade risk and prioritisation decisions make it difficult to diagnose and coordinate responses to upgrade-related incidents (Khoo & Robey, 2007; Tamanna et al. 2025). Research on software upgrades and operational dependability further supports that failures may result from these technical and process-related conditions, complicating upgrade execution and reducing reliability (Khoo & Robey, 2007; Tamanna et al. 2025).

Upgrade decisions in enterprise software contexts are also informed by organisational objectives (see Figure 1). Objectives such as contingencies and interdependencies that extend

beyond the technical aspects of the release (Feldman et al. 2017; Petersen & Seymour, 2021).

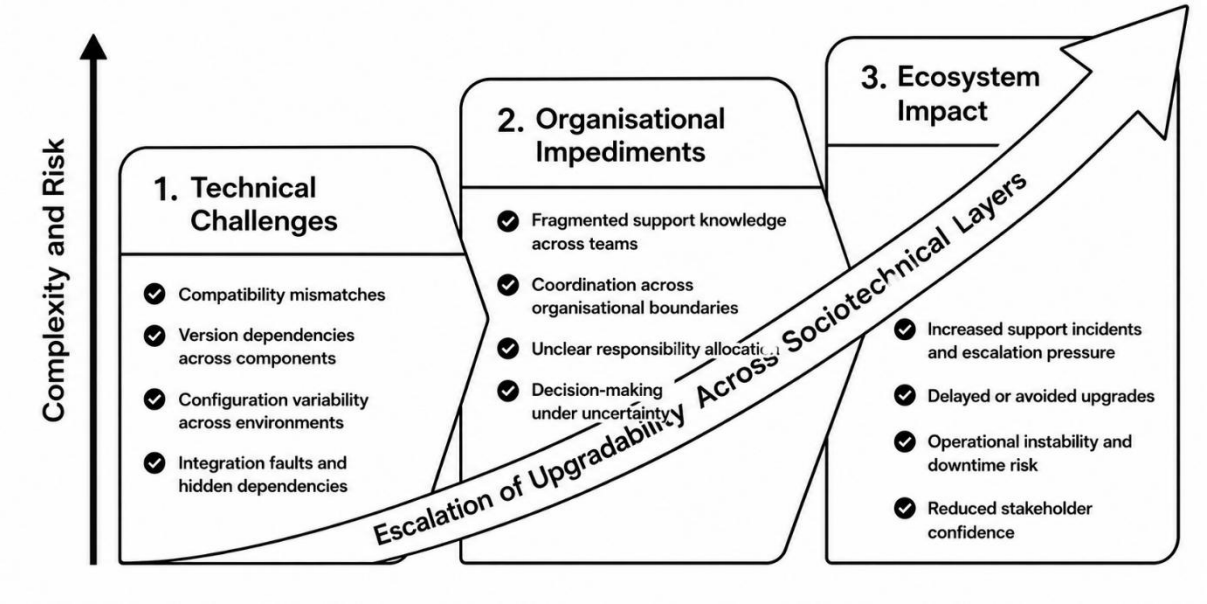


Figure 1: Connection between the three stages of upgrading environment dependencies

Ecosystems are becoming increasingly interdependent, and organisations must therefore manage upgrade instability effectively. Thus, doing so across organisational boundaries to preserve system value and ensure operational continuity (Dumitraş & Narasimhan, 2009; Kotlarsky et al. 2019; Retana et al. 2018; Zhang et al. 2021). Figure 1 reflects these boundaries and complications within them. Although prior research identifies individual causes of upgrade failure, it does not sufficiently clarify how these conditions combine and interact to produce instability in real-world ecosystem settings (Dumitraş & Narasimhan, 2009; Feldman et al. 2017; Jacobides et al. 2024; Khoo & Robey, 2007; Zhang et al. 2021). The following section, therefore, clarifies the specific research gap and emphasises the need for this study.

1.3 Research Gap

This study specifically addresses the underexplored issue of upgrade instability in enterprise IT ecosystems during the post-adoption maintenance phase. Prior research has examined post-adoption system use and continuance (Bhattacherjee, 2001; Jasperson et al. 2005; Techakriengkrai et al. 2021) as well as technical aspects of software upgrades and related failures (Dumitraş & Narasimhan, 2009; Zhang et al. 2021). In parallel, studies on enterprise ecosystems highlight challenges related to coordination, distributed roles, and interdependencies across organisational boundaries (Collopy et al. 2020; Jacobides et al. 2018; Kotlarsky et al. 2019). However, existing research has not sufficiently examined how post-upgrade incidents, support mechanisms, and cross-organisational coordination interact to shape upgrade instability in enterprise IT ecosystems. By focusing on this intersection, the study clarifies a gap in the literature concerning the interplay between operational incidents, support processes, and coordination dynamics in the post-adoption upgrade context.

In software ecosystems, the management of upgrade instability and the support outcomes it produces depends on the quality of guidance, clarity in responsibility allocation, and prioritisation of high-impact issues (Kotlarsky et al. 2019). Consequently, there is a need to clarify

how sociotechnical functions enable actors to coordinate, prioritise high-impact drivers of instability, and ensure that upgrade decisions are transparent and consistent across roles (Jacobides et al. 2024; Techakriengkrai et al. 2021). Based on the identified gap, the next section defines the study's aim and objectives.

1.4 Research Aim and Objectives

This study aims to identify and explain the sociotechnical factors that contribute to instability in software upgrades in enterprise IT ecosystems. The study adopts a sociotechnical perspective. In detail, it aims to examine issues across system dependencies and configuration variability. Additionally, the study views how knowledge distribution, coordination challenges, decision-making under uncertainty, and limitations in system observability affect the issues (Collopy et al. 2020).

This approach builds upon existing Information Systems research on post-adoption system use and maintenance. In addition, drawing on other research on sociotechnical, coordination, and exosystem research. Importantly, the post-adoption use and maintenance phase is critical for ensuring long-term system success (Bhattacherjee, 2001; Petersen & Seymour, 2021). Consequently, the research aim is directly informed by the literature, which emphasises the importance of sustained operational effectiveness, signalling a shift from adoption-centric analyses to an integrated exploration of factors affecting stability throughout the post-adoption lifecycle (Techakriengkrai et al. 2021).

To achieve this aim, the study aims to pursue the following objectives. First, identify key sociotechnical factors contributing to upgrade instability post-adoption. Secondly, analyse how these factors interact in a real-world upgrade scenario with a mixed-method study. Finally, explain how these interactions influence operational outcomes and suggest guidelines to mitigate the problem. To operationalise the research aim, the following research question is formulated.

1.5 Research Question

What sociotechnical factors contribute to instability in software upgrades in enterprise IT ecosystems?

This study is conducted in collaboration with an enterprise IT company operating a large-scale enterprise ecosystem. The organisation provides the empirical context for the study by supplying anonymised incident data and access to technical subject-matter experts involved in the upgrade process, while ensuring confidentiality and professional safeguards. This collaboration allows the research to ground the study in real operational challenges while maintaining generalisability through analytical standards. The findings may yield transferable theoretical insights into sociotechnical factors that apply to similar enterprise ecosystems. To clarify the study's boundaries and feasibility, the following section outlines its scope and key delimitations.

1.6 Scope and Delimitations

This study is subject to several scope and methodological delimitations. First, the study focuses on analysing upgrade instability in enterprise IT ecosystems and aims to identify the sociotechnical factors that contribute to instability following version changes. Second, the study focuses on the post-adoption phase of system use, specifically upgrade situations. Other lifecycle phases, such as initial system adoption, system design, pre-deployment testing, and the full software development process, are outside the scope of this research. Third, the study follows a mixed-method approach, combining quantitative analysis of support incident data with qualitative insights from semi-structured dyadic interviews. While this approach enables a richer understanding of the phenomenon, it relies on aggregated and anonymised data, which may limit the level of technical detail that can be disclosed. Finally, the study adopts a case-based research approach within a single enterprise ecosystem. Accordingly, the findings aim for analytical generalisation rather than statistical generalisation.

Having established the scope and delimitations of the study, the remainder of this master thesis is organised as follows. Section 2 presents a comprehensive review of the relevant literature to contextualise the research problem. Section 3 details the methodology, describing the data collection procedures and data analysis methods employed in the study. Section 4 offers the problem identification and reports the empirical findings derived from the case analysis. Section 5 provides an in-depth discussion of these findings in relation to the research question and existing literature. Finally, the Conclusion summarises the thesis's primary contributions and offers recommendations for future research.

2 Literature Review

This chapter develops the theoretical foundation for the study by examining enterprise IT ecosystems, upgrade instability, and the sociotechnical mechanisms that model post-adoption outcomes. It further reviews how support practices create patterns of coordination that shape decision-making, thereby determining how technical issues manifest as upgrade instability in enterprise ecosystems.

2.1 Enterprise IT Ecosystems as Sociotechnical Environments

2.1.1 Structural Properties of Enterprise IT Ecosystems

Enterprise IT environments are increasingly understood as ecosystems as opposed to siloed systems. Hence, reflecting the interconnected and rapid evolution of modern digital infrastructures (Costa et al. 2024; Hanseth & Lyytinen, 2010; Tiwana et al. 2010). These ecosystems are typically organised around layered modular architectures, where hardware, networks, services, and applications are loosely coupled but remain functionally interdependent (Yoo et al. 2010). Correspondingly, enterprise IT ecosystems operate as dynamic configurations in which system behaviour emerges from the interaction of multiple interconnected components (Jacobides et al. 2018).

Modular architecture is often associated with flexibility and scalability by enabling upgrades without redesigning the entire system. However, prior research suggests that modularity does not eliminate complexity but instead redistributes it across layers and actors. Therefore, the complexity results in the architecture creating more interdependencies that are difficult to anticipate before updates are fully implemented (Henfridsson & Bygstad, 2013; Kapoor et al. 2021; Yoo et al. 2010). The architecture further creates situations where issues often surface in the sections between models within the modular architecture (Collopy et al. 2020). Notwithstanding the modular architecture, upgrade-related instability is more prevalent in large enterprise IT environments due to accumulated complexity and organisational scale (Collopy et al. 2020; Haraldsson & Staron, 2025). These challenges become more pronounced in heterogeneous enterprise ecosystems, where variations in customer configuration and external dependencies introduce contextual differences that complicate upgrade outcomes (Anthony, 2016; Knoche & Hasselbring, 2021).

Heterogeneity is characterised by the coexistence of customised configurations and evolving digital services (Hanseth & Lyytinen, 2010; Jakobsen & Vanhaverbeke, 2025; Knoche & Hasselbring, 2021). This heterogeneity increases system complexity by introducing variability in how components interact, often leading to non-linear and context-dependent outcomes. Drawing on perspectives from digital infrastructure and complexity theory, enterprise IT environments can be understood as complex adaptive systems, where small changes may trigger disproportionate and unpredictable effects across the system (Benbya & McKelvey, 2006;

Drechsler et al. 2022; Vert & Sharpanskykh, 2026; Yoo et al. 2012). In practice, this means that standardised upgrades are rarely implemented uniformly, as local configurations and historical dependencies shape how systems respond to interventions. Sunyaev et al. (2023) similarly argue that enterprise information systems are increasingly shaped by the need to manage heterogeneity while still maintaining interoperability. This reinforces the view that enterprise IT ecosystems cannot be understood as isolated technical systems, since stability depends on how technical components, data flows, configurations, and organisational routines interact.

2.1.2 Distributed Control and Coordination Challenges

As interdependencies and heterogeneity increase, so do the coordination challenges within the ecosystem. Coordination theory suggests that interdependent activities require mechanisms such as communication, shared understanding, and structured processes to ensure alignment (Collopy et al. 2020; Kapoor et al. 2021; Malone & Crowston, 1994). Furthermore, as heterogeneity rises, variability and unpredictability in how systems respond to upgrades mean no individual actor can retain full visibility or control over system behaviour. Thus, making system operation and maintenance inherently dependent on the coordinated, structured involvement of multiple organisational actors (Ghazawneh & Henfridsson, 2013; Tiwana et al. 2010). Thereby, the post-adoption phase becomes particularly vulnerable to instability (Khoo et al. 2011).

In enterprise IT ecosystems, requirements are intensified by the organisation's scale and distribution, making coordination more complex and less predictable. Empirical studies show that breakdowns in coordination, such as unclear responsibilities, misaligned expectations, or fragmented knowledge, are a frequent source of operational disruptions (Dissanayake et al. 2019; Kotlarsky et al. 2019). The following section conceptualises software upgrade instability as an outcome that emerges within this environment.

2.2 Post-Adoption Software Upgrade Instability

2.2.1 Post-Adoption and Importance of Upgrades

The post-adoption phase is the period after a system has been implemented and accepted. In Information Systems research, post-adoption is widely recognised as the stage in which the long-term value of a system is realised through sustained use, adaptation, and ongoing support (Bhattacherjee, 2001; Jasperson et al. 2005). The complexity of upholding use, adoption and ongoing support is important. Thus, the risks associated with system upgrades become particularly pronounced during the post-adoption phase, where systems must remain operational while evolving (Khoo et al. 2011). Furthermore, the primary risk is that upgrades can disrupt established configurations and operational routines. Thereby, the users' fear of the risk creates a fundamental tension for users to upgrade (Feldman et al. 2017; Khoo et al. 2011). However, upgrades are intended to improve functionality, security, and compatibility. That said, this study addresses the need for upgrades in four areas: structural, ecosystem, technical evolution, and value improvement.

Enterprise IT companies need to ensure that upgrades are carried out to maintain security, comply with industry standards, and align with vendor lifecycles. To maintain operational status, companies need to upgrade to keep up with versions and ensure system support, preventing incompatibilities from accumulating (Feldman et al. 2017; Petersen & Seymour, 2021).

Competitive advantage is ensured by continued system success, which depends on adaptation, maintenance, and ongoing upgrades. Ensuring ecosystems function and aligning with external services, third-party components, and evolving standards (Bhattacherjee, 2001; Petersen & Seymour, 2021; Tiwana et al. 2010). However, the need for continuous upgrading also means that enterprise systems are repeatedly exposed to change. Consequently, creating conditions in which unintended upgrade instability may emerge (Dumitraş & Narasimhan, 2009; Zhang et al. 2021).

2.2.2 Upgrade Instability as an Operational Outcome

In this context, upgrade instability refers to unintended disruptions, failures, or performance degradation that occur after system changes and compromise operational continuity. Drawing on the Information Systems Success model, such instability reflects a decline in system quality and service quality, manifested through errors, reduced performance, or increased support demands (DeLone & McLean, 2003). Importantly, instability does not involve only complete system breakdowns but also partial degradations that accumulate over time. Other examples of instability include slower response times, increased incident frequency, and reduced reliability. Understanding why such instability occurs requires examining the conditions under which upgrades are implemented (Dumitraş & Narasimhan, 2009; Jabin et al. 2024; Zhang et al. 2021).

From a technical perspective, upgrade instability is closely linked to interoperability challenges and configuration dependencies within complex system environments. Enterprise systems rely on interconnected components, where software upgrades must interact with existing hardware and external services. As prior research shows, such environments are particularly vulnerable to integration risks, as changes in one component may produce unintended cascading effects across others (Dumitraş & Narasimhan, 2009). These dependencies make it difficult to anticipate the consequences of upgrades before full deployment (Sommerville et al. 2012). Therefore, creating the dilemma of technical risks being structurally invisible before deployment, because the system's behaviour depends on local configuration and interactions (Desouza et al. 2025). Resulting in upgrade instability, which often becomes apparent only after the upgrade (Sommerville et al. 2012).

Moreover, local customisations and configuration drift further complicate this process, as standardised vendor upgrades interact with context-specific system settings in unpredictable ways (Strong & Volkoff, 2010). Prior research shows that post-adoption upgrades can place systems in degraded states that are difficult to reverse once deployment has occurred (Zhang et al. 2021). Troubleshooting actions intended to restore functionality may further alter system configurations, increasing instability instead of resolving it. Resulting in further damage and downtime while support teams attempt incremental fixes without the option of rollbacks (Adel et al. 2025; Jabin et al. 2024).

Support practices play a central role in this process in identifying, diagnosing, and responding to these upgrade instabilities. The organisational response depends on support practises to

translate the issues into an actionable response (Adel et al. 2025; Kotlarsky et al. 2019). Looking into how support practices act as a foundational sociotechnical layer through which instability becomes visible and manageable (Techakriengkrai et al. 2021).

2.3 Support Practices as the Foundational Sociotechnical Layer

2.3.1 Support Practices as Sociotechnical Infrastructure

To explain how upgrade instability emerges in practice, it is necessary to examine the sociotechnical dynamics that form coordination, knowledge sharing, and support processes across ecosystem actors. Sociotechnical systems theory holds that the interaction between social and technical subsystems moulds organisational outcomes (Mumford, 2006; Sarker et al. 2019). A core principle of this perspective is joint optimisation, which suggests that changes to technical systems cannot be effective unless organisational routines, responsibilities, and forms of coordination are adjusted accordingly (Jakobsen & Vanhaverbeke, 2025; Kapoor et al. 2021). A common side effect is that technical changes are introduced faster than the surrounding organisational arrangements can absorb and coordinate them (Jakobsen & Vanhaverbeke, 2025; Kapoor et al. 2021).

In practice, support systems serve as a central mechanism for identifying and managing instability, as incidents are reported, escalated, and resolved across organisational boundaries (Costa et al. 2024). Post-adoption support practices in enterprise IT environments primarily aim to maintain system reliability and address the sociotechnical complexities of user requests, system incidents, and ongoing software upgrades (Techakriengkrai et al. 2021). These practices are typically supported by specialised IT support functions, help desks, and Change Control Boards, which coordinate activities through formal routines such as prioritising service requests, adhering to established troubleshooting procedures, and utilising clearly defined roles and networks (Kotlarsky et al. 2019; Wenrich & Ahmad, 2009).

2.3.2 Reactive vs Proactive Support

Post-adoption research highlights that system success depends not only on implementation but on the effectiveness of ongoing support structures that enable users to resolve issues and maintain system functionality (Jasperson et al. 2005; Retana et al. 2018). Nevertheless, despite the existence of these formal structures, many organisations continue to rely on ad-hoc, reactive troubleshooting and documentation-centric practices. This reliance restricts their capacity to identify recurring instability patterns proactively and to maintain stable operations throughout the system's life cycle (Jasperson et al. 2005; Kotlarsky et al. 2019; Retana et al. 2018).

2.3.3 Visibility and Escalation

Documentation often provides the first impression of a product's quality and is usually the initial resource for understanding new upgrades (Hou & Jansen, 2023). The downside is that documentation is produced by the product’s creators, which can be perceived as subjective, unclear, or insufficient for objectively evaluating the upgrade's value and impact. Thus,

organisations frequently supplement these documents with external consultants or internal "gap-fitting" resolution (Feldman et al. 2017).

During complex incident response, relying on extensive documentation and formal reviews is often less effective than utilising informal channels to consult relevant experts (Collopy et al. 2020). Uludağ et al. (2019) argue that systems that are developed through incremental changes without clear structural guidance risk maintenance issues further down the line. Over time, this can accumulate into a significant architectural gap, necessitating a comprehensive redesign of the system. All because the documentation lagged behind ongoing system changes (Feldman et al. 2017; Sommerville et al. 2012)

Escalation paths are critical components of issue response that dictate how urgent issues are communicated, managed, and resolved (Costa et al. 2024). In large-scale enterprise IT environments, where processes, structures, and hierarchies are more common, formal escalation mechanisms emerge as sociotechnical responses to coordination bottlenecks. The consequence is that support actors sometimes amplify issue severity to bypass slow procedures and ensure timely intervention during critical system events (Kotlarsky et al. 2019).

2.3.4 Information Conditioning

In software ecosystems, operational knowledge is distributed among multiple actors, so no single individual possesses a complete understanding of system behaviour (Hou & Jansen, 2023). To address this fragmentation, organisations use a centralised knowledge hub and incident repositories to document past resolutions, which significantly reduces the time required to diagnose and troubleshoot future upgrade issues (Costa et al. 2024). When formal training or documentation is insufficient, organisations rely on informal knowledge sharing, such as word of mouth, experience, or connections to the software team, to facilitate learning during new upgrades (Wenrich & Ahmad, 2009). However, during emergencies, dependence on informal networks and personal connections can restrict predictability and shared understanding to a small group, potentially impeding broader resolution efforts (Kotlarsky et al. 2019).

Support practices determine what information becomes available for coordination and decision-making by shaping visibility, accessibility and flow of operational knowledge across technical and organisational boundaries (Figalist et al. 2019; Kotlarsky et al. 2019). They act as both conduits and filters, influencing which issues are recognised, how they are interpreted, and when they are prioritised (Kotlarsky et al. 2019). Because many upgrade effects only emerge in operational environments, support systems serve as the primary sites where technical disruptions are first detected and translated into actionable knowledge (Dumitraş & Narasimhan, 2009).

The structure and maturity of support practices affect whether information remains fragmented across individual cases or becomes formalised through incident records, documentation, and shared reference points (Costa et al. 2024). At the same time, escalation routines and severity classification filter which issues receive attention and how, affecting how quickly relevant information reaches decision-makers (Adel et al. 2025; Costa et al. 2024; Kotlarsky et al. 2019).

As prior research has shown, when support documentation and known fixes mature over time, issues become easier to recognise and handle locally (Costa et al. 2024). In contrast, during

early or ambiguous upgrade phases, limited documentation and partial visibility constrain actors' understanding of system-wide implications (Figalist et al. 2019).

Prior research largely treats upgrade instability as a static outcome, offering less insight into how instability evolves as organisations interpret incidents, refine documentation, and develop shared handling practices following deployment. Hence, support practices do not merely respond to technical instability but actively condition the informational landscape within which subsequent coordination across roles must occur (Adel et al. 2025; Costa et al. 2024; Khoo & Robey, 2007; Kotlarsky et al. 2019).

2.4 Coordination Across Roles in Upgrade Contexts

2.4.1 Cross-Boundary Coordination and Role Asymmetry

Because many upgrade-related issues only become visible once systems are operating in real environments, post-adoption support structures become the primary mechanisms through which instability is detected, interpreted, and managed in practice. In enterprise ecosystems, responsibilities for upgrades are distributed across multiple actors, including vendors, integrators, and internal IT teams, each operating with different priorities and knowledge bases (Ghazawneh & Henfridsson, 2013). This fragmentation introduces coordination challenges, as decisions regarding testing, deployment, and risk assessment are often made within partial organisational contexts. Hence, upgrades may be implemented without a comprehensive understanding of system-wide dependencies, increasing the likelihood of unforeseen issues in live environments (Figalist et al. 2019; Khoo & Robey, 2007).

In the context of enterprise IT environments, a coordination challenge regarding upgrade instabilities is a complex sociotechnical problem in which interdependent actors struggle to align their activities, share knowledge, and negotiate responsibilities across organisational boundaries during a system version change (Collopy et al. 2020; Hou & Jansen, 2023; Techakriengkrai et al. 2021). In platform ecosystems, coordination complexity increases with the number of actors and dependencies, making misalignment a key source of operational disruption (Jacobides et al. 2024; Kapoor et al. 2021). Prior research further shows that breakdowns in coordination lead directly to system failures and inefficiencies in distributed IT environments (Kotlarsky et al. 2019)

Beyond this structural fragmentation, coordination during upgrades is further complicated by the uneven distribution of system insights and decision authority across roles. Actors involved in upgrade processes often possess expertise that is narrowly tied to specific components, services, or customer environments, while lacking awareness of how their actions affect the broader ecosystem (Jacobides et al. 2024; Jakobsen & Vanhaverbeke, 2025). This asymmetry constrains collective sense-making during upgrades, as coordination depends on reconciling partial and sometimes conflicting interpretations of system state and risk (Collopy et al. 2020; Ren et al. 2008; Saxena & McDonagh, 2022).

2.4.2 Misalignment and Reactive Coordination Dynamics

Coordination is escalated by the number of actors involved, limited knowledge of the upgrade system failure, and risk prioritisation (Collopy et al. 2020; Saxena & McDonagh, 2022). When responsibilities for diagnosing, escalating, or mitigating upgrade issues are dispersed, individuals rely on local assessments to guide their actions. Consequently, delaying the response period across organisational boundaries (Jacobides et al. 2024; Ren et al. 2008). Accordingly, coordination becomes reactive, with alignment occurring only after instability has materialised (Kotlarsky et al. 2019; Rydén Sonesson et al. 2021). Prior studies indicate that under reactive situations, coordination is achieved through iterative escalation and negotiation. Resulting in predefined processes, which increase uncertainty and prolong resolution efforts (Collopy et al. 2020; Kotlarsky et al. 2019).

The literature shows that distributed governance structures characterise enterprise IT ecosystems. The control and responsibilities are fragmented across organisational boundaries (Jacobides et al. 2018). Prior research shows that this misalignment can create governance bottlenecks, as no single actor possesses both the authority and information required to make decisions. This is often an outcome of decision paralysis and delayed resolution (Rydén Sonesson et al. 2021).

2.5 Decision-Making Under Uncertainty in Post-Upgrade Situations

2.5.1 Decision-Making Under Incomplete Information

Decision-making in IT environments is constrained by the circumstances of actors making choices under conditions of incomplete information and uncertainty (Collopy et al. 2020). In upgrade contexts, limited visibility into dependencies, risks, and system states generates suboptimal decisions that increase the likelihood of instability (Dumitraş & Narasimhan, 2009; Petersen & Seymour, 2021).

Decision-making during post-adoption instability is reactive to the operational context in which systems are deployed. Prior studies note that systems supporting critical environments, such as those in safety-critical or standards-based contexts, prioritise decisions differently. The consequence is downtime that extends beyond technical performance to legal, ethical or public safety risk (Kotlarsky et al. 2019; Ren et al. 2008). IS research further emphasises that the availability and structure of information and support mechanisms outline decision quality in post-adoption phases (Jasperson et al. 2005; Kotlarsky et al. 2019).

Knowledge fragmentation arises when critical system knowledge is distributed across multiple actors, systems, and repositories. This limits the actor's ability to understand system behaviour fully (Dissanayake et al. 2022). In enterprise ecosystems, such fragmentation is common due to specialised roles and distributed ownership, and has been identified as a key barrier to effective problem-solving and system maintenance (Dissanayake et al. 2022; Rydén Sonesson et al. 2021). This causes delayed diagnosis, inconsistent decisions, and increased reliance on informal knowledge sharing (Rydén Sonesson et al. 2021).

2.5.2 Prioritisation Heuristics and Escalation Dynamics

Decision-making depends on the support practices and coordination structures that determine which operational signals become visible and actionable under uncertainty. When system behaviour is not yet well understood in the early post-upgrade phases, decisions are made based on information that surfaces through escalation paths, incident records, and expert involvement. When these structures provide timely access to prior cases or known workaround patterns, decisions can be made with greater confidence despite incomplete information (Collopy et al. 2020; Costa et al. 2024; Kotlarsky et al. 2019; Sommerville et al. 2012). Conversely, when information travels through elongated or fragmented coordination paths, decision-makers must act on partial or distorted inputs, increasing reliance on local judgment and experience (Jacobides et al. 2024; Saxena & McDonagh, 2022). In such cases, decisions tend to prioritise immediate containment over systemic understanding, reinforcing reactive instead of anticipatory response to instability (Rydén Sonesson et al. 2021).

Perceived severity and operational consequences often drive decision-making signals. Prioritisation is based on situational context, such as whether core operations are affected and whether the affected environments are considered mission-critical (Adel et al. 2025). When urgency is high and formal escalation pathways are slow, issue framing becomes a strategic tool. Actors may emphasise severity to accelerate attention and bypass process delays. These practices demonstrate how issue representation directly outlines decision trajectories under time pressure. This suggests that issue severity is not always based on the failure, but is sometimes strategically framed by actors to impact escalation speed and decision attention under constrained organisational conditions(Kotlarsky et al. 2019). High operational impact prompts immediate triage decision, suspending ongoing work and redirecting resources toward restoring stability (Adel et al. 2025). Under these conditions, decisions favour rapid intervention over comprehensive diagnosis (Kotlarsky et al. 2019). Over time, repeated incidents cause organisations to recognise underlying, systemic problems when they treat issues as single events.

When the same problems occur frequently, organisations focus on standardising fixes, adjusting escalation rules, and creating shared reference solutions (Costa et al. 2024). In practice, issue frequency therefore guides longer-term prioritisation, while operational impact determines how urgently decisions are made in the moment. Together, these factors pressure organisations to balance immediate problem resolution with longer-term system stabilisation (Adel et al. 2025; Kotlarsky et al. 2019). While prior studies acknowledge prioritisation and escalation in post-adoption settings, they rarely distinguish between severity-driven urgency and frequency-based learning as distinct logics shaping decision-making under upgrade-related uncertainty (Kotlarsky et al. 2019).

2.5.3 Compatibility Mismatches and Sustained Uncertainty

Compatibility mismatches present significant challenges for decision-making because they arise from interactions across heterogeneous and partially hidden system dependencies (Anthony, 2016). With new version changes, upgrade failures often emerge in the integration section between components, making root causes difficult to identify (Collopy et al. 2020; Zhang et al. 2021).

Decision-makers must act without full visibility into how versions interact across configuration, customer environments, and external services (Adel et al. 2025). Since these dependencies are rarely exhaustively documented, compatibility issues are hard to predict and recur across contexts in inconsistent ways, such as using workarounds (Dumitraş & Narasimhan, 2009; Sommerville et al. 2012). Therefore, decisions are frequently made under uncertainty, relying on incremental testing, escalation, and post-deployment learning (Kotlarsky et al. 2019; Zhang et al. 2021).

The following section synthesises these insights into a coherent summary that guides the empirical investigation.

2.6 Summary of Literature Review

The literature shows that in post-adoption upgrade contexts, instability is rarely a purely technical phenomenon; rather, it emerges from breakdowns in sociotechnical alignment (Sarker et al. 2019; Techakriengkrai et al. 2021). A clear dependency chain models how such disruptions are identified, interpreted and managed in practice. Support practices form the foundational layer by determining the visibility, accessibility, and flow of operational knowledge (Adel et al. 2025; Jasperson et al. 2005). These support mechanisms, whether proactive or heavily reliant on reactive documentation and informal networks, directly constrain or enable coordination across distributed organisational roles. Coordination, in turn, formats decision-making. When support structures produce fragmented knowledge, cross-boundary coordination becomes reactive, forcing actors to make issue-resolution decisions under conditions of incomplete information and uncertainty (Kotlarsky et al. 2019).

This dependency chain is continuously triggered and complicated by the technical characteristics of enterprise IT ecosystems. Layered, modular architectures and heterogeneous configurations mean that technical issues rarely appear as isolated component errors. Instead, compatibility mismatches, integration dependencies, and context-specific configurations create cascading disruptions that cross organisational and technical boundaries and resist full prediction before deployment (Dumitraş & Narasimhan, 2009). Hence, issue characteristics such as recurrence and severe operational impact affect how the system's service and experience are perceived (DeLone & McLean, 2003; Zhang et al. 2021). Therefore, these issues feed directly into coordination and decision-making processes. Thereby influencing how urgently issues are escalated and whether actors prioritise quickly restoring operations over longer-term stabilisation(Kotlarsky et al. 2019; Rydén Sonesson et al. 2021).

Despite these insights, an important gap remains in the Information Systems literature on post-adoption software upgrades. Prior research has examined architectural complexity, post-adoption use, governance, and support structures, but has more often treated these dimensions separately than clarified how they interact in real operational settings to produce upgrade instability (Jasperson et al. 2005; Kotlarsky et al. 2019). In particular, limited attention has been given to how support practices structure information visibility, how distributed actors coordinate across knowledge boundaries, and how decisions are made under uncertainty during live upgrade-related disruptions (Figalist et al. 2019; Kotlarsky et al. 2019).

Addressing this gap requires linking observable operational patterns with the underlying sociotechnical dynamics that produce them. Accordingly, this study adopts a sequential

explanatory mixed-methods design to examine how upgrade instability emerges through the interplay between technical conditions, support practices, coordination across roles, and decision-making processes in enterprise IT ecosystems (Venkatesh et al. 2013).

To synthesise these theoretical perspectives and their relevance to the study, Table 1 provides a structured overview of key themes in the literature, core concepts, and their contributions to the research design, forming the foundation for the methodological approach outlined in the following chapter.

Table 1: Summary of Literature Review

Literature theme	Key concepts and findings	Key references	Relevance to this study
Enterprise IT ecosystems	Enterprise IT environments are complex, layered, and interdependent ecosystems. Modularity supports flexibility but also redistributes complexity across components and actors.	(Hanseth & Lyytinen, 2010; Jacobides et al. 2018; Sunyaev et al. 2023; Tiwana et al. 2010; Yoo et al. 2010)	Frames upgrade instability as an ecosystem-level issue rather than a single system failure.
Heterogeneity and configuration dependencies	Local configurations, third-party services, and compatibility dependencies create unpredictable upgrade outcomes.	(Dumitraș & Narasimhan, 2009; Sommerville et al. 2012; Strong & Volkoff, 2010; Zhang et al. 2021)	Supports the empirical focus on configuration and compatibility issues.
Post-adoption and upgrade instability	Long-term system value depends on continued use, adaptation, support, and maintenance. Upgrades are necessary but can disrupt system quality and service quality.	(Bhattacherjee, 2001; DeLone & McLean, 2003; Jasperson et al. 2005; Petersen & Seymour, 2021)	Defines upgrade instability as a post-adoption operational problem.
Sociotechnical support practices	Support practices shape how incidents are detected, interpreted, documented, escalated, and resolved. Support can be reactive or proactive, depending on the level of knowledge maturity.	(Costa & Fontão, 2024; Kotlarsky et al. 2015, 2019; Techakriengkrai et al. 2021)	Positions support as the main setting where upgrade instability becomes visible and manageable.
Coordination and knowledge boundaries	Responsibilities and expertise are distributed across roles. Coordination becomes difficult when knowledge is fragmented and ownership is unclear.	(Dissanayake et al. 2019; Kotlarsky et al. 2015, 2019; Malone & Crowston, 1994)	Provides the basis for analysing handoffs, escalation, and role dependencies.
Decision-making under uncertainty	Actors make decisions under incomplete information, time pressure, and limited system visibility. Severity and recurrence set up prioritisation.	(Adel et al. 2025; Collopy et al. 2020; Jasperson et al. 2005; Kotlarsky et al. 2019; Sommerville et al. 2012)	Explains why support actors must prioritise and escalate during unclear upgrade incidents.
Research gap	Prior research often discusses architecture, post-adoption support, coordination, and decision-making separately, but it also explains how they interact during live upgrade instability.	(Jasperson et al. 2005; Kotlarsky et al. 2019; Venkatesh et al. 2013, 2016)	Justifies the sequential explanatory mixed-methods design linking ticket patterns with dyadic interview explanations.

3 Research Methodology

This chapter outlines the research design choices for this study. It presents the research philosophy, research approach, data collection and analysis methods, ethical considerations, and the study's scientific quality.

3.1 Research Philosophy

This research aimed to identify and explain the sociotechnical factors contributing to upgrade instability in enterprise IT ecosystems. To support this aim, the study adopted a pragmatist research philosophy. Pragmatism was appropriate because it treated upgrade instability as a complex organisational problem requiring actionable, utility-driven knowledge (Goldkuhl, 2012; Morgan, 2014). This orientation aligned with empirical Information Systems research that seeks to generate contextually grounded and practically relevant insights into organisational phenomena (Ågerfalk, 2013; Venkatesh et al. 2016).

Ontologically, the study treated upgrade instability as a real operational condition observable through system failures, degraded performance, and recurring support incidents. At the same time, these events unfolded within organisational contexts shaped by coordination practices, governance arrangements, and role responsibilities. Accordingly, upgrade instability was understood as both a technical phenomenon and a coordination phenomenon. This reflects a sociotechnical perspective in which system outcomes emerge from the recursive interaction between technological capabilities and organisational practices (Baxter & Sommerville, 2011; Techakriengkrai et al. 2021). The study, therefore, conceptualised upgrade instability not merely as a technical failure but as an emergent sociotechnical outcome.

Epistemologically, the study generated knowledge through purposeful problem solving and inquiry. Knowledge was produced through two interconnected processes. First, a diagnostic analysis of aggregated support incidents provided measurable insight into where upgrade instability occurred and which patterns required prioritisation. Second, qualitative engagement with ecosystem actors offered an interpretive understanding of why coordination gaps and organisational dynamics contribute to these failures. Combining these forms of evidence empowered the development of empirically grounded meta-inferences that link observable system behaviour with underlying coordination mechanisms (Ågerfalk, 2013; Venkatesh et al. 2016).

In line with pragmatism, the validity of knowledge was assessed based on its ability to provide credible and useful explanations of real-world phenomena (Morgan, 2014). The study, therefore, focused on producing actionable insights into the mechanisms through which upgrade instability emerged and was managed in practice (Goldkuhl, 2012). This ensured both practical relevance and theoretical contribution to the understanding of enterprise IT ecosystems.

3.2 Research Approach

This study adopted a mixed-methods research approach as its primary research strategy. The objective of the study was to empirically investigate and analyse upgrade instability in enterprise IT ecosystems by integrating quantitative and qualitative evidence. A mixed-methods approach was appropriate because it enabled a comprehensive understanding of both observable operational patterns and the underlying organisational dynamics that embody them (Venkatesh et al. 2016).

Compared to alternative research designs, such as purely quantitative or qualitative approaches, mixed-methods research was particularly suitable because upgrade instability is inherently sociotechnical. Quantitative analysis alone could identify recurring patterns in support incidents but could not clarify why they occur, while qualitative approaches alone could lack grounding in operational data. By combining both approaches, the study ensured a more robust and holistic understanding of the phenomenon (Ågerfalk, 2013; Venkatesh et al. 2016).

The study followed a sequential mixed-methods design, where quantitative analysis was conducted first, followed by a qualitative investigation (Venkatesh et al. 2016). This design allowed findings from quantitative data to both inform and delimit the subsequent qualitative inquiry. Specifically, the semi-structured dyadic interviews were defined by the key patterns and dimensions identified in the quantitative phase. This ensured that the qualitative phase focused specifically on demonstrating the most significant patterns observed in the data (Ågerfalk, 2013; Venkatesh et al. 2016).

In the first phase, the study adopted a diagnostic orientation. A quantitative analysis of aggregated support incidents identified recurring patterns of upgrade-related failures, including their characteristics and operational impact (Zhang et al. 2021). These findings provided an empirical baseline, highlighting key drivers of instability and directly informing the design of subsequent qualitative dyadic interviews with technical subject matter experts and support specialists involved in the ecosystem's upgrade support process. This grounded the qualitative discussions in observed failure patterns, ensuring that participants reflected on concrete operational situations.

In the second phase, qualitative data were collected through semi-structured dyadic interviews to explore how organisational roles, support practices, coordination across roles, decision-making, and stakeholder experiences contribute to upgrading instability. This method was appropriate because dyadic interviews combine the depth of individual interviews with interaction between participants, enabling participants to build on shared experiences while also revealing differences in interpretation and decision-making (Lobe et al. 2022; Morgan et al. 2013). This makes the method particularly suitable for examining coordination across organisational roles and boundaries, where shared understandings, handoffs, and role dependencies dominate how IT work is carried out in practice (Kotlarsky et al. 2019). The term dyadic interviews is used to reflect the two-participant format of the sessions, which supports more focused interaction and detailed probing than larger group-based formats (Morgan et al. 2013). This second phase aimed to describe and contextualise the patterns identified in the quantitative analysis, providing deeper insight into the sociotechnical mechanisms underlying instability .

Finally, the study integrated findings from both phases to develop meta-inferences that link technical failure patterns with organisational and coordination-related factors (Venkatesh et al. 2013, 2016). This integration facilitated a comprehensive explanation of how upgrade instability emerges in enterprise ecosystems (Mingers, 2001). In line with case study research, such explanations support analytical generalisation to theoretical propositions beyond the specific empirical context (Yin, 2018, pp.37–38).

3.3 Data Collection Method

The data collection strategy was designed to align directly with the study's research design and empirical research objective of explaining upgrade instability in enterprise ecosystems. The objective was to generate empirically grounded insights that support the identification and explanation of sociotechnical factors contributing to upgrade instability. Consistent with mixed-methods research principles, the study combined quantitative evidence from operational support incident data with qualitative insights from ecosystem actors, thereby capturing both the technical manifestations and the underlying sociotechnical coordination mechanisms of upgrade instability (Ågerfalk, 2013; Baxter & Sommerville, 2011; Venkatesh et al. 2013, 2016).

The empirical setting consisted of a large-scale enterprise IT ecosystem operated by the collaborating organisation. This context provided access to anonymised support ticket records and relevant stakeholders across the upgrade support chain. The unit of analysis was defined as an upgrade instability event, which is understood as an operational disruption occurring after a system version change and documented through support incidents. Focusing on discrete instability events within a single ecosystem enabled the tracing of recurring patterns across interdependent components and organisational boundaries, consistent with prior Information Systems research on post-adoption system dynamics (Jasperson et al. 2005; Khoo et al. 2011).

The data collection process followed the logic outlined in the research approach, where quantitative analysis informed the qualitative inquiry. This anchored the dyadic interviews in empirically observed instability patterns and strengthened traceability between support ticket findings and their qualitative explanation (Venkatesh et al. 2016).

3.3.1 Selection of Respondents

Respondent selection followed a purposive sampling strategy to capture perspectives from key actors involved in upgrade processes across the ecosystem. Participants were selected based on their direct experience with upgrade-related incidents and their roles in diagnosing, resolving, or managing such events. The target groups included technical subject matter experts and support specialists involved in the upgrade support process, as these roles represented critical coordination points where instability in the upgrade frequently arises.

This selection strategy was grounded in prior research demonstrating that upgrade failures often arise at organisational boundaries and during handoffs between roles as opposed to from isolated technical defects (Kotlarsky et al. 2019). By including participants from multiple roles, the study captured diverse perspectives on coordination, responsibility, and decision-making. This diversity was essential for uncovering sociotechnical dynamics and ensuring a

comprehensive understanding of coordination across organisational boundaries (Baxter & Sommerville, 2011; Kotlarsky et al. 2019).

Participants were recruited in collaboration with the partner organisation while ensuring voluntary participation and minimising power asymmetries, which is critical for generating authentic and unbiased accounts in organisational research (Myers & Newman, 2007; Walsham, 2006). Selection prioritised individuals with recent experience of upgrade-related incidents to ensure that discussions were grounded in concrete operational situations.

3.3.2 Quantitative Data Collection (Support Incident Data)

The first empirical data source was an anonymised, aggregated dataset of support incidents extracted from the organisation's support ticket system. The dataset included tickets associated with system upgrades, version changes, or post-upgrade failures. Case selection was guided by documented upgrade events and the operational disruptions that followed. The use of unobtrusive, archival operational data as an empirical basis is well established in Information Systems research, providing objective insights into organisational processes and system performance without the reactive measurement biases of traditional surveys (Mithas et al. 2011).

Given the unstructured nature of support ticket data, which functions as rich, text-heavy digital traces of organisational operations, an AI-assisted extraction approach was employed to transform these textual records into structured analytical variables. The objective of this step was to convert ticket descriptions, comments, and resolution notes into a consistent dataset suitable for quantitative and pattern-based analysis. Within Information Systems research, the use of text analytics and machine learning techniques to extract structured representations from unstructured digital traces, such as operational logs and user-generated problem reports, is well established as a rigorous methodological approach (Abbasi et al. 2018; Chen et al. 2012; Müller et al. 2016).

Building on this foundation, an initial filtering step (Stage 1) was applied to identify upgrade-related tickets before structured classification, as part of the broader data preparation process required for working with unstructured textual data. Tickets were classified as upgrade-related only when the case description clearly linked the reported issue to a system version change. This ensured that classification was based on explicit or strongly implied evidence, rather than keyword presence alone. Following this filtering, a structured data schema (Stage 2) was developed to guide the extraction process (see Appendix 2: Support Ticket Data Extraction and Classification Schema), capturing key analytical dimensions of the support ticket lifecycle, including failure mechanisms, issue categories, operational impact, escalation behaviour, and documentation effectiveness. This supported systematic categorisation and comparability across cases, strengthening the empirical analysis (Debortoli et al. 2016). Although documentation effectiveness was included in the initial extraction schema, the ticket data did not provide sufficiently consistent evidence to support a reliable quantitative distribution for this variable. Documentation and support limitations were therefore examined through the Technical Support Handbook, an organisational artefact, and further explored in the qualitative phase.

To avoid over-interpreting the support ticket data, the classification followed an explicit-evidence approach. Variables were assigned only when the ticket text provided clear or strongly supported evidence. Where the available information was insufficient to assign a reliable

value, the variable was classified as “Unknown”. “Unknown” values were treated as variable-level missingness, meaning that the ticket remained in the dataset but was excluded from the distribution for that specific variable. This approach was used to reduce unsupported inference and maintain consistency across the dataset. It was particularly important because the study analysed unstructured operational text, where records may contain uneven levels of detail, ambiguous language, and limited contextual information for some analytical dimensions (Abbasi et al. 2018; Debortoli et al. 2016; Müller et al. 2016).

The Stage 2 categories were developed inductively through an initial review of the support tickets, recurring terms in the dataset, and categories used in the organisation’s support context. The aim was to ensure that the classification reflected the empirical structure of the ticket data rather than imposing categories directly from prior literature. Existing research was used as a sensitising background to guide the interpretation of the extracted variables. This follows the logic of sensitising concepts, where prior concepts help guide the analysis without forcing the data into fixed categories (Bowen, 2006). Literature on upgrade failures and distributed systems helped sensitise the analysis to version dependencies, configuration mismatches, component interactions, and post-deployment behaviour (Dumitraş & Narasimhan, 2009; Sommerville et al. 2012; Zhang et al. 2021). The Information Systems Success model supported the interpretation of operational impact and service disruption (DeLone & McLean, 2003), while research on post-adoption support, incident handling, and coordination in IT support work informed the interpretation of escalation behaviour and support-related variables (Jasperson et al. 2005; Kotlarsky et al. 2019; Retana et al. 2018). In this way, the classification remained grounded in the empirical data while being theoretically informed by established Information Systems literature.

The AI model was prompted with structured instructions and clear category definitions, and with constrained output formats to ensure consistency. In line with the explicit-evidence rule, the model assigned the most appropriate dominant category only where the ticket text provided sufficient evidence. Otherwise, the variable was classified as “Unknown” rather than being forced into a category. This approach follows a rule-guided content analysis logic, in which coding depends on clear category definitions, explicit coding rules, and a sufficient fit between the text and the category. It also reduces the risk of unsupported AI-generated classification, which is important because AI-based text analysis may produce imprecise categorisations when prompts and coding rules are not tightly constrained (Hsieh & Shannon, 2005; Mayring, 2025). The extraction process was applied uniformly across the dataset, enabling comparability across cases. Outputs were aggregated into a structured dataset, allowing for subsequent analysis of patterns such as issue category distributions, operational impact, escalation behaviour, and resolution patterns. Such approaches enable researchers to make sense of large volumes of unstructured, human-generated text while preserving critical contextual signals necessary for interpreting organisational processes (Abbasi et al. 2018).

To enhance the reliability of the AI-assisted classification, a sample of classified tickets was manually reviewed against the original ticket text and the classification schema. This verification assessed whether assigned categories were supported by explicit evidence, whether “Unknown” was used appropriately where evidence was insufficient, and whether the classification rules were applied consistently across cases. A manual audit of 50 randomly selected tickets showed 98% alignment between the assisted classification and the manual review. Where inconsistencies were identified, the classification instructions were refined, and the outputs were rechecked before aggregation. This process reduced variability in interpretation

and helped ensure that the extracted dataset was sufficiently robust for downstream analysis while maintaining scalability across a large volume of support tickets. Consistent with prior research showing that structured analysis of upgrade failures can reveal recurring fault patterns in enterprise and distributed systems, this process supported the identification of recurring instability patterns in the support ticket data (Dumitraş & Narasimhan, 2009; Zhang et al. 2021).

3.3.3 Qualitative Data Collection (Semi-Structured Dyadic Interviews)

The second empirical data source consisted of qualitative data collected through semi-structured dyadic interviews with purposively selected participants. Two dyadic interviews were conducted with participants from support specialist and software engineering roles. This method was selected because it facilitates interaction between participants while allowing for a more intensive focus on specific technical cases than is possible in larger focus group settings. Dyadic interviews combine the depth of individual interviews with interaction between participants, allowing participants to respond to and build on each other's contributions and co-construct explanations of their experiences (Morgan et al. 2013).

The dyadic interview discussions were guided by a semi-structured protocol developed after the quantitative analysis phase. This ensured that discussions were anchored in empirically identified instability cases, thereby strengthening methodological integration (Venkatesh et al. 2016). More specifically, the dyadic interview discussions were explicitly grounded in patterns identified from the support ticket analysis. Participants were asked to reflect on the specific issue categories, impact levels, escalation patterns, and documentation limitations observed in the quantitative data. In this way, the quantitative phase delimited the scope of the dyadic interviews by narrowing the discussion to the most relevant patterns of instability.

To further support this process, visual representations of the quantitative findings (e.g., issue categories, escalation patterns, and distributions of operational impact) were used during the dyadic interviews to guide discussion and elicit detailed, experience-based explanations. This approach draws on the broader qualitative tradition of visual elicitation, where visual artefacts are used during interviews to prompt comments and discussion (Banks & Zeitlyn, 2015, p.87), and more specifically applies graphic elicitation techniques in which data visualisations and analytical representations are used as stimuli to prompt richer reflection (Crilly et al. 2006). In complex sociotechnical contexts such as enterprise IT ecosystems, graphic elicitation is particularly valuable as it helps participants engage with aggregated data patterns and relate them to their own situated experiences, thereby enhancing the depth and specificity of qualitative insights.

Consistent with the research design, the dyadic interviews were used to explore the mechanisms underlying the instability patterns identified in the quantitative phase. Participants were asked to describe why certain types of issues were particularly difficult to resolve, how decisions were made under conditions of incomplete information, and how coordination across roles influenced resolution outcomes in interdependent environments (Kotlarsky et al. 2019). This approach enabled the study to move beyond descriptive accounts and examine potential sociotechnical explanations for the observed patterns, focusing on areas such as coordination dynamics and decision-making under uncertainty (Baxter & Sommerville, 2011; Venkatesh et al. 2016).

All sessions were recorded and transcribed with participant consent. Questions were neutrally framed and grounded in real cases to mitigate recall bias and socially desirable responses. In addition, organisational artefacts such as release notes and internal support documentation, specifically the Technical Support Handbook, were collected to compare formal procedures with enacted practices (Jakobsen & Vanhaverbeke, 2025; Khoo & Robey, 2007). The interactive nature of dyadic interviews facilitated the emergence of shared understandings, differences in perspective, and collectively constructed explanations of recurring patterns of instability. This interaction allowed participants to build on each other's experiences and refine interpretations in real time, strengthening the interpretive validity of the findings (Morgan et al. 2013).

3.3.4 Data Collection Timeline

The data collection process followed a structured timeline aligned with the research design. The first phase (weeks 13–15) focused on quantitative data extraction and preparation, including AI-assisted structuring of support ticket data and initial frequency analysis. This phase identified key instability patterns and operational priorities.

The second phase (weeks 16–18) involved developing the qualitative data collection instrument, recruiting participants, and conducting semi-structured dyadic interviews. Transcription and initial familiarisation with qualitative data were conducted immediately after each session to support timely analysis and allow emerging insights to inform subsequent data collection.

This staged approach ensured methodological coherence, an efficient use of limited research time, and insights into the sociotechnical mechanisms underlying upgrade instability.

3.4 Data Analysis Method

The data analysis strategy was designed to integrate quantitative and qualitative data into coherent empirical insights that explain upgrade instability. The analysis was conducted in stages to connect quantitative incident patterns with qualitative explanations. This structure supported traceability between the support ticket findings, interview themes, and integrated meta-inferences.

3.4.1 Quantitative Analysis

The support ticket data was analysed using descriptive statistics and categorisation techniques to identify recurring issue types, affected components, and common failure patterns. The analysis began with the quantitative examination of the structured support incident dataset derived from the AI-assisted extraction process. The primary analytical technique used in this phase was descriptive frequency analysis, which involved categorising instability events by variables such as issue category, operational impact, and escalation behaviour, as defined in the data schema (see Appendix 2). For each variable, cases classified as "Unknown" were retained in the working dataset for transparency but excluded from percentage distributions where the relevant variable could not be reliably determined. As a result, the denominator

varied across some figures depending on the number of cases with sufficient evidence for that specific variable.

The ticket classification was conducted in batches, with category distributions reviewed across batches to assess whether stable issue patterns had emerged. Pattern stability was considered reached when additional batches did not show new major categories and the proportional distribution of the main categories remained consistent. The purpose of this approach was not to establish statistical representativeness, but to ensure the analytical subset captured the most frequent and dominant patterns of upgrade instability observed in the support tickets.

This analysis provided a systematic overview of which types of upgrade instability occurred most frequently and which had the greatest operational consequences. Prior research shows that such log-based analyses are effective in identifying recurring failure patterns in complex systems (Dumitraş & Narasimhan, 2009; Zhang et al. 2021). The outcome of this phase was a set of empirically grounded instability patterns that informed subsequent qualitative analysis.

3.4.2 Abductive Thematic Coding

Dyadic interview data were analysed using thematic analysis, supported by pattern coding techniques (Miles et al. 2014), to identify recurring themes related to coordination, responsibility, and organisational factors contributing to instability. Following the quantitative phase, the qualitative analysis was conducted abductively, involving iteration between empirical observations and theoretical concepts to demonstrate underlying coordination mechanisms (Goldkuhl, 2012; Morgan, 2014). This explanatory phase contextualised the quantitative findings and supported the development of theoretically grounded meta-inferences (Venkatesh et al. 2013).

The coding process was conducted in two stages. First, open coding was used to identify key practices, challenges, and experiences described by participants. Second, pattern coding was applied to group these codes into higher-level themes reflecting coordination breakdowns, decision-making dynamics, and role interactions (Miles et al. 2014).

The analysis was guided by sensitising concepts from prior literature, including ecosystem coordination (Kotlarsky et al. 2019) and decision support (Silver, 1991), while remaining open to emergent insights. Qualitative themes were then systematically integrated with quantitative findings to generate meta-inferences linking technical instability patterns to organisational coordination challenges (Venkatesh et al. 2013, 2016). These integrated insights provided a comprehensive explanation of how sociotechnical factors interact to produce upgrade instability in enterprise ecosystems.

3.4.3 Integration

Findings from both phases were integrated to develop a comprehensive understanding of upgrade-related instability, linking observed technical patterns with underlying sociotechnical explanations. This integration facilitated the development of meta-inferences that connect quantitative patterns with qualitative insights, thereby supporting analytical generalisation to theoretical propositions (Venkatesh et al. 2013; Yin, 2018, pp.37–38). Table 2 provides an

overview of the quantitative, qualitative, and integration phases, showing how each phase contributed to the overall mixed-method design.

Table 2: Overview of Quantitative, Qualitative, and Integration Phases

Phase	Objective	Method
Quantitative phase	Identify patterns of upgrade instability	Incident analysis (AI-assisted extraction)
Qualitative phase	Explain sociotechnical coordination mechanisms	Semi-structured dyadic interviews with ecosystem actors
Integration phase	Develop meta-inferences linking patterns and explanations	Mixed-method integration

3.5 Scientific Quality

Ensuring scientific rigour is paramount when conducting a sequential explanatory mixed-methods case study. Rigour is achieved through the systematic integration of quantitative and qualitative evidence, transparent analytical procedures, and robust interpretive validation (Venkatesh et al. 2013; Yin, 2018). Accordingly, the scientific quality of this study is discussed in terms of methodological rigour, analytical reliability, and interpretive validity.

3.5.1 Methodological Rigour (Integration and Triangulation)

Methodological rigour concerns the extent to which the study accurately captures and clarifies the complexity of upgrade instability within the enterprise ecosystem. To strengthen construct validity, the study employed methodological triangulation, combining quantitative evidence from incident records with qualitative insights from dyadic interviews and supporting documentation (Recker, 2013, pp.70–71, 91; Venkatesh et al. 2016). This multi-source evidence reduced the risk of single-method bias and enabled examination of recurring instability patterns from both operational and coordination perspectives. Triangulation was further strengthened by explicitly delimiting the qualitative phase by the quantitative findings, allowing the second method to explain and validate the patterns identified in the first phase. This established convergence among observed instability patterns, escalation behaviour, and actor-based explanations, thereby strengthening the robustness of the findings.

3.5.2 Analytical Reliability (Traceability and Transparency)

Analytical reliability refers to the systematic process through which empirical findings are documented, coded, and integrated. To ensure transparency and traceability, the study maintained a clear “chain of evidence” connecting raw operational data to final theoretical meta-inferences (Yin, 2018, pp.134–136). To support traceability, an organised evidence base was maintained for the study, including anonymised support ticket extracts, interview transcripts, the interview guide, and analytical coding outputs. This approach is consistent with case-study guidance on maintaining an evidence database to organise and document research

records (Recker, 2013, pp.97–98). During qualitative analysis, researchers collaboratively reviewed coding decisions to minimise subjective bias and ensure that emerging themes accurately reflected the coordination challenges described by participants (Miles et al. 2014; Recker, 2013, p.86).

3.5.3 Interpretive Validity and Analytical Generalisation

Interpretive validity concerns the extent to which the findings provide a plausible, authentic, and theoretically sound explanation of the phenomenon (Klein & Myers, 1999; Walsham, 2006). In this study, interpretive validity was strengthened by anchoring the qualitative analysis in empirically identified quantitative patterns, ensuring that the concluding themes reflect observed operational realities.

Finally, the study aimed to produce analytical generalisation. By developing theoretically grounded meta-inferences, the study elevated empirical observations into broader propositions that can inform upgrade coordination practices and theoretical discourse in similar enterprise ecosystems (Tsang, 2014; Yin, 2018, pp.37–38).

3.6 Ethical Considerations

Ethical considerations in this study focused on confidentiality, voluntary participation, and responsible handling of sensitive organisational data. Since the study involved a collaborative industry partner and enterprise systems, particular care was taken to protect both organisational information and the professional standing of individual participants.

During the quantitative phase, support incident analysis required strict data protection. To mitigate exposure risks, all enterprise logs were aggregated and abstracted, with company-specific data entirely black-boxed to ensure confidentiality. The organisation's name was anonymised in the public thesis, and internal operational processes, system architecture, and proprietary algorithms were not disclosed. A formal review process verified confidentiality without influencing the academic interpretation of the findings. AI-assisted data extraction was governed by strict data-handling procedures in line with organisational data protection policies, ensuring that no sensitive enterprise data was exposed to external or public models. These measures are consistent with established ethical practices in Information Systems research involving organisational data (Stahl et al. 2014).

For the qualitative phase, protecting participants was the primary mandate. Discussing coordination failures can carry significant professional risk for IT staff operating in high-pressure environments (Kotlarsky et al. 2019). To mitigate power asymmetries, recruitment avoided direct nominations from supervisors. Participation was voluntary and based on explicit informed consent to protect disclosures and minimise professional risk (Myers & Newman, 2007). All results were presented at an aggregated role- and theme-level, with transcripts de-identified to remove any potentially identifying information, and using carefully anonymised quotes to prevent traceability. In line with interpretive research ethics, particular care was taken to ensure that participant accounts cannot be linked to identifiable individuals or specific organisational actions, while preserving the contextual richness necessary for meaningful interpretation (Walsham, 2006).

Finally, ethical responsibility extends to the interpretation and reporting of findings. Given the case-based nature of the study, care was taken to avoid attributing causality or responsibility to specific individuals, teams, or organisational units. Findings were presented as analytically generalisable insights for organisational performance. This approach ensures that the study contributes to theoretical understanding while minimising reputational and operational risks for the participating organisation (Walsham, 2006; Yin, 2018).

4 Empirical Findings and Analysis

This chapter presents the empirical findings from the support ticket analysis and the dyadic interviews. It combines observed patterns of upgrade-related instability with qualitative explanations of how these patterns emerge in practice. All empirical findings in this chapter are presented in aggregated and anonymised form to ensure organisational confidentiality. No individual customer, employee, ticket, system environment, or organisational unit is identifiable in the reported figures.

4.1 Quantitative Analysis of Support Incidents

The quantitative analysis is based on a structured dataset derived through a two-stage extraction process. First, support tickets were filtered to identify upgrade-related incidents in Stage 1 as defined in the Support Ticket Data Extraction and Classification Schema (Appendix 2). Second, the filtered dataset was classified into structured variables using the structured schema (Stage 2), also outlined in detail in Appendix 2. The results presented below focus on patterns observed across these structured variables, specifically examining issue categories, operational impact, and escalation behaviour.

4.1.1 Dataset Construction and Filtering (Stage 1)

The quantitative analysis began with a filtering process to extract upgrade-related tickets from the total support tickets (Stage 1). A total of 7,829 tickets related to a specific software product were received over six months from October 2025 to March 2026. From this initial dataset, 3,054 were identified as upgrade-related during the Stage 1 filtering process, representing 39% of the total volume. This substantial proportion indicates that, within the analysed six-month product dataset, upgrade-related issues represented a significant share of support activity.

Analysis of this filtered subset (n=3,054) also revealed notable patterns regarding resolution outcomes and customer engagement. Within the subset, 1,668 tickets were reported as solved, representing 55% of the upgrade-related cases. Within the same filtered subset, 583 tickets were marked as having no customer response, representing 19% of the subset. These status indicators provide an initial view of the support lifecycle surrounding upgrade-related incidents. The solved cases show that many upgrade-related tickets reached formal closure within the support process. At the same time, the presence of tickets with no customer response indicates that some cases depended on additional customer input, such as logs, configuration details, or confirmation of system behaviour, before they could be further diagnosed, verified or closed. These patterns are important because they show that upgrade instability is not only expressed through technical failure categories, but also through the coordination required to diagnose, verify and close incidents. This pattern provides an empirical basis for the subsequent qualitative analysis of customer input, missing information, and coordination among support actors.

The filtered subset of 3,054 tickets from Stage 1 served as the empirical basis for categorisation in Stage 2 (see Appendix 2), which is presented in the following sections.

4.1.2 Categorisation of Upgrade Instability Events (Stage 2)

This section presents the results of the Stage 2 classification, identifying the technical domains affected by upgrade-related instability. Stage 2 classification was conducted in batches of 50 tickets from the filtered dataset of 3,054 upgrade-related tickets. After 600 tickets had been classified, repeated classification showed stable issue patterns, and no new major issue categories emerged. Pattern stability was assessed by comparing category distributions across batches and checking whether new batches introduced new issue types or changed the dominance of the main categories. This 600-ticket subset, therefore, serves as the basis for the quantitative findings presented in this section.

The issue categories used in Stage 2 were developed inductively through iterative review of the ticket content, recurring terms in the dataset, and the organisation's specific support context. Existing literature on upgrade failure, configuration dependency, distributed system behaviour, and support practices was used as a sensitising background to guide interpretation while keeping the categories grounded in the ticket data (Dumitraş & Narasimhan, 2009; Jasperson et al. 2005; Kotlarsky et al. 2019; Sommerville et al. 2012; Zhang et al. 2021). The final categories were service and application failure, database or storage issue, licensing and registration, component-specific failure, configuration or compatibility issue and Unknown.

Service and application failures are incidents in which the upgraded service or application failed to start, function, or remain available. Database or storage issues refer to failures involving stored data, database access, recording storage, or related persistence functions. Licensing and registration refer to problems with license activation, registration, ownership, or entitlement after an upgrade. Component-specific failure relates to a particular system component or module. A configuration or compatibility issue is an incident caused by mismatches among versions, customer-specific settings, dependencies, integrations, or environment-specific conditions. Unknown was used when the ticket lacked sufficient evidence to assign a clear technical domain.

The findings indicate that instability is not evenly distributed across system components but is concentrated in specific areas associated with system configuration, licensing, and component-level dependencies.

4.1.2.1 Distribution of Upgrade Phases

The distribution of upgrade phases is presented in Figure 2. This classification shows the stage at which incidents occur, distinguishing between those that occur before, during, and after the upgrade. In line with the classification approach described in Chapter 3, upgrade phases were assigned only when the timing of the incident relative to the upgrade was evident in the ticket data. As a result, 551 out of 600 cases yielded a definitive upgrade phase, while the remaining 49 cases were classified as "Unknown" due to insufficient evidence, and these are excluded from this distribution.

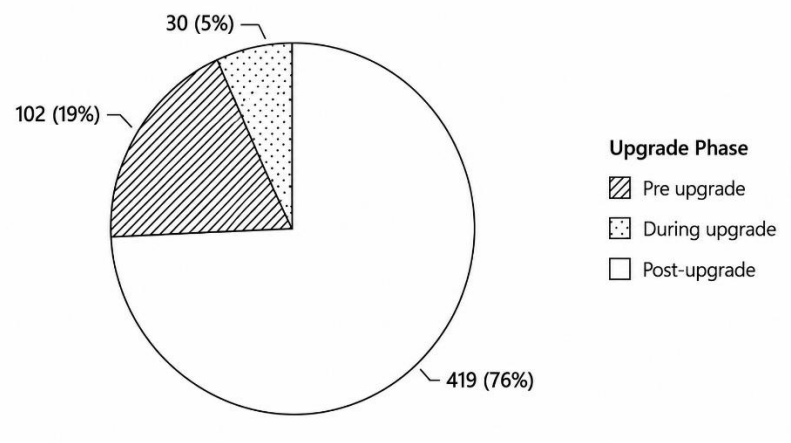


Figure 2: Distribution of Upgrade Phases in Upgrade-Related Incidents (n =551)

The results show that most upgrade-related incidents occurred after the upgrade was complete. Post-upgrade incidents account for 419 out of 551 classified cases, representing 76% of the definitive upgrade-phase dataset. In comparison, pre-upgrade incidents account for 102 of 551 cases (19%), while incidents occurring during the upgrade process account for 30 of 551 cases (5%).

The results indicate that upgrade instability is most often observed as a post-upgrade event. This suggests that many problems become visible only when the upgraded system is used in the customer's operational environment. The post-upgrade concentration is therefore important because it shows that instability is not limited to the upgrade activity itself, but also concerns the operational period after implementation.

4.1.2.2 Distribution of Issue Categories

The distribution of issue categories identified through the Stage 2 classification is presented in Figure 3. In line with the classification approach described in Chapter 3, issue categories were assigned only when the responsible technical domain was evident in the ticket data. As a result, 575 out of 600 cases yielded a definitive issue category, while the remaining 15 cases were classified as "Unknown" due to insufficient evidence and are excluded from this distribution.

The classified issue categories were defined as follows. Configuration and compatibility issues refer to cases where setup, environment, access, network, security, platform, or compatibility conditions prevent the system from working correctly. Licensing and registration issues include problems with licences, entitlements, registration, onboarding, or organisation linking that prevent correct system activation or use. Service and application instability or failure refers to cases where a core service or application cannot start, stops running, crashes, restarts, or fails during use. Database-related and storage issues cover problems involving databases, recording storage, logs, backups, restores, or storage access that affect system operation. Component-specific failures refer to cases where a named component fails, is missing, or becomes unavailable.

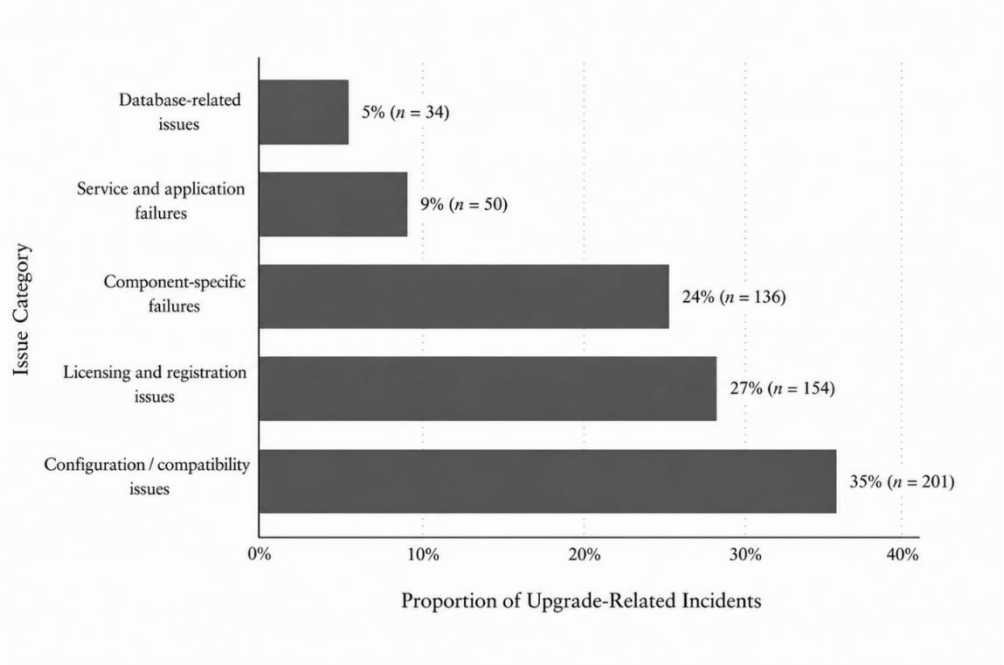


Figure 3: Distribution of Issue Categories in Upgrade-Related incidents (n = 575)

The analysis of issue categories reveals that upgrade-related incidents are predominantly concentrated in a limited number of technical domains. The largest category is configuration and compatibility issues, accounting for 35% (n = 201) of all cases. This is followed by licensing and registration issues (27%, n = 154) and component-specific failures (24%, n = 136). In contrast, service and application failures (9%, n = 50) and database-related issues (5%, n = 34) represent the least frequent categories.

This distribution indicates that upgrade instability is primarily associated with environment-specific conditions and integration dependencies. Configuration and compatibility issues, licensing and registration issues, and component-specific failures together account for 491 of the 575 classified cases, representing 85% of the identifiable issue-category dataset. In contrast, service and application failures and database-related issues together account for 84 of the 575 classified cases, representing 15% of the identifiable dataset.

These findings suggest that upgrade instability is most frequently observed at points where the system interacts with heterogeneous configurations, licensing arrangements, and component-level dependencies, rather than mainly within central application or database components. This concentration of issue categories provides an empirical basis for examining how such integration-intensive challenges are managed in practice, a further exploration in the qualitative analysis.

4.1.2.3 Operational Impact and Severity

The distribution of operational impact scores across the categorised incidents is presented in Figure 4. In line with the classification approach described in Chapter 3, impact scores were assigned only where the operational consequences were explicitly stated or strongly supported in the ticket data. As a result, 396 out of the 600 classified tickets yielded a definitive impact score, while 204 out of 600 cases were classified as “Unknown” due to insufficient evidence and are excluded from the impact-score distribution. This exclusion is methodologically necessary because assigning an impact score without clear evidence would reduce classification

reliability. However, the missing impact data cannot be treated as necessarily random. It may reflect differences in the consistency with which operational consequences were documented in the ticket records. Therefore, the results in this subsection should be interpreted as describing the 396 cases with identifiable impact, rather than the full 600-ticket subset.

The impact score was used to classify the severity of disruption based on predefined criteria. Scores 1 and 2 represent minor disruption, where the incident caused limited inconvenience or had a low operational impact. Score 3 represents degraded operation, where the system remained usable but with reduced performance, instability, or functional limitations. Score 4 represents a major operational limitation affecting production use, while Score 5 represents a critical disruption where system functionality became unavailable.

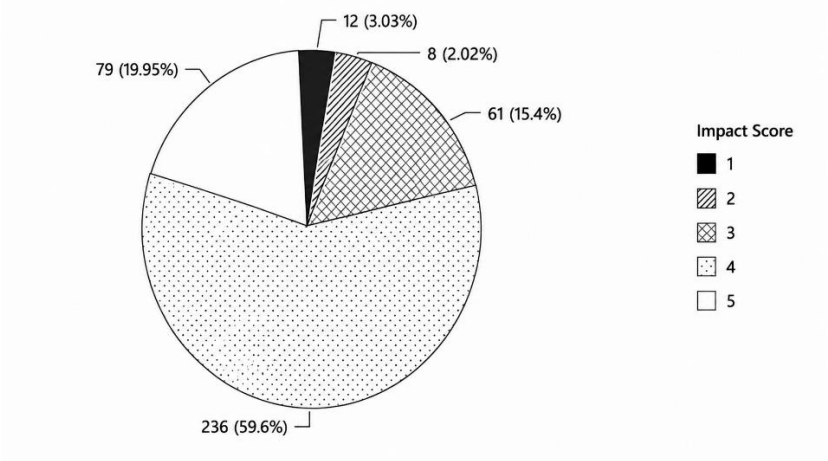


Figure 4: Distribution of Impact Scores for Upgrade-Related Incidents (n=396)

The analysis shows that the majority of the 396 cases with identifiable impact fall into the higher-impact categories. Score 4, which represents major operational limitations affecting production use, accounts for 236 out of 396 identifiable impact cases, representing 59.6%. Score 5, where system functionality becomes unavailable, accounts for 79 out of 396 identifiable impact cases, representing 19.95%. Combined, Score 4 and Score 5 incidents account for 315 out of 396 identifiable impact cases, representing approximately 80% of the identifiable impact-score dataset. In contrast, Score 3 accounts for 61 of 396 cases (15.4%), while Scores 1 and 2 together account for a smaller share of the identifiable cases, at approximately 5%.

These results indicate that, among cases where operational impact could be identified, upgrade-related incidents were more often documented as major or critical disruptions than as minor inconveniences. This suggests that upgrade instability is not limited to small technical errors, but is often connected to disruptions that affect production use, system availability, or operational continuity. At the same time, because 204 of the 600 classified tickets did not contain enough information to assign an impact score, this conclusion applies only to the identifiable impact subset and should be interpreted cautiously. The finding nevertheless provides an empirical basis for the qualitative phase, where high-impact incidents are used to examine how support actors prioritise, coordinate, and make decisions under operational pressure.

4.1.2.4 Relationship Between Issue Categories and Operational Impact

To examine how different technical domains relate to operational severity, a cross-tabulation was conducted between the distribution of issue categories (Figure 3 in section 4.1.2.2) and

selected impact scores (Figure 4 in section 4.1.2.3). The cross-tabulation focuses only on Impact Score 5 and Impact Score 3 because they represent two analytically distinct forms of upgrade instability. As already highlighted in section 4.1.2.3, Impact Score 5 captures a critical system failure in which system functionality becomes unavailable, while Impact Score 3 captures moderate degradation where the system remains usable but with reduced performance. Comparing these two levels is useful because critical incidents are likely to set up urgent escalation and immediate prioritisation, while moderate incidents may become important when they recur and create repeated support demand. Figure 5 shows the distribution of issue categories for impact score 5, while figure 6 shows the distribution of issue categories for impact score 3.

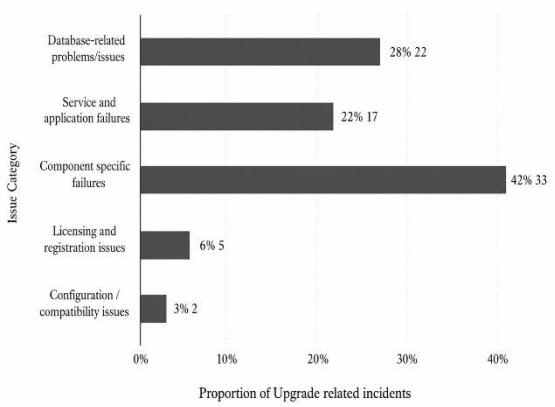


Figure 5: Issue Category Distribution for Impact Score 5 (n = 79)

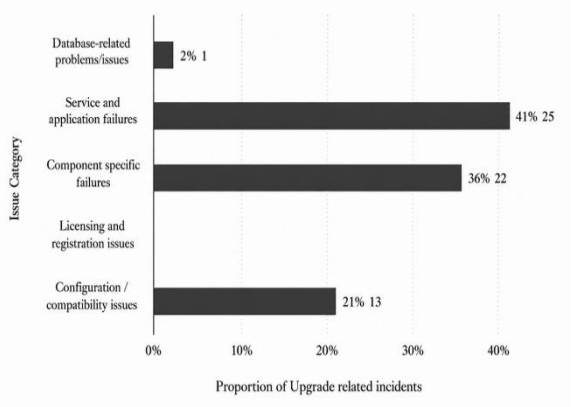


Figure 6: Issue Category Distribution for Impact Score 3 (n = 61)

For incidents with an Impact Score of 5, Figure 5 shows that the distribution is concentrated in issue categories that directly affect system availability and core functionality. Component-specific failures account for the largest share of critical incidents, at 42% (n = 33). Database-related issues account for 28% (n = 22), while service and application failures account for 22% (n = 17). In contrast, configuration and compatibility issues (3%, n = 2) and licensing and registration issues (6%, n = 5) are less represented at this highest impact level. This pattern suggests that the most critical upgrade-related incidents are linked to failures that directly interrupt system operation or affect core technical dependencies. Component-specific failures are particularly important at this level, indicating that failures in individual modules or components can have wider operational consequences when they affect dependent system functions. The presence of database-related failures and service or application failures also indicates that critical instability is often linked to areas where system availability, data access, or core functionality is directly impaired.

For incidents with an Impact Score of 3, the distribution in Figure 6 shows a different pattern. Service and application failures form the largest category, accounting for 41% (n = 25), followed by component-specific failures at 36% (n = 22), and configuration and compatibility issues at 21% (n = 13). Database-related issues account for only 2% (n = 1), while licensing and registration issues are not included in this impact category. This pattern indicates that moderate-impact incidents are mainly associated with service or application instability and component-level problems. These incidents may not make the system fully unavailable, but they can still reduce usability, cause repeated disruptions, and increase support demand. This is important because moderate incidents may be easier to contain in the short term, but repeated occurrences can still make them operationally significant.

Taken together, the comparison shows that issue categories vary across levels of operational impact. Critical incidents are dominated by component-specific failures, database-related issues, and service or application failures, suggesting a strong link between high-impact disruption and failures affecting core system functionality. Moderate incidents, by contrast, are dominated by service or application failures and component-specific failures, suggesting that recurring functional instability can remain operationally important even when it does not reach a critical failure level. This pattern highlights how the distribution of technical domains varies across levels of operational severity and provides an empirical basis for examining how different types of incidents are interpreted and prioritised in the qualitative analysis.

4.1.2.5 Escalation Behaviour and Resolution Complexity

This section examines two dimensions of resolution complexity: escalation level and average resolution time. Escalation level shows how far an upgrade-related incident moved within the support structure before resolution, while average resolution time indicates how long cases took to close, depending on whether they required technical-fix involvement. Together, these two dimensions provide insight into the organisational effort required to resolve upgrade-related incidents, including whether cases were handled within frontline support or required deeper technical intervention. This is important because resolution complexity is not only reflected in whether a case is escalated, but also in how long the resolution process takes.

1. Escalation Levels for Upgrade-Related Incidents

This dimension examines the organisational effort required to resolve upgrade-related incidents. By identifying the furthest point of escalation, the analysis provides insight into the level of technical involvement required to resolve it. Consistent with previous sections, results are presented for cases where escalation behaviour was explicitly identifiable (n = 499 out of a total of 600), and the remaining 101 cases were classified as “Unknown” due to insufficient evidence and are excluded from this distribution. The results are shown in Figure 7.

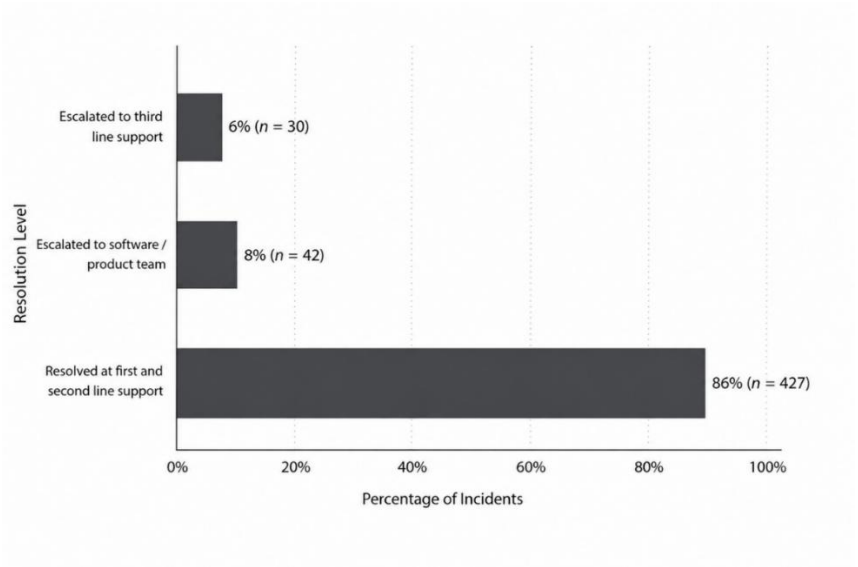


Figure 7: Escalation Levels for Upgrade-Related Incidents (n = 499)

The results are presented in aggregated form to preserve organisational confidentiality. The distribution of escalation levels shows that the majority of incidents were resolved within

frontline support. Specifically, 427 out of 499 cases with identifiable escalation behaviour were handled at first- and second-line support levels, representing 86% of the identifiable escalation dataset. In contrast, 42 of 499 cases (8%) were escalated to the software or product team, while 30 of 499 cases (6%) were escalated to third-line support.

When considered alongside the operational impact results (Section 4.1.2.3), a notable pattern emerges. Among the 396 cases with identifiable impact, 315 cases were classified as Score 4 or 5, indicating major or critical operational impact. At the same time, among the 499 cases with identifiable escalation behaviour, 427 cases were resolved at first- and second-line support. Although these are different subsets of the 600-ticket sample, meaning impact and escalation were not directly cross-tabulated, the aggregate comparison suggests that high operational impact does not automatically correspond to escalation to higher technical tiers. Instead, it indicates that even severe upgrade-related incidents are frequently managed and contained within frontline support processes.

However, the subset of incidents requiring escalation highlights cases where additional expertise or deeper system-level intervention was necessary. These findings provide an empirical basis for examining how support actors manage and coordinate the resolution of complex, high-impact incidents in practice, a further exploration in the qualitative analysis (Section 4.2).

2. Average Resolution Time by Escalation to Technical Fix

This dimension examines the average number of days it took to resolve a ticket, depending on whether it required technical-fix involvement. The results are shown in Figure 8.

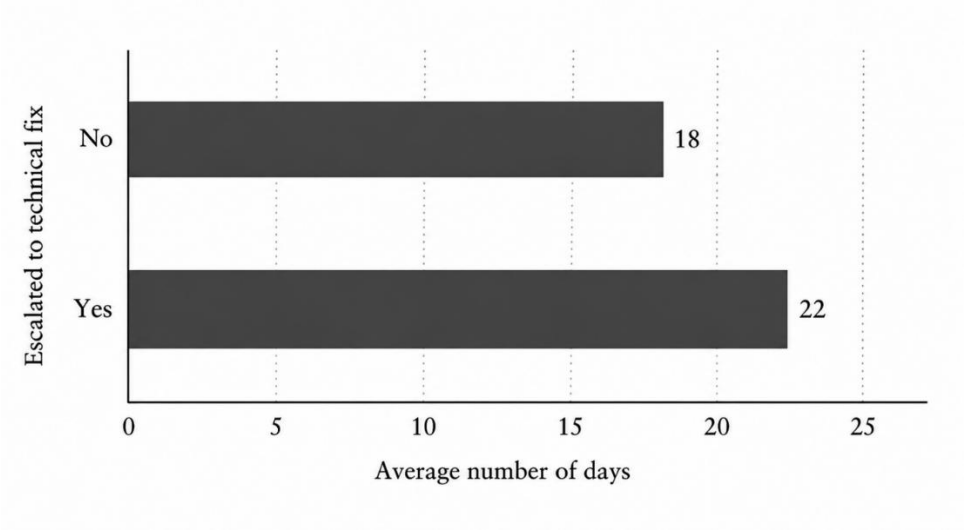


Figure 8: Average Resolution Time by Escalation to Technical Fix

The results show that tickets escalated to a technical fix had a longer average resolution time than those that were not. Cases requiring technical-fix involvement took an average of 22 days to resolve, compared with 18 days for cases that were not escalated. This indicates that technical fix escalation is associated with a longer resolution process.

The four-day difference suggests that cases requiring technical-fix involvement may involve greater diagnostic complexity, deeper system-level investigation, or dependencies beyond

frontline support. However, the difference is understandable given a technical fix; 18 days are quite extensive without one. Indicating that escalation alone does not explain extended resolution times. Non-escalated cases still took an average of 18 days to resolve, suggesting that delays may also occur within earlier stages of support handling, such as diagnosis, information gathering, customer communication, or coordination across support roles.

Overall, the chart indicates that technical fixes are associated with slightly longer resolution times. Still, extended resolution appears to be part of a broader support process rather than solely a consequence of escalation. This supports the need to interpret resolution time in light of issue complexity, available information, and coordination across support levels. These patterns provide a basis for the qualitative analysis that follows, where missing customer information, role handoffs, and cross-role coordination are examined in more detail.

4.1.2.6 Documentation and Formal Support Structures

The previous sections identified recurring patterns in issue categories, operational impact, escalation behaviour, and resolution time. However, the support ticket data did not provide sufficiently consistent evidence to quantify documentation issues in the same way as issue category, impact score, or escalation level. Although the Stage 2 extraction schema included a variable for documentation issues, direct mentions of missing, unclear, or inadequate documentation in the ticket text were too limited to form a reliable quantitative distribution. In this dataset, support tickets typically documented the reported problem, technical actions, and resolution steps, rather than the absence or quality of reference materials.

Because documentation issues could not be treated as a standalone quantitative distribution, this section uses the organisation's Technical Support Handbook as a support-process artefact. Appendix 5 presents the Support Journey Mind Map derived from this handbook. The figure outlines a multi-tiered support structure with formally defined responsibilities and dependencies across customers, system administrators, system integrators, first-, second-, and third-line support, product specialists, and the software team.

The Support Journey Mind Map illustrates that a single actor or team does not handle upgrade-related support. Instead, resolution depends on information moving across several organisational and technical boundaries, including customer-provided information, logs, configuration details, troubleshooting notes, support documentation, escalation records, and product-level feedback. This structure distributes authority, system visibility, and diagnostic responsibility across multiple roles, meaning that no single actor holds complete access to all customer context, technical evidence, and product-level knowledge at the same time.

This formal structure demonstrates why documentation and structured information transfer are central to support for upgrades. If customer information, logs, configuration details, or escalation notes are incomplete, the formal support pathways lack the required inputs to proceed efficiently.

Taken together, the support ticket analysis identifies recurring patterns in upgrade-related incidents, including issue categories, operational impact, escalation behaviour, and resolution time. In contrast, the Support Journey Mind Map shows the support dependencies through which such incidents are expected to be handled. However, these sources do not fully explain why these patterns occur or how support actors manage in practice. The subsequent phase, therefore, examines how support practices, coordination across roles, missing information,

documentation gaps, and decision-making under uncertainty contribute to the observed instability patterns.

4.2 Qualitative Insights from Ecosystem Actors

Semi-structured dyadic interviews were conducted with ecosystem actors directly involved in managing post-adoption software upgrades (see Appendix 3 for the Interview Guide). The dyadic interview setup enabled in-depth exploration of role-specific interactions and cross-role dependencies. These interviews complemented the quantitative analysis of support ticket data to examine how organisational structures, support practices, and coordination mechanisms affect upgrade-related decision-making. Specifically, the interviews examined how misalignments amid technical and organisational interaction, as well as issue severity and recurrence, affect decisions under uncertainty.

Support ticket data reveals patterns of upgrade instability and escalation but provides limited insight into how support practices are implemented. The dyadic interviews clarify how coordination unfolds across roles and how knowledge is generated and circulated within the enterprise IT ecosystem. In addition to how actors make trade-offs when system behaviour is only partially understood. These qualitative findings describe how actors understand and react to technical compatibility misalignment, coordination breakdowns, and information gaps during post-upgrade situations, with particular attention to prioritisation, escalation, and risk trade-offs in operations.

The findings are structured around eight analytical themes: Support practice, Knowledge fragmentation, Coordination breakdowns, Degraded system state, Ecosystem complexity, Governance deadlocks, Decision-making under uncertainty, Customer importance. The categories are derived from abductive thematic coding of the interview data (Appendix 4; Interviews A or B, quotations from interviews are referred to as A.X or B.X), which demonstrates the observed patterns in the support ticket analysis (Chapter 4.1). Table 3 provides an overview of the qualitative insight, theme, representative ID, and the quantitative data that connect to that insight, divided by each of the following chapters.

Table 3: Combining Qualitative and Quantitative Insights Overview

Chapter	Qualitative Insight	Theme(s)	Ids	Quantitative Data
4.2.1	Documentation is used as a first reference, but is insufficient after upgrades.	Support practice	B.2	Post-upgrade incidents: 76% (4.1.2.1)
	Support work involves investigative troubleshooting beyond manuals.	Support practice	A.1, A.2, A.3	Configuration & compatibility issues: 35% (4.1.2.2)
	Informal workarounds serve as substitutes for missing official guidance.	Support practice	B.1, B.4	Component-specific failures: 24% (4.1.2.2)
	Training relies primarily on experiential learning rather than documentation.	Knowledge fragmentation	A.4	Documentation limitations (4.1.2.6)
4.2.2	Upgrade cases move across multiple support levels.	Coordination breakdowns	A.5	Tiered escalation structure (4.1.2.6)
	Context and system knowledge are degraded during handoffs.	Coordination breakdowns	A.6	Average resolution ≈ 20 days (4.1.2.5)

	Troubleshooting actions prior to escalation worsen system states.	Degraded system state	A.7, A.8, A.9	Configuration issues dominate: 35% (4.1.2.2)
	Customer delays and missing logs stall coordination.	Coordination breakdowns	B.5, B.6	Customer non-response: 19% (4.1.1)
	Knowledge is siloed around components and services.	Knowledge fragmentation	B.7	Extended resolution and escalation patterns (4.1.2.5)
4.2.3	Rollback after upgrades is difficult or impossible.	Degraded system state	A.10, A.11	Impact levels 4–5 ≈ 80% (4.1.2.3)
	Component failures propagate across dependent modules.	Ecosystem complexity	A.12	Impact-5 component failures: 42% (4.1.2.4)
	Licensing issues are blocked by unclear ownership.	Governance deadlocks	A.13, A.14	Licensing issues: 27% (4.1.2.2)
	Support lacks the authority to resolve licensing barriers.	Governance deadlocks	A.15, A.16	Prolonged resolution periods (4.1.2.5)
	Configuration variability prevents standardised fixes.	Ecosystem complexity	B.7	Configuration & compatibility dominance (4.1.2.2)
4.2.4	Decisions are made before root causes are fully understood.	Decision-making under uncertainty	B.8	High-impact escalation patterns (4.1.2.3)
	Diagnostic work proceeds through ruling-out logic.	Decision-making under uncertainty	B.9	Partial resolutions are common (4.1.1)
	Experience substitutes for formal system knowledge.	Decision-making under uncertainty	B.10	Frontline resolution: 86% (4.1.2.5)
	Escalation is avoided when coordination delays are anticipated.	Decision-making under uncertainty	B.11	Frontline handling dominance (4.1.2.5)
	Customer operational context shapes urgency.	Customer importance	B.12, B.13	Impact categorisation logic (4.1.2.3)
	Safety-critical contexts override technical equivalence.	Customer importance	B.14	Severity–category mismatch (4.1.2.4)
	Recurring moderate-impact issues drive longer-term coordination.	Decision-making under uncertainty	B.9, B.10	Impact-3 service/application failures: 41% (4.1.2.4)
	Many issues are stabilised rather than fully resolved.	Decision-making under uncertainty	B.11	Resolved tickets: 55% (4.1.1)

4.2.1 Role of Support Practices

To start, interviewees were given a sense of the scale of the problem: 39% of enterprise support activity was directly associated with upgrade instability (4.1.1). Interviewees were then asked to expand on recent issues and why they are difficult to solve. Findings indicate that support practices involve actors consulting official documentation to analyse and resolve upgrade issues, following substantial investigative and compensatory work on their behalf, suggesting that the difficulty has increased. This is especially true during and after a major software upgrade, as reflected in data showing that a large share of issues arise after the upgrade (76%; 4.1.2.1). The first point of reference is often official documentation, such as user manuals, migration guides, and release notes, when an issue arises. However, interviewees

repeatedly described these materials as incomplete, fragmented, or quickly outdated in practice (B.2: “The documentation is correct when it’s released, but not always later.”).

The organisation's official support guidelines state that current documentation ensures consistency (4.1.2.6). Interviewees reflected that documentation is often accurate at the moment of publication but may no longer reflect actual system behaviour when issues arrive on their desk (B.2). This was noted as noticeable after major upgrades, when large changes in system logic, licensing models, and user interfaces led to upgrades that outpaced formal guidelines. Thus, the support staff encountered gaps in documentation. The gap was especially apparent for complex, frequent problems such as component-related failures (24%) and post-upgrade configuration states (35%). Those problems represent the two largest issue categories (4.1.2.1). In extending response, support practices relied heavily on experiential knowledge and informal methods. Interviewees described creating team-specific informal workaround support documents (B.1; B.4: “Sometimes you create your own workaround because nothing else exists yet.”). These methods were used to compensate for gaps in the official manuals. Notably, when visual examples would be very helpful and sometimes required for understanding. Unfortunately, interviewees added that those types of methods were not supported by those responsible for documentation.

The organisation's official support guidelines state that current documentation facilitates training by contributing a “central knowledge base” (4.1.2.6). However, interviewees reported that documents were fragmented and that they relied heavily on experience during training. Both second- and third-line support actors described that training new staff consisted mainly of direct walkthroughs of the system interface and real cases with documentation serving as a secondary reference (A.4: “Most training happens from experience, not manuals.”). To accommodate complex upgrade issues, interviewees estimated that onboarding was highly dependent on expertise. Particularly to fit case-specific examples and to be versatile.

These quantitative findings help illustrate why mitigating the repeated issue categories (4.1.2.1) proved difficult despite the availability of standard documentation. Furthermore, the next section expands the findings, examining how coordination manages the complex support structure.

4.2.2 Coordination Breakdowns and Role Dependencies

The session started by showing interviewees the relevant quantitative findings for matching coordination questions. The quantitative findings consisted of the complex structure example of the support journey mind map (Appendix 5) and the extended resolution period (20 days; 4.1.2.5). The interviews confirm that upgrade-related incidents routinely require coordination across multiple support levels and roles. Delays and breakdowns are emerging largely at escalation boundaries, as shown in the map. The reason given for the extended period was that the issues involved multiple escalations.

Interviewees were shown the distribution of escalation levels (4.1.2.6). Accordingly, interviewees described a tiered support structure in which first- and second-line support work on initial diagnosis, while more complex cases are escalated to third-line support. Furthermore, in some cases, escalation went further to the product specialist or development teams (A.5: “The case moves between different support levels.”). In addition, interviewees said escalation often occurred due to limited visibility, missing information, or unclear system states.

Examples of large dependencies on ecosystem actors were found in the data and interviews. Interviewees emphasised that customer information was required before proceeding with issues, and that was a common factor in delays (customer reported non respondent 19%; 4.1.1). Third-line support actors report unresolved cases as going in circles due to insufficient information. Further examples were mentioned, such as logs or configuration details, leading to back-and-forth communication. Additionally, when interviewees mention issues that need to be escalated to software teams, the information is outdated or hard to decipher due to a long communication line (B.5; B.6: "There's a lot of back and forth before anything moves forward."). Interviewees noted that this time and effort could otherwise have been spent performing technical analysis.

Interview findings further suggest that upgrade-related coordination challenges can be intensified by resolution or troubleshooting steps. The steps involve support staff taking incorrect actions before escalation, which can extend resolution times when issues are escalated. Interviewees described situations in which, before the issue reached them, it had gone through actions which relied on practices that had been effective in earlier software versions but were no longer sufficient for the newest version (A.7; A.8). Upgrades with changes on the software architecture, licensing model, and component relationships (4.1.2.2), previously valid troubleshooting methods in some cases did not work ("Older methods don't apply to the upgraded system.", A.8). Even in several cases, interviewees noted that attempted fixes altered system configuration in ways that the issue was in higher degraded state after support than initially reported by customers (A.9), adding on to the problem of high configuration/compatibility issues (35%; 4.1.2.2). Interviewees attributed it to limited capacity to keep up with technical knowledge amid frequent system changes, even when documentation was available.

Participants reporting issues increased due to the distribution of knowledge across roles, as shown in the Support Journey Mind Map (4.1.2.6). Interviewees described support teams evolving into more specialised experts, siloed around specific components, licensing systems, or cloud services. Therefore, in line with patterns from specific components and licensing issues of the issue categories in the data (4.1.2.2). Even when key individuals were unavailable, knowledge gaps slowed diagnosis and forced teams to revert to exploratory troubleshooting or additional escalations. Hence, interviews explicitly linked these dynamics to a longer resolution period (4.1.2.5). These findings closely align with quantitative patterns showing longer resolution times for issues requiring technical fixes and higher escalation levels (4.1.2.5), and further demonstrate how coordination dependencies in the Support Journey Mind Map (4.1.2.6) contribute to prolonged instability. Furthermore, coordination issues arose when incidents required escalation beyond initial support. In addition, participants noted that escalation often occurred because the available information was insufficient to determine root causes or appropriate fixes, giving rise to the next chapter.

4.2.3 Impact Factor, Issues Details and Ecosystem Complications

Interviewees affirmed that the difficulty of resolving upgrade issues depended on both the technical characteristics of the issue and the ecosystem environment. These findings were apparent when clarification was sought for the distribution of the issue categories (4.1.2.2). They specifically pointed out issue categories, configuration and compatibility misalignment, licensing and registration problems, and component-specific failures.

Impact factor was shown for relevance of severity for upgrade issues (marked 4 or 5;80%; 4.1.2.3). Interviewees noted that limited rollback options to earlier system versions increased the operational impact of upgrade issues. Once an upgrade had been initiated, reverting to a prior stable state was described as difficult and, in some cases, infeasible ("Once you upgrade, it's very hard to go back," A.10; A.11). Hence, even issues that might otherwise have been temporarily mitigated required full diagnosis and resolution within the upgraded environment. The interviewees noted that the absence of rollbacks reduced flexibility in issue handling and added pressure on support teams. Support actors described upgrade issues as even more challenging when failures included multiple linked components, unclear system states, or variance between interface indicators and backend logs (A.12; B.7). Circling back to the core issue of the ecosystem issue categories being especially difficult and complex to handle.

Licensing-related upgrade issues (27%; 4.1.2.2) were usually difficult to resolve when ownership structures were unclear or divided across organisational entities. Interviewees reported cases in which systems were technically operational yet blocked due to uncertainty about license ownership, organisational responsibility, or historical configuration decisions made before the upgrade ("Sometimes it's unclear who owns the license or organisation.", A.13). In such situations, support teams lacked the authority or visibility to act (A.15; A.16). Consequently, prolonging resolution period (4.1.2.5) despite there being no technical problem.

The findings indicate that the ecosystem complicates the resolution of the issue. Interviewees regularly cited common subcategories of configuration and compatibility issues (4.1.2.2), including variability across customer environments, such as differences in component combinations, third-party integrations, network architectures, and regional usage patterns. An unfortunate reality is that similar outcomes arise from different underlying causes, limiting the applicability of known fixes and diagnostic knowledge across cases. These accounts help contextualise why certain issue categories dominate the dataset and why fixes are not easily standardised despite obvious recurrences. They also shed light on how actors reach decisions when easy fixes are nowhere to be found.

4.2.4 Decision-Making Under Uncertainty

Decision-making during upgrade-related incidents was described as inherently uncertain and immensely responsive. Responsive to operational pressure or the impact factor shown in the quantitative data (4.1.2.3). Interviewees frequently reported making decisions before fully comprehending the root causes, especially when systems were partially or fully unavailable (B.8: "You often act before fully understanding the root cause.").

Interviewees were asked whether they spotted recurring issue categories (4.1.2.2) with non-high-impact incidents (impact levels 1–3; 4.1.2.3). They answered that individual cases may not trigger immediate escalation, but the issues become difficult to ignore when they recur at scale. Quantitative findings show that for impact level 3 issues, the dominant issue category is service and application instability or failure, accounting for 41% of cases (4.1.2.4). For clarification, interviewees said these issues were manageable individually but became disruptive when they recurred. Additionally, they noted that recurrent issues were prioritised if they involved frequent customer contact or broader system issues. In such situations, interviewees noted that decision-making shifted from incident-level resolution to longer-term decisions. Furthermore, interviewees explained that repeated occurrences of similar issues prompted broader discussions on coordination. Those discussions included potential platform changes,

documentation upgrades, or changes in support guidance. In some cases, the volume of customer calls would trigger operational decisions, such as temporarily relaxing system constraints or reducing customer impact. Thereby, illustrating how issue frequency at moderate impact levels served as a trigger for lessons learned, ultimately leading to technical escalation.

Impact factor emerged as the primary driver of immediate decision-making. High-impact incidents, including category 4 or 5 incidents, account for 80% of the combined share. The categories involved system downtime, safety-critical environments, or standards-based requirements. They were prioritised quickly and often addressed before other ongoing work. Interviewees described reprioritising tasks and escalating issues based on apparent urgency rather than diagnostic completeness. The highest issue category, with an impact factor of 5, was component-specific, accounting for 42% of these cases (4.1.2.4). Interviewees described such incidents as especially high-impact because failures at the component level often propagated across dependent modules. Furthermore, adding that the incidents are obscuring system behaviour and complicating recovery ("There are many components involved that can fail," A.12; "by the time it reaches us, the system state is worse," A.9). These accounts illustrate how component-specific failures escalate quickly into critical operational disruptions. Therefore, triggering urgent decision-making under uncertainty.

Decision-making was also influenced by time, patterns from the resolution period (4.1.2.5), and organisational constraints, as outlined in the Support Journey Mind Map (Appendix 5). Interviewees noted that escalations to global distributed teams often introduce delays. Cross-regional handoffs can add several days to resolution time. To avoid these delays, support actors frequently tried to resolve unclear cases locally. Furthermore, the data show that the majority of cases (86%; 4.1.2.5) were solved at the frontline, even when available information was incomplete (B.11).

Interviewees explained that the operational context of the affected customer strongly pressured prioritisation. Thereby, deepening the findings of what controls decision-making, other than the frequency of categories (4.1.2.2) and severity of the cases (4.1.2.3). Issues impacting systems deployed in safety- or security-critical environments were treated with greater urgency (B.12; B.13). Interviewees mentioned that a similar technical failure could prompt different escalation and handling strategies depending on the potential practical consequences of system unavailability (B.14). This situational prioritisation influenced decision-making even when diagnostic uncertainty remained high.

Taken together, these findings align with the quantitative evidence showing a strong relationship among operational impact, escalation behaviour, and resolution time. They further indicate that decision-making during upgrade instability is guided by pragmatic trade-offs as opposed to deterministic rules, a pattern reinforced by the fact that only 55% of upgrade-related tickets were marked as resolved (4.1.1), suggesting that many issues were stabilised operationally. Building on the qualitative analysis, the next section brings together the findings to further develop the overall interpretation of the results.

4.3 Integrated Meta-Inferences

This section presents a set of integrated meta-inferences that combine quantitative with qualitative accounts. Showing how incident patterns identified through the data analysis can be

explained through organisational practices. Thus, incident data highlight recurring operational disruptions, and the organisational practices perspective explains why these disruptions continue. By synthesising findings from Chapters 4.1 and 4.2, the meta-inferences move beyond single-method results to identify cross-cutting mechanisms through which technical conditions and support practices interact. These meta-inferences, which integrate quantitative patterns with qualitative explanations, are represented in Table 4.

Table 4: Meta-Inference Theme Table

#	Meta-Inference Theme	Quantitative Basis	Qualitative Basis
1	Configuration and Compatibility Complexity Dominates Upgrade Instability	Configuration and compatibility issues constitute the largest category of upgrade-related support tickets	Multi-component interactions, hidden dependencies, and environment-specific behaviour obscure fault isolation and increase diagnostic effort
2	Support Practices Limit Transparency of Upgrade Instability	Post-deployment issue registration; delayed detection; extended resolution period; low proportion of cases escalating to technical fixes	Reliance on informal workarounds and experiential knowledge; limited visibility into system behaviour and root causes
3	Decision-Making Is Shaped by Coordination Breakdowns and Knowledge Fragmentation in the Support System	Support Journey Mind Map illustrating multi-tiered responsibilities, dependencies, and escalation paths across support roles	Iterative, experience-based decision-making under limited system visibility; reliance on escalation, ruling-out practices, and fragmented, person-specific knowledge
4	Upgrade Instability Sustains Degraded System States and Governance Deadlock	High operational impact; extended resolution periods; the majority of cases handled at frontline support levels	Troubleshooting actions compound system degradation; rollback is constrained by licensing; unclear authority and fragmented ownership limit permanent corrective action

4.3.1 Meta-inference 1: Configuration and Compatibility Complexity Dominates Upgrade-Instability

The findings indicate that configuration complexity and compatibility predominantly drive upgrade instability. Quantitative analysis shows that configuration and compatibility issues constitute the largest category of upgrade issues. Qualitative findings explain this prevalence by highlighting the multiplicity of interacting components, environmental dependencies, and version mismatches involved in post-upgrade configurations.

Interviewees described difficulty in identifying failure points when components appeared operational in isolation but failed when integrated. Meaning that failures propagated across dependent modules. Also, backend logs and interface states are inconsistent. These issues were often environment-specific and insufficiently documented, limiting predictability and increasing diagnostic effort. The upgrade changed architecture, licensing logic, and component relations. These findings support the dominance of dependency complexity and component failures in the propagation of cascading modules. Leading to the majority of failures occurring post-upgrade and with a high concentration of high-impact failures, especially when rollback options are limited.

Together, the findings suggest that component failures and complex configuration interactions increase diagnostic ambiguity, coordination dependency, escalation uncertainty, and resolution duration, thereby contributing to upgrade instability. The instability exceeds current support visibility and tooling, leading to the next chapter.

4.3.2 Meta-Inference 2: Transparency of Instability Emerges Through Support Practices

The findings indicate that upgrade-related instability does not become immediately visible through technical monitoring alone but instead emerges progressively through support practices. The issues do not have a straightforward solution, as only half of the tickets were solved during the time of the analysis. The quality of the data reported to the support team on the issue is lacking, as shown by both qualitative and quantitative accounts. The data shows that 20% of the time, the customers fail to respond to the support team for necessary information. Even though the impact factor is leveraged to escalate problems, as stated in the interviews, the impact factor of the issues is not clearly stated in one-third of the issues. Furthermore, researchers had difficulty determining whether documentation was lacking for issues during data analysis, indicating a lack of structure for identifying important missing documentation.

Interviewees mentioned that the documentation available became straight away outdated after upgrades, as it couldn´t keep up with post-adoption issues. Hence, quantitative patterns show that many instability incidents are recorded only after system deployment, often with delayed detection, and that the majority are resolved by frontline support and have extended resolution periods. These quantitative patterns align with qualitative accounts describing how instability surfaces through user complaints, support tickets, and escalation workflows rather than through proactive detection mechanisms. Leading to informal workarounds replacing documents and official guidance. As well as experiential troubleshooting preferred over manuals. This results in training that is largely reliant on shadowing experts and live cases, as well as outdated onboarding material.

Interview data suggest that interviewees frequently recognised instability retrospectively, as users report degraded performance, missing functionality, or unexpected behaviour during regular operation. Through prioritisation, categorisation, and escalation processes, these signals are gradually transferred into a shared understanding of instability. This limits early detection and prevention. Instability becomes transparent as an accumulation of symptoms translated through organisational routines and coordination.

4.3.3 Meta-Inference 3: Decision Making Is Shaped by Coordination Breakdowns and Knowledge Fragmentation in the Support System

The findings indicate that decision-making during upgrade issues is moulded by structurally embedded uncertainty and fragmented knowledge across the support organisation. The Support Journey Mind Map (Appendix 5) shows a structure that distributes authority, system visibility, and diagnostic responsibility across multiple roles. Qualitative findings align closely with this structural configuration. Interviewees described routinely making decisions without a clear understanding of root causes, emphasising practices such as ruling out possibilities, drawing on experience, and escalating cases when required information or expertise was unavailable. Contrary to following linear diagnostic pathways, decision-making was described as iterative and provisional, modulated by delayed feedback, partial system data, and dependence on other roles.

At the same time, coordination across this structure is complicated by knowledge fragmentation. While the handbook presents documentation as a mechanism to ensure

consistency, facilitate training, and provide a centralised knowledge base, interviewees reported that critical system understanding is dispersed across individuals, specialised teams, and informal communication channels. Expertise related to specific components, configurations, or upgrade behaviours was often siloed, creating reliance on particular individuals and increasing vulnerability to handover gaps when cases moved between support levels.

Together, these findings show that decision-making challenges are not solely the result of complex technologies, as only a very small percentage of them require technical fixes from the software team. As well as decision-making not being solely due to insufficient information, it also arises from the interaction between structural dependencies and fragmented knowledge, which is then compounded with every handoff as information context is lost. When the support team has a hard time making a decision with limited knowledge, they are more likely to make mistakes, causing a further degradation of the system state.

4.3.4 Meta-Inference 4: Upgrade Instability sustaining Degraded State and Governance deadlock

The findings indicate that upgrade instability is sustained through the interaction of degraded system states and fragmented governance arrangements. Quantitative analysis shows that a substantial proportion of upgrade-related incidents are high-impact and have extended resolution times, with many cases remaining unresolved at frontline support levels. These patterns suggest that instability persists beyond initial remediation efforts and shifts the focus away from discrete failure events.

Qualitative findings explain this persistence by revealing how post-upgrade systems enter degraded states that are difficult to reverse. Interviewees described how rollback options are constrained after deployment due to technical dependencies and licensing arrangements, making reversion contractually restricted or operationally impractical. Therefore, support teams rely on successive troubleshooting steps and incremental fixes that modify system configurations to restore partial functionality. Over time, these interventions compound, reinforcing degraded operating conditions within the upgraded system state. At the same time, the fragmentation of governance and responsibility constrains decisive intervention. Interviewees reported dispersed ownership across development, operations, support, and vendor organisations, often without clear authority to implement permanent corrective actions. In the absence of clear accountability, responses focused on maintaining service continuity through temporary stabilisation measures rather than resolving underlying causes.

Together, these findings indicate that upgrade instability is not an individual failure event but an evolving condition sustained by governance deadlock and technical irreversibility. Fragmented responsibility limits corrective action, while constrained rollback and incremental interventions entrench degraded system states, enabling instability to persist following upgrades.

5 Discussion

In this chapter, the findings from the quantitative incident analysis and qualitative interviews, as well as the meta-inferences, are discussed in relation to the literature, theory and practice. The results are analysed to interpret the nature of upgrade instability and to deduce conclusions about the sociotechnical factors that shape post-adoption operational outcomes.

5.1 Sociotechnical Overview

The discussion draws in particular on coordination theory and sociotechnical perspectives, while also reflecting insights from digital infrastructure and complexity theory that emphasise incremental change, accumulated dependencies, and constrained reversibility (Benbya & McKelvey, 2006; Drechsler et al. 2022; Malone & Crowston, 1994; Yoo et al. 2012). Table 5 summarises the dual conceptualisation of upgrade instability adopted in the discussions. It indicates where each perspective is emphasised in the thesis and supported empirically.

Table 5: Overview: Connection Between Findings and Supporting Literature

Sub-Chapter	Empirical Anchors (Chapter 4)	Key Supporting Literature
5.1 Sociotechnical Overview	Meta-Inference 1 (configuration & compatibility complexity); Meta-Inference 2 (transparency through support practices); interaction between technical and organisational conditions (Figure 9)	**Core IS + Sociotechnical Theory:** (Baxter & Sommerville, 2011; Sarker et al. 2019) **Post-Adoption / IS Success:** (Bhattacherjee, 2001; DeLone & McLean, 2003; Jasperson et al. 2005; Techakriengkrai et al. 2021) **Ecosystem + Complexity:** (Hanseth & Lyytinen, 2010; Kapoor et al. 2021; Tiwana et al. 2010) **Upgrade Instability / Technical:** (Dumitraş & Narasimhan, 2009; Sommerville et al. 2012; Zhang et al. 2021)
5.2 Process: How Instability Is Sustained Over Time	All four meta-inferences (MI1–MI4), reinforcing cycle (Figure 11–12); configuration complexity, limited visibility, knowledge fragmentation, escalation, decision-making under uncertainty, and degraded states	**Ecosystem + Complexity:** (Hanseth & Lyytinen, 2010; Henfridsson & Bygstad, 2013; Jacobides et al. 2018; Yoo et al. 2012) **Coordination / Uncertainty / Support:** (Collopy et al. 2020; Costa et al. 2024; Figalist et al. 2019; Kotlarsky et al. 2015, 2019; Malone & Crowston, 1994; Ren et al. 2008; Rydén Sonesson et al. 2021; Saxena & McDonagh, 2022) **Upgrade Instability / Technical:** (Adel et al. 2025; Dumitraş & Narasimhan, 2009; Zhang et al. 2021)
5.3 Upgrade Instability as a Sociotechnical Outcome	Meta-Inference 1 (configuration & compatibility); Meta-Inference 2 (support visibility); quantitative outcomes (post-upgrade concentration, high impact, issue distributions)	**Post-Adoption / IS Success:** (Bhattacherjee, 2001; DeLone & McLean, 2003; Jasperson et al. 2005; Techakriengkrai et al. 2021) **Upgrade Instability / Technical:** (Dumitraş & Narasimhan, 2009; Sommerville et al. 2012; Zhang et al. 2021) **Ecosystem + Complexity:** (Hanseth & Lyytinen, 2010; Kapoor et al. 2021; Tiwana et al. 2010) **Coordination / Support:** (Kotlarsky et al. 2019)
5.4 Implications for Practice	Findings on degraded system states, coordination breakdowns, documentation gaps, escalation dynamics, and decision-making under uncertainty	**Coordination / Uncertainty / Support:** (Adel et al. 2025; Costa et al. 2024; Figalist et al. 2019; Kotlarsky et al. 2019) **Post-Adoption / Support:** (Feldman et al. 2017; Retana et al. 2018) **Upgrade Instability / Technical:** (Sommerville et al. 2012)
5.5 Generalisability of Findings	Case characteristics: enterprise IT ecosystem, post-adoption upgrades, distributed governance, recurring instability patterns	**Ecosystem + Complexity:** (Hanseth & Lyytinen, 2010; Jacobides et al. 2018; Kapoor et al. 2021; Yoo et al. 2010) **Coordination / Support:** (Kotlarsky et al. 2019) **Case Study / Generalisation:** (Tsang, 2014; Yin, 2018)

The findings show that upgrade instability develops within broader ecosystem, organisational, and technical conditions. Prior literature has highlighted how modular layered architectures introduce distributed complexity and interdependencies that give rise to instability mechanisms, which are strongly reflected in this study's empirical findings (Henfridsson & Bygstad, 2013; Yoo et al. 2010). In the architecture, the components are loosely coupled yet remain interdependent, leading to distributed complexity across both technical layers and organisational actors. (Henfridsson & Bygstad, 2013; Yoo et al. 2010). Heterogeneous customer environments, distributed coordination structures, fragmented responsibilities, and configuration dependencies further amplify these conditions, creating situations where instability becomes difficult to fully diagnose, coordinate, and resolve (Hanseth & Lyytinen, 2010; Khoo & Robey, 2007; Techakriengkrai et al. 2021; Yoo et al. 2012; Zhang et al. 2021). Furthermore, the findings reinforce the notion that these instability mechanisms are highly context-dependent in heterogeneous environments. Showing that variations in configurations, dependencies, and organisational arrangements shape how issues emerge and evolve (Anthony, 2016; Knoche & Hasselbring, 2021).

Drawing on digital infrastructure and complexity theory, small changes can trigger cascading, unpredictable effects across the ecosystem. These incremental and continuous changes give rise to instability and reinforce persistent processes through which that instability continues to emerge, circulate, and evolve over time (Benbya & McKelvey, 2006; Vert & Sharpanskykh, 2026; Yoo et al. 2012). The meta-inferences (MI, Section 4.3) form an inner connection among themselves, depicted in Section 5.2. They form a process perspective on how instability arises, persists, and is managed through coordination practices, decision-making under uncertainty, and governance constraints (Bhattacherjee, 2001; Jasperson et al. 2005; Tamanna et al. 2025; Zhang et al. 2021).

In contrast, operational outcomes observed following software upgrades, highlighted in section 5.3, include downtime and increased support incidents. An examination of the operational consequences reveals that the interaction of technical, organisational, and ecosystem-level mechanisms produces retained outcomes, including degraded system states, extended resolution times, recurring upgrade incidents, and reactive escalation cycles (Dumitraş & Narasimhan, 2009; Kotlarsky et al. 2019; Zhang et al. 2021).

Figure 9 represents an integrated explanation of upgrade instability derived from multiple empirical layers. The purpose of the figure is to present the sociotechnical factors as conditions, interactions, and outcomes, and to show the collective scope of the thesis. The left column of Figure 9 establishes pre-existing conditions (the Antecedents) in which these sociotechnical factors emerge. The Triangle symbolises the sociotechnical process from the technical, ecosystem, and organisational scope. The meta-inferences and the qualitative themes (QLT, Section 4.2) are visible in the triangle. Finally, the right column shows the operational consequences of upgrading, largely supported by the quantitative data (QNT, Section 4.1).

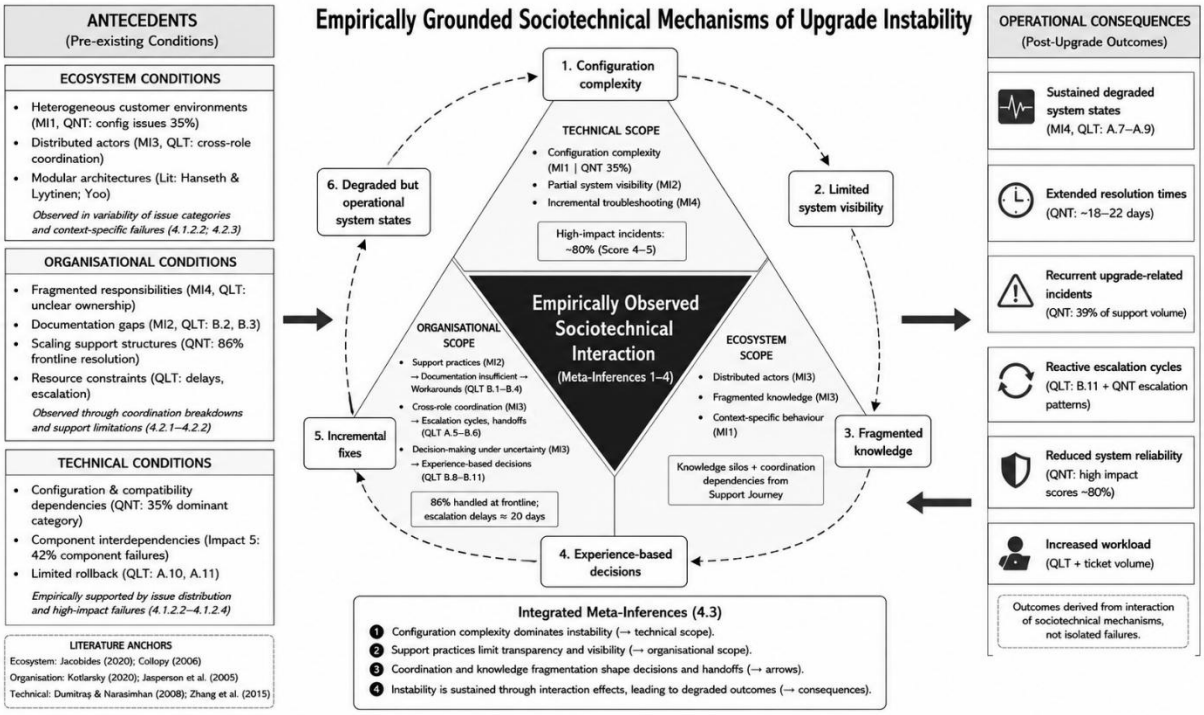


Figure 9: Empirically Grounded Sociotechnical Mechanisms of Upgrade Instability

Together, these elements form an empirically grounded sociotechnical mechanisms that link the measurable operational outcomes to the underlying processes that continuously generate instability (Bhattacherjee, 2001; Jasperson et al. 2005; Tamanna et al. 2025; Zhang et al. 2021), leading to the next chapter.

5.2 Process of Instability Over Time

Previous research on software upgrades has mainly focused on discrete failure events and their resolution (Dumitraş & Narasimhan, 2009; Khoo & Robey, 2007; Tamanna et al. 2025; Zhang et al. 2021). The present findings challenge this view by showing that post-adoption upgrade instability can persist despite substantial corrective efforts (Adel et al. 2025; Costa et al. 2024; Khoo & Robey, 2007; Kotlarsky et al. 2019).

The findings show that post-adoption upgrade instability is sustained through multiple interactions of sociotechnical mechanisms (Jacobides et al. 2018). For example, the findings across studies demonstrate that instability does not automatically decrease after attempts at resolution. Sequential troubleshooting, local workarounds, and configuration upgrades restore partial functionality while altering system states, increasing irreversibility and future complexity (Dumitraş & Narasimhan, 2009; Jabin et al. 2024; Zhang et al. 2021). The meta-inferences showcase how the interactions connect and form a continuous cycle of persistent upgrade instability (see Figure 10).

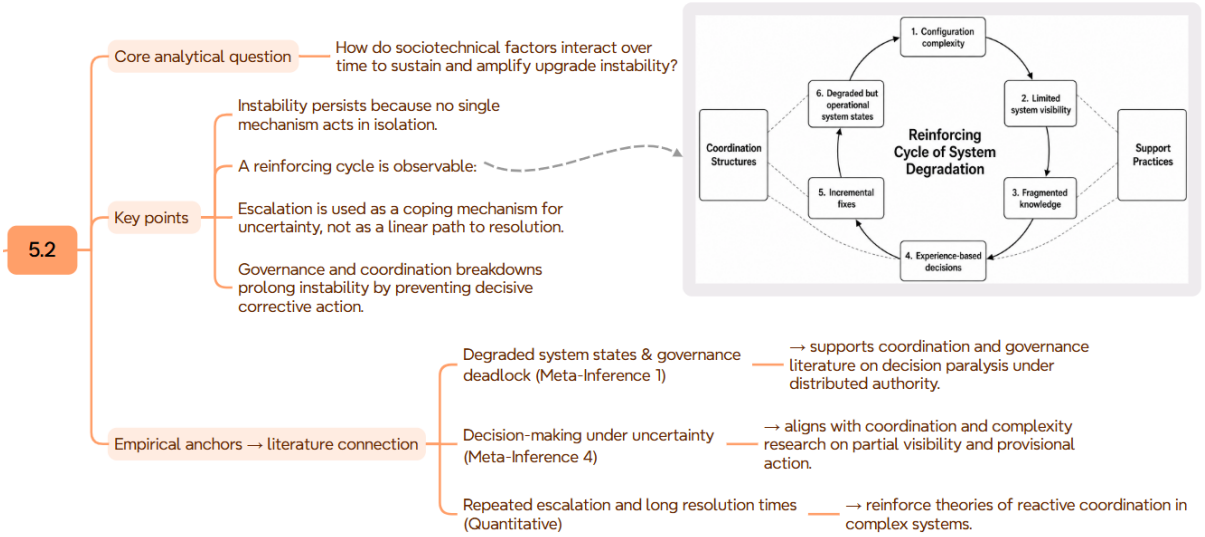


Figure 10: Long-Term Upgrade-Instability Over Time Mind Map

5.2.1 Reinforcing Cycle of Instability

The findings and prior research indicate that actors can't anticipate complex problems due to configuration complexity. Upgrade instability is primarily a problem of interdependence rather than individual component failure. Meta-Inference 1 on configuration and compatibility complexity (Section 4.3.1) appears at the top of the cycle in Figure 11, where configuration complexity and hidden dependencies are the primary triggers of post-upgrade instability. This thereby limits visibility on the system state and directly affects knowledge across support roles. Hence, support actors operate with partial insight into dependencies and historical configuration decisions (Dumitraş & Narasimhan, 2009; Jacobides et al. 2024; Jakobsen & Vanhaverbeke, 2025; Yoo et al. 2012).

At the same time, documentation is often incomplete or lags behind the pace of system development (Figalist et al. 2019). Accordingly, no sole actor possesses a complete understanding of the problem, as responsibilities and expertise are scattered across roles and organisational boundaries (Ghazawneh & Henfridsson, 2013). Coordination theory suggests that interdependent activities require mechanisms such as communication, shared understanding, and structured processes to ensure alignment (Collopy et al. 2020; Kapoor et al. 2021; Malone & Crowston, 1994). However, the observed fragmentation undermines these mechanisms, suppressing coordination and making shared understanding difficult. Thus, errors occur at a higher rate during high-impact incidents when time pressure is high (Khoo & Robey, 2007). Pointing to Meta-Inference 2 on transparency through support practices (Section 4.3.2), as shown in Figure 11, reveals a shift from limited system visibility to fragmented operational knowledge, as reliance on support tools, ticket-based diagnostics, and role-specific perspectives restricts an extensive understanding of the system state.

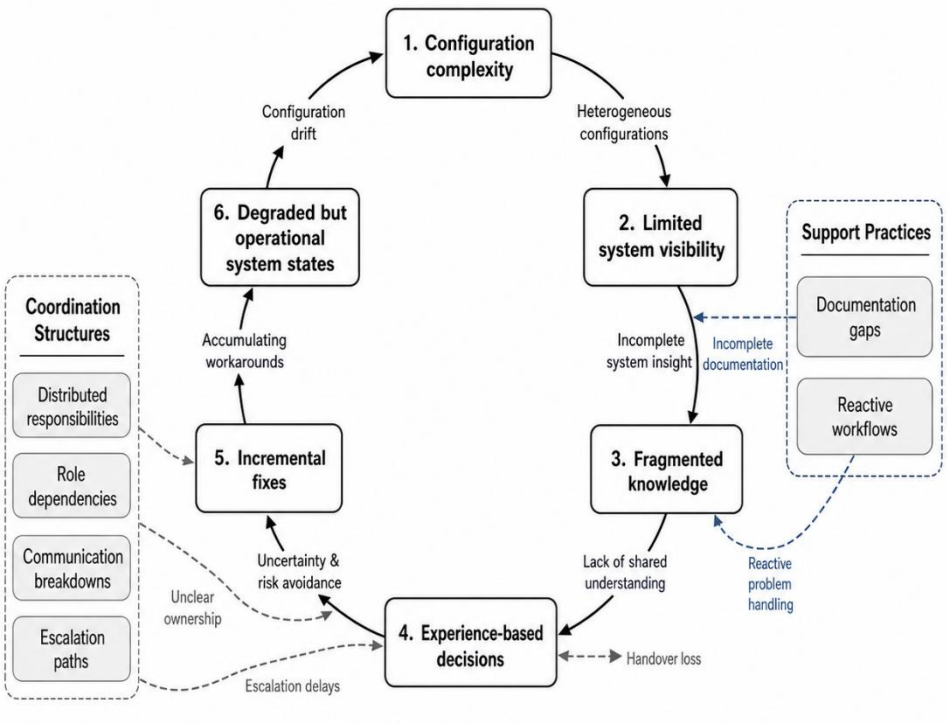


Figure 11: Reinforcing Cycle of System Degradation

The results indicate that, under this knowledge fragmentation, decision-making becomes calculative and experience-based (Jacobides et al. 2018; Saxena & McDonagh, 2022). Represented in Meta-Inference 3 on decision-making, shaping coordination breakdowns, and knowledge fragmentation in the Support System (Section 4.3.3). Figure 11 demonstrates how fragmented knowledge and coordination failures lead to experience-based decision-making and delayed or repeated escalation. Clarifying, findings show that actors often act before root causes are completely understood. Therefore, relying on prior experience, excluding practices, and using exploratory troubleshooting approaches instead of definitive diagnoses. This pattern fits past research on decision-making under uncertainty in complex systems. Thus, actors need to balance incomplete information against operational urgency. Thus, uncertainty is often created by the structure of the ecosystem itself, not only by missing information. Accordingly, more information or escalation does not always remove uncertainty (Collopy et al. 2020; Kotlarsky et al. 2019). Resulting in decisions being decreasingly formed by verified system knowledge and more by situational intuition and perceived risk by support actors (Rydén Sonesson et al. 2021).

According to the findings, escalation plays a critical role in this process but also serves as a coping mechanism for uncertainty. Hence, it allows actors to transfer cases when information, authority, or expertise is lacking. In enterprise ecosystems, uncertainty comes not only from limited information, but also from unclear responsibilities, authority, and ownership across actors (Jacobides et al. 2018; Rydén Sonesson et al. 2021). The quantitative findings show long resolution times. Additionally, they also reflect a multi-tiered support structure with distributed authority. These findings support prior research that describes escalation as a reactive coordination response under conditions of limited visibility and distributed control (Costa et al. 2024; Jacobides et al. 2024; Kotlarsky et al. 2019; Ren et al. 2008). Thereby, repeated escalation introduces additional handovers and loss of context, leading to fragmented knowledge and delays in decision-making (Figalist et al. 2019).

The observations reveal governance and coordination breakdowns that, in turn, supplement this cycle by limiting the ability to implement permanent fixes. In depth, governance structures influence decision-making by shaping what actors know, what authority they have, and when they can act (Jacobides et al. 2018; Rydén Sonesson et al. 2021). Decision-making under uncertainty should therefore be understood as a result of how sociotechnical systems are organised, not only as an individual limitation or lack of competence (Jacobides et al. 2018; Rydén Sonesson et al. 2021).

The findings showed that degraded system states are preserved by fragmented authority and unclear ownership. Notably, singling out licensing and upgrade governance context. Hence, when no single actor holds both the authority and the knowledge required to enact system changes, the decision defaults (Adel et al. 2025; Jacobides et al. 2018; Rydén Sonesson et al. 2021). This governance deadlock enhances reliance on provisional decisions, workarounds and incremental fixes (Kotlarsky et al. 2019; Zhang et al. 2021). As illustrated in Figure 11 by Meta-Inference 4, on upgrade instability, sustained degraded system states, and governance deadlock (4.3.4). Therewith, the issues increase configuration complexity and further restrict the system's ability to achieve stable recovery in future upgrade cycles (Anthony, 2016; Dumitraş & Narasimhan, 2009; Jakobsen & Vanhaverbeke, 2025; Sommerville et al. 2012; Zhang et al. 2021). As shown in Figure 11, the resulting complexity further reduces system visibility, thereby closing the reinforcing cycle (Hanseth & Lyytinen, 2010; Jacobides et al. 2018; Yoo et al. 2012), allowing instability to persist over time.

Collectively, these contributions advance the IS research by integrating post-adoption research, sociotechnical systems theory, and ecosystem perspectives into a process-oriented explanation of upgrade instability (Mumford, 2006; Sarker et al. 2019). Furthermore, the next section will interpret upgrade instability as an operational outcome observed after software upgrades.

5.3 Upgrade Instability as a Sociotechnical Outcome

Building mainly on Meta-Inference 1 and Meta-Inference 2, the section explains how technical complexity contributes to instability, while organisational practices shape how that instability becomes visible, prioritised, and acted upon. As summarised in Figure 12, upgrade instability arises from the interaction among technical complexity, support practices, coordination structures, and decision-making routines. This framing connects the empirical findings to prior research on sociotechnical systems, post-adoption support, enterprise IT ecosystems, and IS success, particularly in relation to system and service quality (Baxter & Sommerville, 2011; DeLone & McLean, 2003; Jasperson et al. 2005; Kapoor et al. 2021).

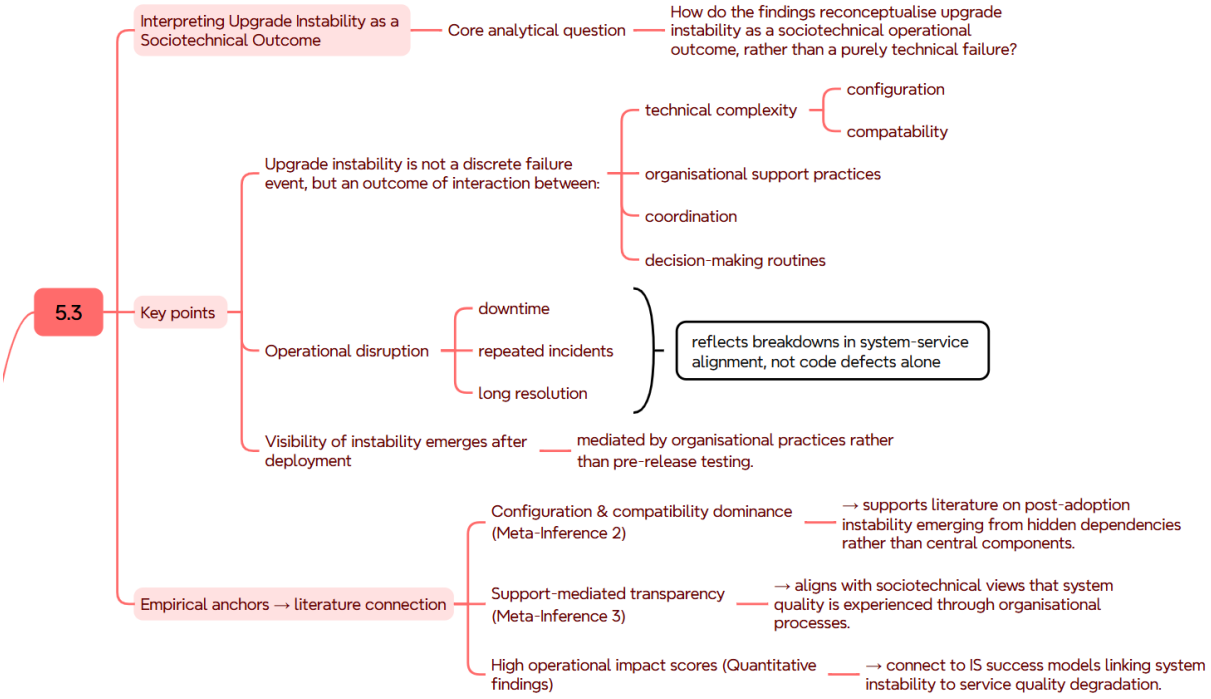


Figure 12: Sociotechnical Upgrade Instability Mind Map

5.3.1 Upgrade Instability as More than Technical Failure

The findings challenge a purely technical explanation of upgrade failure. Although many incidents were triggered by technical changes, such as version updates, configuration changes, licensing dependencies, and component interactions, their operational consequences were shaped by how the organisation detected, interpreted, and responded to them. This interaction between technical change and organisational response aligns with sociotechnical systems theory, which argues that system outcomes emerge through the interaction between technical and organisational conditions (Baxter & Sommerville, 2011; Sarker et al. 2019; Techakriengkrai et al. 2021).

The quantitative findings support this interpretation. Post-upgrade incidents accounted for 76% of classified upgrade-phase cases, indicating that instability was most evident after the upgrade entered the operational environment. The data suggest that many upgrade failures were difficult to predict before full deployment and became apparent when the upgraded system interacted with customer-specific configurations, licensing arrangements, component dependencies, and real-world operating conditions. Prior research reflects the issue that upgrade failures can arise from hidden dependencies and cross-version interactions in distributed systems, with many failures becoming visible only when systems are upgraded and used in operational settings (Desouza et al. 2025; Dumitraş & Narasimhan, 2009; Zhang et al. 2021).

5.3.2 Configuration and Compatibility Complexity

Configuration and compatibility complexity were the clearest technical drivers of upgrade instability in the support data. As established by the issue-category distribution in Figure 3 and further integrated into Meta-Inference 1, instability was concentrated around versions,

configurations, licenses, components, and customer-specific environments. This pattern suggests that configuration and compatibility complexity should be understood as an integration-related challenge. In this study, upgrade outcomes depended on how technical dependencies, components, licences, and customer-specific settings fitted together in practice (Hanseth & Lyytinen, 2010; Kapoor et al. 2021).

The concentration of incidents around versions, configurations, licenses, components, and customer-specific environments reflects the character of enterprise IT ecosystems, where layered architectures, heterogeneous configurations, and distributed actors make system behaviour difficult to predict (Hanseth & Lyytinen, 2010; Kapoor et al. 2021; Tiwana et al. 2010). Recent literature reinforces the point that heterogeneity, interoperability demands, and multiple forms of complexity shape enterprise systems. Sunyaev et al. (2023) argue that demands for heterogeneity and interoperability increasingly shape enterprise information systems. Haraldsson and Staron (2025) add that both internal complexity, such as variants, and external complexity, such as organisational coordination, influence the effort needed to maintain complex software systems. In this study, these ideas help explain why upgrade instability appeared as an integration-related challenge between technical components, local configurations, and organisational roles.

5.3.3 Support Practices, Visibility and Operational Outcomes

Upgrade instability did not become transparent automatically through the technical system itself. The findings show that instability became visible mainly through support practices, including support tickets, customer reports, documentation, troubleshooting routines, escalation paths, and informal knowledge sharing. Technical monitoring and system-level evidence remain important for understanding complex IT system behaviour, especially where system interactions are difficult to predict (Sommerville et al. 2012). Upgrade-specific testing and technical detection methods can also help identify upgrade failures and cross-version incompatibilities in distributed systems (Zhang et al. 2021). However, the findings of this study suggest that such technical evidence was not, in itself, sufficient to make upgrade instability transparent in practice. The central role of support practices aligns with Meta-Inference 2, which shows that transparency of upgrade instability emerged through support practices rather than through technical monitoring alone.

The organisational production of visibility extends post-adoption research by showing how support practices shape system understanding after deployment. Post-adoption research shows that system value depends on continued use, adaptation, and support after implementation (Bhattacherjee, 2001; Jasperson et al. 2005; Retana et al. 2018). Qualitative findings showed that documentation was often the first point of reference. However, it was not always sufficient when real customer environments revealed unexpected upgrade behaviour. This also reflects a temporal misalignment between the system's evolution and its support documentation. A documentation gap in which documentation is accurate at release but quickly becomes outdated as systems continue to evolve through upgrades (Uludağ et al. 2019). Support actors, therefore, relied on logs, experience, workarounds, local guides, and colleagues' knowledge, as mentioned in the previous section. Consequently, support practices acted as sociotechnical filters, shaping which problems became visible, how they were understood, and when they were treated as urgent. This aligns with research on IT support and coordination, where fragmented knowledge and incomplete information flows can make problem-solving dependent

on escalation, experience, and informal knowledge sharing (Costa & Fontão, 2024; Kotlarsky et al. 2015; Kotlarsky et al. 2019).

The quantitative results show that visibility through support practices was closely linked to operational severity. Among cases with an identifiable impact, 80% were classified as major or critical (Adel et al., 2025; Zhang et al., 2021). At the highest severity level, component-specific failures accounted for 42% of impact score 5 cases, indicating that localised technical failures can cause serious operational disruptions. The substantial proportion of tickets with 'Unknown' impact scores (204 out of 600) is not merely a data limitation but a sociotechnical finding indicating that support actors often fail to clearly document the operational consequences of issues, limiting the ability of analytical systems to interpret and prioritise severity (Costa et al. 2024; Kotlarsky et al. 2019).

Although many incidents had a major or critical impact, 86% of identifiable escalation cases were resolved within first- and second-line support, positioning frontline support as a central layer for interpreting and containing instability (Figalist et al. 2019). The concentration of resolutions at frontline support indicates that operational stability is largely maintained through decentralised, experience-based practices rather than formal escalation structures. However, cases requiring technical-fix involvement took longer to resolve, averaging 22 days compared with 18 days for cases without technical-fix involvement. This suggests that when instability could not be contained at the front line, resolution became increasingly dependent on broader coordination across knowledge boundaries (Figalist et al. 2019; Rydén Sonesson et al. 2021).

The findings show that partial visibility also shaped decision-making across the support organisation. The fact that 19% of upgrade tickets were closed due to a lack of customer response further highlights coordination breakdowns, in which unresolved dependencies between support actors and customers directly impede issue resolution and sustain operational uncertainty. Response decisions were influenced by escalation routines, fragmented expertise, and information available through support practices, which directly connect to Meta-Inference 3. Decision-making during these incidents was shaped by the information available through support practices and the coordination needed across roles to move cases toward resolution. The findings suggest that transparency is developed through coordination across support roles, rather than through technical oversight alone. This is consistent with research on distributed control and interdependent actors in enterprise ecosystems (Hanseth & Lyytinen, 2010; Jacobides et al. 2018; Kapoor et al. 2021).

The operational consequences of upgrade instability are directly linked to the Information Systems Success model, in which system quality and service quality are central to system success. Upgrade instability reduced system quality when upgraded systems became unavailable, degraded, unstable, or difficult to operate. The issues also reduced service quality when support actors had to manage customer impact under incomplete information and time pressure (DeLone & McLean, 2003).

Overall, upgrade instability cannot be reduced to technical issues alone. It is also shaped by how the organisation makes the breakdown visible, interprets its severity, and coordinates a response. Configuration and compatibility complexity explain why instability emerges, while support practices and sociotechnical filters explain how it becomes visible and actionable. Building on sociotechnical systems theory, upgrade instability should therefore be understood as a sustained operational condition in which technical dependencies, support knowledge,

coordination structures, and decision-making routines jointly shape post-adoption operational outcomes as depicted in Figure 9 (Baxter & Sommerville, 2011; Kapoor et al. 2021; Sarker et al. 2019). The study therefore extends post-adoption literature by showing upgrade instability as a reinforcing sociotechnical cycle rather than a one-off failure event.

5.4 Implications for Practice

This study provides operational insights and guiding principles for organisations in four sections. The first three sections: coordination design, changing or accepting the ongoing condition, re-evaluating, and documentation, are sorted by how the suggestions are featured in Figure 14. The final section on improving decision-making provides a quick-paced summary of what should be prioritised and provided before actors need to make decisions based on the study's findings. See Figure 13 for a summary, key points, and an overview of the connection to the literature for this chapter.

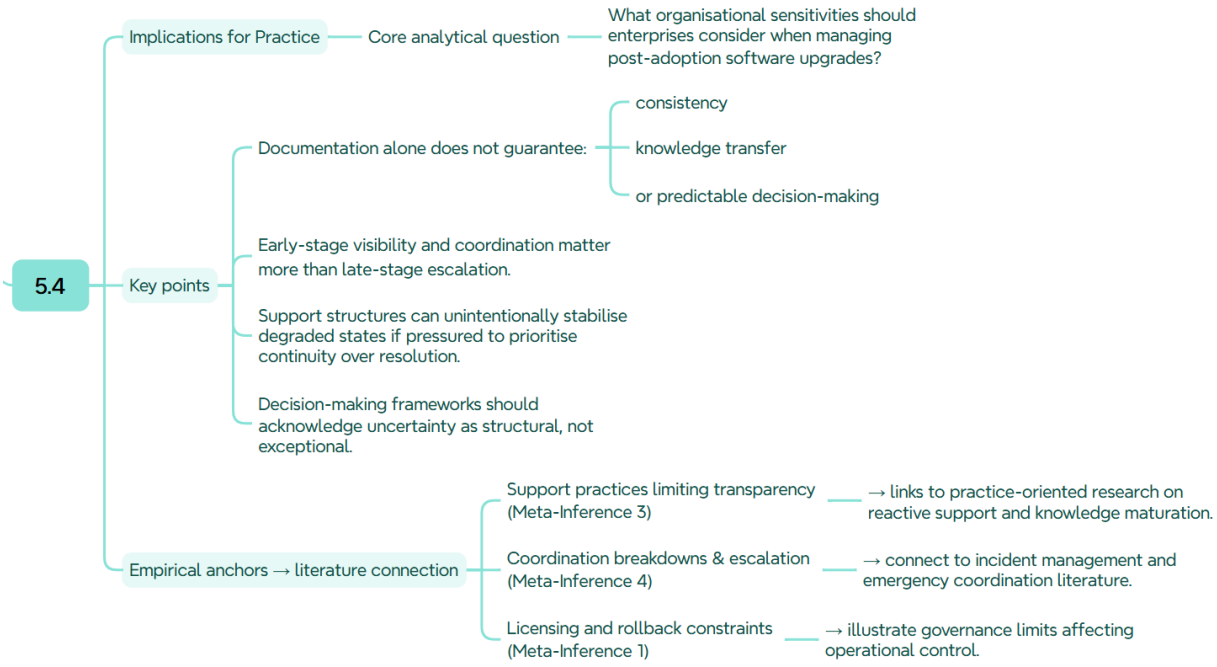


Figure 13: Implication for Practice Mind Map

5.4.1 Coordination and Escalation Routes Enhancing Transparency

The evidence presented in this study underscores the need for transparency to enhance the support process. In addition, escalation in support practices should be defined as a coordination mechanism (Adel et al. 2025; Costa et al. 2024; Kotlarsky et al. 2019). As a matter of fact, each escalation introduces handovers and potential context loss. Therefore, organisations should focus on preserving diagnostic context across support levels, as shown in Figure 14, Preserve Escalation Context. Improving the quality of data collected by the Support Teams, for example, by enhancing metrics for the impact factor of cases.

The study shows that long resolution periods are due not only to technical complexity but also to distributed responsibilities and delayed information exchange (Adel et al. 2025; Figalist et

al. 2019). Thus, one improvement would be earlier visibility into the system state, as demonstrated in Figure 14 (Improve Early Visibility). In some cases, improving visibility into customer configurations may be more valuable than further technical escalation. The organisation should take proactive steps to enhance visibility by collecting data through the interface. The visibility is currently limited to user complaints, with fragmented knowledge. Additionally, to address knowledge gaps, organisations should prioritise the completeness of information at handover points, including logs, configuration details, and prior troubleshooting actions (Adel et al. 2025; Costa et al. 2024), as illustrated in Figure 14, Standardise Handover Info. The takeaway is that reducing unnecessary escalation can be as important as accelerating it when authority and expertise are missing. Thus, decreasing the context lost via handovers (Figalist et al. 2019).

5.4.2 Upgrade Instability from Ongoing Operational Condition

As the findings show, operational pressure during high-severity upgrade incidents often prioritises rapid stabilisation, allowing degraded system states to persist (Adel et al. 2025; Feldman et al. 2017; Kotlarsky et al. 2019; Vert & Sharpanskykh, 2026). The Track Recurring Upgrade Fixes suggestion, shown in Figure 14, for organisations to mitigate this problem is to better understand it by monitoring configuration changes. Thereby tracking how fixes accumulate across upgrades. Because rollbacks are rarely used, decisions made after deployment can have long-term effects (Adel et al. 2025; Jabin et al. 2024). This makes early stabilisation choices especially important. Additionally, the suggestion Review Upgrade Governance Ownership, to mitigate these high-severity issues, governance structures should be reviewed to ensure that decision authority and system knowledge are aligned (Jacobides et al. 2024; Rydén Sonesson et al. 2021). A drill-down mitigation action for this suggestion is to align the Aligning Authority with System Knowledge, particularly in addressing licensing and upgrade decisions. The result should increase the organisation's awareness of how often issues recur after temporary solutions are implemented.

A suggestion would be for organisations to choose to accept the ongoing operational condition. Hence, continuing to reduce escalations and make room to solve other prioritised problems, while recognising that an operational system is not necessarily entirely stable (Kotlarsky et al. 2019). Therefore, shifting the perspective to clarify when to make this trade-off and to recognise that these decisions are within the organisation's control. This helps organisations assess what can remain as is, what requires follow-up, and what should be addressed in future upgrades. This distinction supports more realistic planning, clearer ownership, and better long-term system management. Therefore, shifting the focus from closing cases to understanding and managing residual instability in the system (Jacobides et al. 2024; Kotlarsky et al. 2019).

5.4.3 Support Practices Beyond Documentation

This study provides practical examples of how documentation practices can be improved. Support organisations benefit from mechanisms that capture recurring workarounds and turn them into shared guidelines, reducing reliance on individual experience (Costa et al. 2024). This is mentioned in Figure 14 as Capture Recurring Workarounds. The findings suggest that organisations should not count on documentation as a comprehensive knowledge bank for handling upgrade instability. Documentation should be used as a baseline reference, not as a

full reflection of post-upgrade system behaviour. It often lags behind system changes, especially after major upgrades, so gaps should be expected and planned for (Feldman et al. 2017; Sommerville et al. 2012).

The qualitative interviews showed that informal workarounds and experience-based knowledge naturally develop during post-upgrade support activities. A suggestion, as depicted in Figure 14, to integrate Customer Configuration Insight, would be to capture, review, and formalise these methods where appropriate, instead of them remaining undocumented and isolated (Jakobsen & Vanhaverbeke, 2025). Once organisations have navigated these materials, they can focus on improving support practices. From Figure 14, Include Real Incidents in Onboarding. Thus, Training and onboarding should recognise that post-upgrade support relies on experience and situational judgment. This can be supported by incorporating real incident cases and operational examples (Wenrich & Ahmad, 2009). This leads to a recap figure (Figure 14) that summarises the main practical suggestions and their relationship to the study's findings.

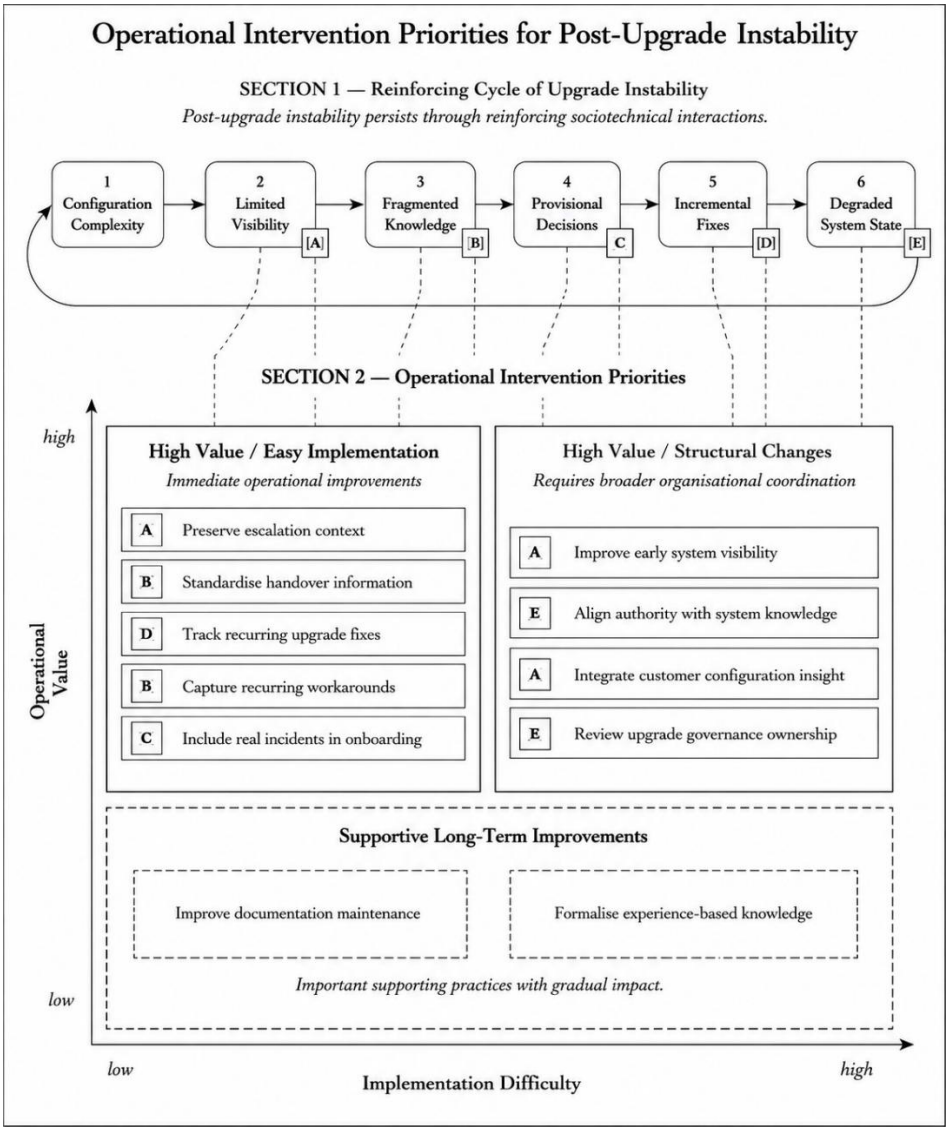


Figure 14: Practical Implication Implementation Strategy

Figure 14 shows an implementation strategy for organisations connecting the issues to the reinforcing cycle of sustained instability. The strategy is based on the chapter's suggestions. The recommendation is analysed in terms of operational value versus implementation difficulty. Therefore, organisations can easily prioritise, prompting consideration of their generalisability beyond this context. When organisations have aligned their shortcomings with these suggested practical implications, it is logical to move on to improving the decision-making process.

5.4.4 Certain Decision-Making

This study has brought to light evidence that decisions during upgrade incidents are made with incomplete and evolving knowledge of the system. Consequently, these decisions are provisional, iterative, and shaped by organisational constraints and limited system transparency (Collopy et al. 2020). The study, therefore, offers five suggestions for mitigating the problem by addressing it from several structural perspectives. First, processes and tools should be designed with the recognition that incomplete information is normal, not the exception (Vert & Sharpanskykh, 2026). Second, experience-based judgement is often necessary; however, organisations can reduce risk by ensuring that experience is distributed across roles (Collopy et al. 2020). Third, the customer's operational context should be explicitly integrated into the prioritisation structure, recognising that technical compatibility issues do not imply equal urgency (Kotlarsky et al. 2019). Fourth, when troubleshooting time comes, all recent information must be available to specialists. When it's hard to predict what information will be needed beforehand, similar prior issues should guide and enhance information collection from the customer (Costa & Fontão, 2024). Fifth, aiming for perfect diagnoses before action is not optimal; organisations should focus on reversible decisions where possible and on clear documentation of assumptions when reversibility is limited (Adel et al. 2025; Vert & Sharpanskykh, 2026).

5.5 Generalisability of Findings

This section discusses the transferability of the findings beyond the case organisation. Since this study examines a single enterprise IT ecosystem, the findings are not intended to be statistically generalisable to all organisations. Instead, the study supports analytical generalisation by connecting empirical findings to theoretical mechanisms that may apply to comparable contexts (Tsang, 2014; Yin, 2018). As illustrated in Figure 15, the generalisable contribution is not the exact distribution of issue categories, impact scores, escalation patterns, or resolution times, since these are shaped by the specific product, organisation, and data period. Rather, the broader contribution lies in the sociotechnical mechanisms identified through the integrated meta-inferences in Chapter 4 (Section 4.3). These mechanisms explain how configuration complexity, support practices, fragmented knowledge, distributed responsibility, and decision-making under uncertainty interact to produce and sustain upgrade instability.

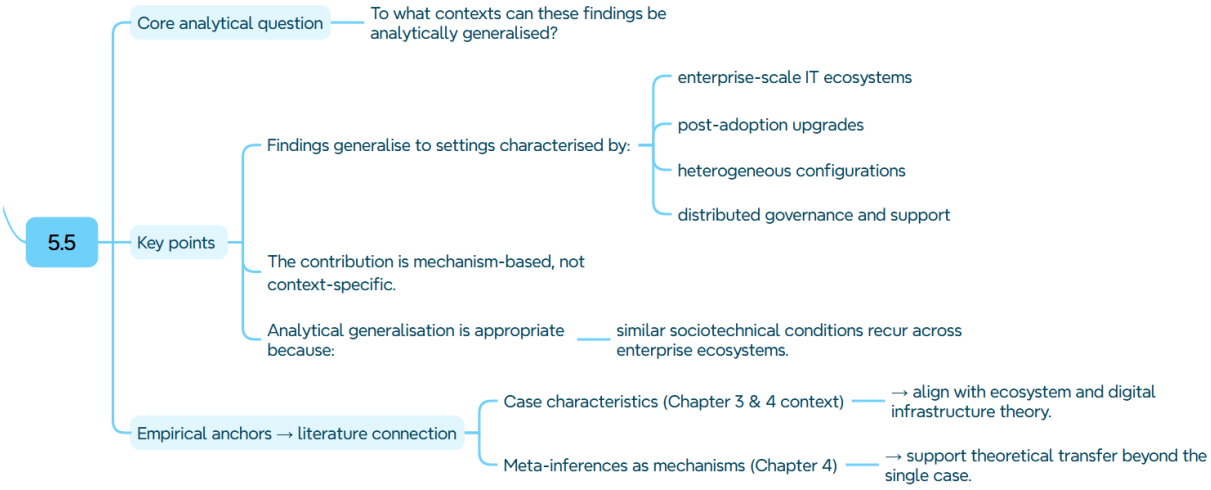


Figure 15: Analytical Generalisation Mind Map

The case examined in this study reflects conditions that are likely to exist in comparable enterprise IT ecosystems. The analysed product operates in an enterprise environment where upgrades interact with customer-specific configurations, licensing arrangements, component dependencies, support routines, and distributed responsibility structures. These characteristics are not unique to the case organisation, as similar conditions are likely to be found in other enterprise IT ecosystems where software vendors, support teams, product teams, system integrators, and customers share responsibility for system stability. While the specific licensing arrangements are case-specific, the broader conditions of heterogeneity, technical interdependence, modular components, distributed actors, and shared responsibility are consistent with prior research on information infrastructures and enterprise ecosystems (Hanseth & Lyytinen, 2010; Jacobides et al. 2018; Kapoor et al. 2021).

The findings are especially relevant where governance and system knowledge are distributed across several actors. In this study, upgrade instability was shaped by unclear ownership, handoffs between support levels, limited authority to act, and dependence on specialised knowledge. Similar issues have been identified in research on knowledge boundaries and IT support coordination, where fragmented expertise, role dependencies, and unclear coordination routines can slow problem-solving and increase uncertainty (Kotlarsky et al. 2015, 2019). This strengthens the relevance of the findings for enterprise settings where no single actor has full visibility or control over the upgrade environment.

The findings are also relevant to organisations managing post-adoption upgrades across complex dependency chains. Prior research shows that upgrade failures can arise from hidden dependencies, integration risks, version mismatches, and cross-version interactions in distributed systems (Dumitraş & Narasimhan, 2009; Zhang et al. 2021). In this study, similar mechanisms were visible through configuration and compatibility issues, component-specific failures, delayed diagnosis, and reliance on support practices to interpret post-upgrade behaviour (Sections 4.1-4.3). Different organisations may have different system architectures, monitoring tools, rollback options, support models, and governance structures. However, these contextual differences do not weaken the theoretical contribution; rather, they clarify its conditions of relevance. The theoretical explanation developed in this study is transferable to comparable contexts where technical dependencies, support practices, distributed governance, and decision-making routines interact during post-adoption upgrades (Tsang, 2014; Yin, 2018).

6 Conclusion

This thesis examined post-adoption software upgrade instability in enterprise IT ecosystems from a sociotechnical perspective. The purpose of the study was to identify and explain the sociotechnical factors that contribute to upgrade instability after software systems have already been adopted and are in operational use. The study was guided by the following research question: What sociotechnical factors contribute to instability in software upgrades in enterprise IT ecosystems?

In conclusion, the study's main contribution is that upgrade instability is not a temporary condition but a persistent operational issue. Instability stems from the interaction of technical complexity, support practices, coordination structures, and decision-making under uncertainty. Moreover, upgrade instability is sustained by reinforcing sociotechnical mechanisms over the long term instead of being resolved through isolated corrective actions. These mechanisms form an enterprise IT ecosystem in which organisational, technical, and ecosystem conditions are linked across conditions, processes, and outcomes. By uncovering these mechanisms, the study provides a process-based explanation of how upgrade instability is generated and sustained. Thereby answering the research question and advancing understanding of post-adoption dynamics in complex enterprise environments.

The reality is that nearly 40% of support tickets are linked to upgrade-related instability, and 80% are categorised as high-impact, underscoring the scale and urgency of the problem. While these figures alone highlight a significant operational burden, the findings suggest that the issue extends far beyond individual organisations. Thus, pointing to a broader pattern across complex enterprise IT ecosystems. The study combines quantitative data (patterns) and qualitative insights (explanations), suggesting that instability follows a cycle that begins with configuration complexity. Decisions are frequently made with limited system visibility, fragmented knowledge, and distributed responsibilities, making it difficult to fully diagnose and resolve upgrade issues. This contribution highlights why substantial organisational effort does not necessarily restore stability. Instability, therefore, becomes part of ongoing operational work, where system performance, service quality, and coordination are continuously managed. This underscores the importance of analysing post-adoption dynamics to understand long-term operational outcomes in enterprise IT environments.

The study offers practical guidance for organisations seeking to address this challenge. The findings are most applicable to similar post-adoption enterprise IT ecosystems. Main suggestions for the organisation are to improve system visibility by strengthening support practices beyond formal documentation, including knowledge sharing, incident learning, and experience-based support. Coordination and escalation processes should be redesigned to reduce delays, preserve context, and minimise information loss between support roles. Decision-making authority should be aligned with the operational consequences of upgrade-related decisions across interconnected systems. By focusing on these areas, organisations can move from reactive issue handling to a more structured, transparent approach to managing post-upgrade instability.

This study contributes to the field by reframing upgrade instability as an ongoing, dynamic process embedded in sociotechnical interactions. It extends existing research by demonstrating how instability is continuously reproduced through everyday operational

practices. Finally, it offers both a theoretical explanation and a practical foundation for more effective management of complex enterprise systems.

6.1 Future Research

The interconnected relationship between software instabilities and ecosystems with a sociotechnical aspect is underexplored. Thus, many opportunities lie in extending the research, and the first step would be to replicate these findings. Thereby, reproduce the study across multiple enterprise ecosystems to compare sociotechnical patterns. Thus, including different industry contexts, especially safety-critical domains, which are prioritised based on the findings. It should also assess whether upgrade instability may carry regulatory, ethical, and societal consequences beyond operational disruption. These environments often have stricter constraints, and those constraints on influence over downtime, rollback options, and decision authority could potentially intensify stabilisation-over-resolution dynamics.

Another suggestion is to investigate how governance and ownership models influence escalation outcomes and the authority to resolve them. The findings indicate that inefficient governance and unclear ownership limit the ability to implement permanent corrective action. Future research could explicitly examine how different governance and ownership models affect escalation paths and the authority to resolve issues. These suggestions would extend sociotechnical system theory by linking institutional arrangements directly to operational outcomes and, finally, by exploring the role of tooling and automation in shaping the visibility of the system state to support coordination and decision-making. This focus reflects a core motivation underlying the present study, namely, to understand how sociotechnical infrastructures translate system behaviour into organisational action.

Appendix 1 - AI contribution statement

Use of AI tools:

- Supported idea generation, brainstorming, and chapter structuring (ChatGPT, Gemini, Microsoft 365 Copilot)
- Assisted with improving language clarity, conciseness, grammar, and readability (ChatGPT, Gemini, Grammarly, Microsoft 365 Copilot)
- Helped rephrase sentences, reduce repetition, and improve transitions and flow (ChatGPT, Gemini, Microsoft 365 Copilot)
- Supported refinement of academic phrasing, conceptual framing, and table structures (ChatGPT, Gemini, Microsoft 365 Copilot)
- Assisted in aligning findings, discussion, conclusions, and practical implications (ChatGPT, Microsoft 365 Copilot)
- Supported development of conceptual models, visual figure ideas, and summaries (ChatGPT, Gemini, Microsoft 365 Copilot). Figures generated by ChatGPT with detailed instructions from the authors and content set up by the authors.
- Assisted with organising and exploring interview transcriptions by identifying recurring themes and patterns for manual review by the authors (Microsoft 365 Copilot)
- Assisted with sorting and structuring support ticket summaries according to categories defined and validated by the authors (Microsoft 365 Copilot)
- Assisted with organising literature, tracking references, identifying connections between themes, and recommending related literature (NotebookLM)
- Used to navigate and verify referencing practices, including the LUSEM referencing guide and thesis-writing material (NotebookLM)
- Supported navigation of support ticket data structures, understanding extraction capabilities, and handling large volumes of support-related data (Case company internal AI tool)

All AI-generated outputs were critically reviewed, validated, edited, and integrated manually by the authors. AI tools were used as supportive aids only and did not replace independent academic judgement. All interpretations, analytical decisions, conclusions, and final content remain the responsibility of the authors.

Appendix 2 – Support Ticket Data Extraction and Classification Schema

This schema defines the structured variables extracted from unstructured support ticket data using a two-stage, AI-assisted classification pipeline. In Stage 1 (Data Filtering), the dataset was filtered to identify upgrade-related incidents using explicit inclusion and exclusion rules that required exact textual triggers. Ambiguous cases were conservatively classified as non-upgrade-related to prevent false positives. In Stage 2 (Structured Classification), a structured data schema and classification rules guided the extraction of the remaining variables, requiring the assignment of one dominant value per dimension based solely on explicit or clearly supported evidence in the ticket text.

The model was explicitly restricted from inferring or fabricating causes, impacts, or escalation behaviour. Where evidence was insufficient, the value "Unknown" was assigned to preserve analytical validity and consistency across cases. The schema includes only observable, explicitly documented attributes in support tickets. Interpretive constructs such as uncertainty, coordination breakdown, or knowledge gaps were not inferred during the extraction process, since digital trace data alone do not provide sufficient social context for such interpretations (Howison et al. 2011), but were instead examined in the qualitative phase. This ensures analytical reliability and aligns with the research design.

Stage 1: Data filtering for upgrade relevance (applied to all tickets)

Variable	Description	Allowed Values
is_upgrade_related	Boolean representation	true/false
Trigger_Phrase	Exact phrase indicating upgrade relevance	Extracted text, e.g., upgrade, update, version change, migration / "none"
Reason	Short justification (max 10 words)	Free text

The Stage 2 categories were developed inductively through repeated review of the support ticket content, recurring terms in the dataset, and the organisation's support context. Prior literature on upgrade instability, support practices, and distributed system failures was used as a sensitising background to guide interpretation while keeping the categories grounded in the ticket data. The final category labels were therefore shaped by the empirical ticket content and refined to ensure consistency across the classification process.

Stage 2: Structured Classification (applied to upgrade-related tickets only)

Item	**Variable**	**Description**	**Allowed Values**
Case Context & Upgrade Alignment	Case_Number	Unique identifier for each support ticket	[Alphanumeric String]
	Upgrade_Phase	Timing of issue relative to upgrade event	Pre-upgrade / During upgrade / Post-upgrade / Unknown
Issue Characterisation	Issue_Category	The technical domain responsible for the issue	Service & application failure / Database or storage issue / Licensing & registration / Component-specific failure / Configuration or compatibility issue / Unknown
Operational Impact	Impact_Score	Severity of disruption based on predefined criteria	5 (Critical) / 4 (Major limitation) / 3 (Degraded) / 1–2 (Minor) / Unknown
	Impact_Evidence	Supporting textual evidence from the ticket	[Extracted text or paraphrased evidence]
Escalation & Technical Handling	Escalation_Level	Highest level of organisational involvement	Resolved at first or second-line / Escalated to third-line / Escalated to software-product team / Unknown
	Escalation_Justification	Evidence supporting escalation classification	[Extracted text evidence]
	Escalated_to_Technical_Fix	Whether resolution required product/code-level intervention	Yes / No / Unknown
Documentation Effectiveness	Documentation_Issue	Explicit issue with official documentation	Missing / Incorrect / Confusing or misleading / No documentation issue evident

Appendix 3 – Dyadic Interview Guide

1. [Show the percentage of upgrade failures in total support cases for the product.] Upgrade-related issues make up a significant portion of support cases. Can you walk us through a recent upgrade issue you handled and what made it challenging?
    a. Where did the complexity come from?
    b. What made it difficult to resolve?
2. [Connect to Issue Categories Bar chart] We've identified several recurring upgrade issue types in our data, such as service and application failure, licensing and registration issues, and component-specific failures. From your experience, there seem to be real pain points. What is it about these top issue categories specifically that makes the organisation work so much harder to close them?
    a. What makes them harder, technical complexity, limited visibility or hard to coordinate?
    b. Do they require more escalation or cross-team involvement?
    c. Are these issues predictable, or do they emerge unexpectedly in practice?
3. [Connect to Issue categories, Bar Chart and Complexity to solve] We also see that some issue types take longer to resolve. How does the nature of the issue affect your ability to make decisions and move toward resolution?
    a. When do you lack sufficient system information or visibility?
    b. How do you assess risks when the root cause is unclear?
    c. Do certain issues force you to make decisions under uncertainty?
4. [Involvement of roles; Escalation to technical fix] Our data shows that several cases involve multiple escalations. Do these situations require cross-role coordination, and where does this process become difficult? You can use the Support Journey Mind Map for support.
    a. Where do dependencies between roles create delays?
    b. What makes responsibilities unclear in practice?
    c. How does a lack of shared understanding affect resolution?
5. In situations where issues are unclear and involve multiple dependencies, how do you actually make decisions about what to do next?
    a. What kind of information is typically missing?
    b. How confident are those decisions?

   c. Do you ever have to act before fully understanding the issue?

6. *"reliance on documentation ensures consistency, facilitates training for new agents, and provides a centralised knowledge base for resolving diverse issues".* How would you say that the documents provided for support processes are accurate in regard to consistency?

   a. Follow-up questions: Would you say that the training is sufficient?

   b. Would you say that the documents are sufficient for diverse issues?

7. [Show distribution of impact levels] We've categorised operational impact levels (1–5). We see that some issues have a much higher operational impact than others. What determines whether an issue becomes high-impact, and how does that influence how it is prioritised?

   a. Is prioritisation driven more by severity or frequency?

   b. How do customer consequences influence decisions?

   c. Do high-impact issues change coordination or escalation behaviour?

8. How do industry standards or formal requirements influence how these issues are handled?

   a. Do they ever limit your ability to resolve issues effectively?

9. Closing Reflection

   a. What makes certain upgrade issues especially hard to resolve compared to others?

   b. Is there something about upgrade issues that we haven't asked but should understand?

   c. Anything else you want to add?

Appendix 4 – Coding Table of Interviews

Note: The interviews were conducted as group discussions. Individual speaker attribution was not consistently identifiable; therefore, excerpts are attributed to interview sessions rather than respondents.

Interview A

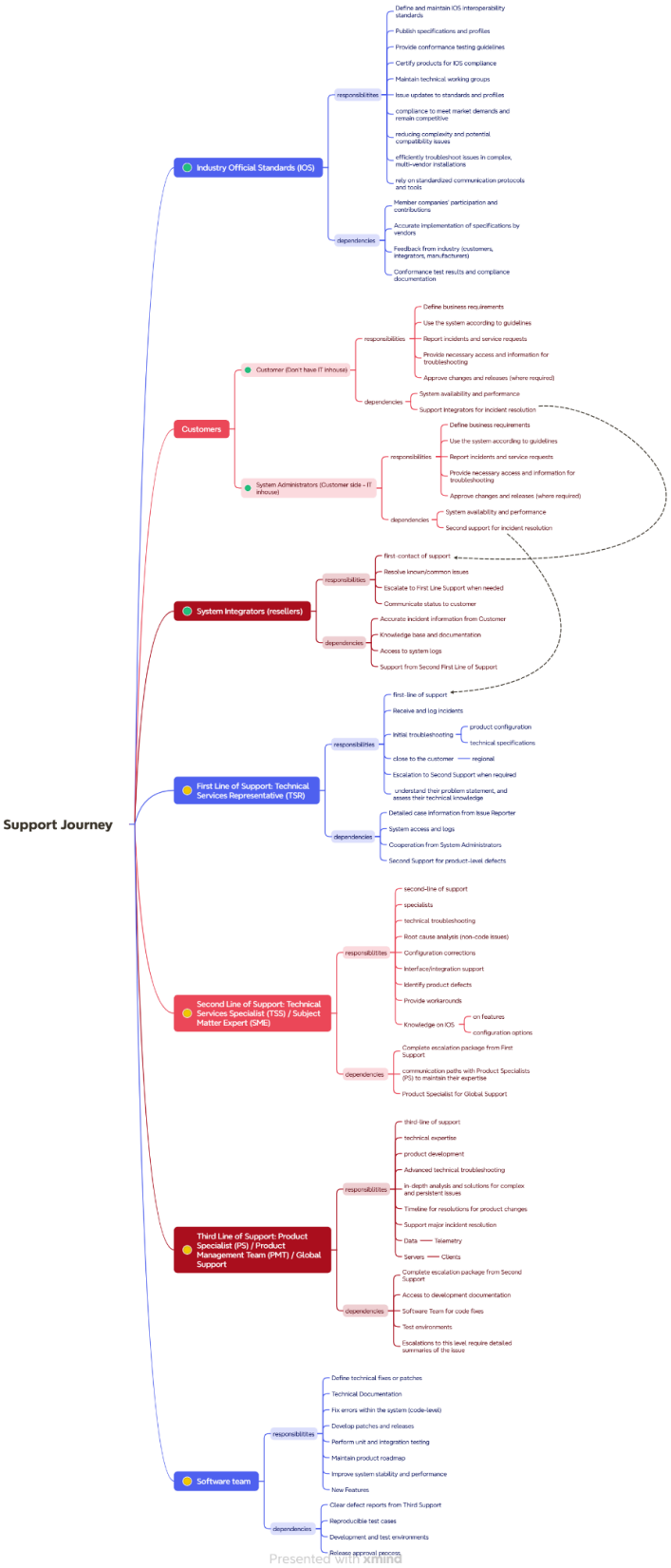
Quote ID	Original Excerpt (Redacted)	Code	Theme
A.1	"You often need to figure out what the system is actually doing after the upgrade."	Figuring out system behaviour	Support practice
A.2	"It's not obvious from the interface, so you rely a lot on logs."	Investigating logs due to UI opacity	Support practice
A.3	"You try different things to see what fixes it, especially early on."	Trial and error troubleshooting	Support practice
A.4	"Most training happens from experience, not manuals."	Training based on experience	Knowledge fragmentation
A.5	"The case moves between different support levels."	Escalation between levels	Coordination breakdowns
A.6	"When it moves between teams, context can be lost."	Handoffs across roles	Coordination breakdowns
A.7	"People try things that worked before, but not anymore."	Outdated troubleshooting	Degraded system state
A.8	"Older methods don't apply to the upgraded system."	Legacy methods post-upgrade	Degraded system state
A.9	"By the time it reaches us, the system state is worse."	System altered before escalation	Degraded system state
A.10	"Once you upgrade, it's very hard to go back."	No rollback option	System irreversibility
A.11	"You're basically stuck in the new version."	Irreversible upgrades	System irreversibility
A.12	"There are many components involved that can fail."	Interdependent components	Ecosystem complexity
A.13	"Sometimes it's unclear who owns the license or organisation."	Licensing ownership unclear	Governance deadlocks
A.14	"Different organisational entities are involved."	Fragmented entities	Governance deadlocks

A.15	"We don't have the authority to move licenses."	No authority to transfer	Governance deadlocks
A.16	"Manual actions are often required to fix it."	Manual intervention required	Governance deadlocks

Interview B

Quote ID	Original Excerpt (Redacted)	Code	Theme
B.1	"Sometimes you create your own workaround because nothing else exists yet."	Local workaround development	Support practice
B.2	"The documentation is correct when it's released, but not always later."	Documentation lag	Knowledge fragmentation
B.3	"Information is spread across different manuals and places."	Fragmented documentation	Knowledge fragmentations
B.4	"We sometimes create PDF guides or screenshots ourselves."	Informal PDFs	Knowledge fragmentation
B.5	"A lot of time is spent waiting for customers to send logs."	Waiting for customer input	Coordination breakdowns
B.6	"There's a lot of back and forth before anything moves forward."	Repeated back-and-forth	Coordination breakdowns
B.7	"When the installer fails, it's not clear where the logs are."	Installer failure opacity	Ecosystem complexity
B.8	"You often act before fully understanding the root cause."	Acting without root cause	Decision-making under uncertainty
B.9	"You rule out possibilities one by one."	Ruling out possibilities	Decision-making under uncertainty
B.10	"Experience guides many decisions."	Experience-based judgment	Decision-making under uncertainty
B.11	"You don't escalate immediately unless necessary."	Reluctance to escalate	Decision-making under uncertainty
B.12	"Some customers cannot afford downtime."	Safety-critical customers	Customer importance
B.13	"Systems in hospitals or prisons are treated differently."	Critical domains	Customer importance
B.14	"These systems affect public and personal safety."	Safety implications	Customer importance

Appendix 5 – Support Journey Mind Map



References

Abbasi, A., Zhou, Y., Deng, S., & Zhang, P. (2018). Text Analytics to Support Sense-Making in Social Media: A Language-Action Perspective, *Management Information Systems Quarterly*, vol. 42, no. 2, pp.427–464

Adel, A., Alani, N. H. S., Jan, T., & Prasad, M. (2025). A Review of Major ICT Failures and Recovery Strategies: Strengthening Digital Resilience, *Computers & Security*, vol. 159, p.104678

Ågerfalk, P. J. (2013). Embracing Diversity through Mixed Methods Research, *European Journal of Information Systems*, vol. 22, no. 3, pp.251–256

Anthony, R. J. (2016). The Architecture View, in *Systems Programming: Designing and Developing Distributed Applications*, pp.277–382

Banks, M., & Zeitlyn, D. (2015). Visual Methods in Social Research, 2nd edn., Los Angeles: Sage

Baxter, G., & Sommerville, I. (2011). Socio-Technical Systems: From Design Methods to Systems Engineering, *Interacting with Computers*, vol. 23, no. 1, pp.4–17

Benbya, H., & McKelvey, B. (2006). Using Coevolutionary and Complexity Theories to Improve IS Alignment: A Multi-Level Approach, SSRN Scholarly Paper, 1560603

Bhattacherjee, A. (2001). Understanding Information Systems Continuance: An Expectation-Confirmation Model, *MIS Quarterly*, vol. 25, no. 3, pp.351–370

Bowen, G. A. (2006). Grounded Theory and Sensitizing Concepts, *International Journal of Qualitative Methods*, vol. 5, no. 3, pp.12–23

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact, *Management Information Systems Quarterly*, vol. 36, no. 4, pp.1165–1188

Collopy, A. X., Adar, E., & Papalambros, P. Y. (2020). On the Use of Coordination Strategies in Complex Engineered System Design Projects, *Design Science*, vol. 6, p.e32

Costa, L. A., Fontão, A., Santos, R. P., & Serebrenik, A. (2024). Towards an Incident Management Framework in Proprietary Software Ecosystems, arXiv:2410.09320

Crilly, N., Blackwell, A. F., & Clarkson, P. J. (2006). Graphic Elicitation: Using Research Diagrams as Interview Stimuli, *Qualitative Research*, vol. 6, no. 3, pp.341–366

Debortoli, S., Müller, O., Junglas, I., & Brocke, J. vom. (2016). Text Mining For Information Systems Researchers: An Annotated Topic Modeling Tutorial, *Communications of the Association for Information Systems*, [e-journal] vol. 39, no. 1

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update, *Journal of Management Information Systems*, vol. 19, no. 4, pp.9–30

Desouza, K. C., Watson, R. T., & Xie, Y. (2025). Managing Digital Dependencies in a Connected Society: Policy Options for Ecosystem Transitivity, *Business & Information Systems Engineering*, vol. 223

Dissanayake, I., Mehta, N., Palvia, P., Taras, V., & Amoako-Gyampah, K. (2019). Competition Matters! Self-Efficacy, Effort, and Performance in Crowdsourcing Teams, *Information & Management*, vol. 56, no. 8, p.103158

Dissanayake, N., Jayatilaka, A., Zahedi, M., & Babar, M. A. (2022). Software Security Patch Management - A Systematic Literature Review of Challenges, Approaches, Tools and Practices, *Information and Software Technology*, vol. 144, p.106771

Drechsler, K., Grisold, T., Gau, M., & Seidel, S. (2022). Digital Infrastructure Evolution: A Digital Trace Data Study, in *ICIS 2022 Proceedings*, 9 December 2022

Dumitraş, T., & Narasimhan, P. (2009). Why Do Upgrades Fail and What Can We Do about It? Toward Dependable, Online Upgrades in Enterprise System., in J. M. Bacon & B. F. Cooper (eds), *Erratum to: Middleware*, Vol. 5896, [e-book] Berlin, Heidelberg: Springer Berlin Heidelberg, pp.349–372

Feldman, G., Shah, H., Chapman, C., Pärn, E. A., & Edwards, D. J. (2017). A Systematic Approach for Enterprise Systems Upgrade Decision-Making: Outlining the Decision Processes, *Journal of Engineering, Design and Technology*, vol. 15, no. 6, pp.778–802

Figalist, I., Elsner, C., Bosch, J., & Olsson, H. (2019). Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems, in P. Kruchten, S. Fraser, & F. Coallier (eds), *Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montréal, QC, Canada, May 21–25, 2019, Proceedings*, Vol. 355, [e-book] Cham: Springer International Publishing, pp.189–206

Ghazawneh, A., & Henfridsson, O. (2013). Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model, *Information Systems Journal*, vol. 23, no. 2, pp.173–192

Goldkuhl, G. (2012). Pragmatism vs Interpretivism in Qualitative Information Systems Research, *European Journal of Information Systems*, vol. 21, no. 2, pp.135–146

Hanseth, O., & Lyytinen, K. (2010). Design Theory for Dynamic Complexity in Information Infrastructures: The Case of Building Internet, *Journal of Information Technology*, vol. 25, no. 1, pp.1–19

Haraldsson, B., & Staron, M. (2025). Aspects of Complexity in Automotive Software Systems and Their Relation to Maintainability Effort. A Case Study, in *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering*, EASE '25: Evaluation and Assessment in Software Engineering, 17 June 2025, Istanbul Turkiye: ACM, pp.68–78

Henfridsson, O., & Bygstad, B. (2013). The Generative Mechanisms of Digital Infrastructure Evolution, *MIS Quarterly*, vol. 37, no. 3, pp.907–931

Hou, F., & Jansen, S. (2023). A Systematic Literature Review on Trust in the Software Ecosystem, *Empirical Software Engineering*, vol. 28, no. 1, p.8

Howison, J., Wiggins, A., & Crowston, K. (2011). Validity Issues in the Use of Social Network Analysis with Digital Trace Data, *Journal of the Association for Information Systems*, [e-journal] vol. 12, no. 12

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis, *Qualitative Health Research*, vol. 15, no. 9, pp.1277–1288

Jabin, M. S. R., Wepa, D,. & Hassoun, A. (2024). A Case Report of System Configuration Issue in Medical Imaging Due to System Upgrade– Changes in Hardware and Software, *Frontiers in Digital Health*, vol. 6, p.1371761

Jacobides, M. G., Cennamo, C., & Gawer, A. (2018). Towards a Theory of Ecosystems, *Strategic Management Journal*, vol. 39, no. 8, pp.2255–2276

Jacobides, M. G., Cennamo, C., & Gawer, A. (2024). Externalities and Complementarities in Platforms and Ecosystems: From Structural Solutions to Endogenous Failures, *Research Policy*, vol. 53, no. 1, p.104906

Jakobsen, A. M. S. Ø., & Vanhaverbeke, W. (2025). A Socio-Technical Perspective on Product Configuration Systems: Insights from Grundfos

Jasperson, J. S., Carter, P. E., & Zmud, R. W. (2005). A Comprehensive Conceptualization of Post-Adoptive Behaviors Associated with Information Technology Enabled Work Systems, *MIS Quarterly*, vol. 29, no. 3, pp.525–558

Kapoor, K., Ziaee Bigdeli, A., Dwivedi, Y. K., Schroeder, A., Beltagui, A., & Baines, T. (2021). A Socio-Technical View of Platform Ecosystems: Systematic Review and Research Agenda, *Journal of Business Research*, vol. 128, pp.94–108

Khoo, H. M., Robey, D., & Rao, S. V. (2011). An Exploratory Study of the Impacts of Upgrading Packaged Software: A Stakeholder Perspective, *Journal of Information Technology*, vol. 26, no. 3, pp.153–169

Khoo, M. H., & Robey, D. (2007). Deciding to Upgrade Packaged Software: A Comparative Case Study of Motives, Contingencies and Dependencies, *European Journal of Information Systems*, vol. 16, no. 5, pp.555–567

Klein, H. K., & Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, *Management Information Systems Quarterly*, vol. 23, no. 1, pp.67–93

Knoche, H., & Hasselbring, W. (2021). Continuous API Evolution in Heterogenous Enterprise Software Systems, in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, March 2021, pp.58–68

Kotlarsky, J., Van Den Hooff, B., & Geerts, L. (2019). Under Pressure: Understanding the Dynamics of Coordination in IT Functions under Business-as-Usual and Emergency Conditions, *Journal of Information Technology*, vol. 35, no. 2, pp.94–122

Kotlarsky, J., Van Den Hooff, B., & Houtman, L. (2015). Are We on the Same Page? Knowledge Boundaries and Transactive Memory System Development in Cross-Functional Teams, *Communication Research*, vol. 42, no. 3, pp.319–344

Lobe, B., Morgan, D. L., & Hoffman, K. (2022). A Systematic Comparison of In-Person and Video-Based Online Interviewing, *International Journal of Qualitative Methods*, vol. 21, p.16094069221127068

Malone, T. W., & Crowston, K. (1994). The Interdisciplinary Study of Coordination, *ACM Computing Surveys*, vol. 26, no. 1, pp.87–119

Mayring, P. (2025). Qualitative Content Analysis With ChatGPT: Pitfalls, Rough Approximations and Gross Errors. A Field Report, *Forum Qualitative Sozialforschung Forum: Qualitative Social Research*, [e-journal] vol. 26, no. 1

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). Qualitative Data Analysis: A Methods Sourcebook, 3rd edn, Thousand Oaks, Califorinia: SAGE Publications, Inc

Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology, *Information Systems Research*, vol. 12, no. 3, pp.240–259

Mithas, S., Ramasubbu, N., & Sambamurthy, V. (2011). How Information Management Capability Influences Firm Performance, *Management Information Systems Quarterly*, vol. 35, no. 1, pp.237–256

Morgan, D. L. (2014). Pragmatism as a Paradigm for Social Research, *Qualitative Inquiry*, vol. 20, no. 8, pp.1045–1053

Morgan, D. L., Ataie, J., Carder, P., & Hoffman, K. (2013). Introducing Dyadic Interviews as a Method for Collecting Qualitative Data, *Qualitative Health Research*, vol. 23, no. 9, pp.1276–1284

Müller, O., Junglas, I., Brocke, J. vom, & Debortoli, S. (2016). Utilizing Big Data Analytics for Information Systems Research: Challenges, Promises and Guidelines, *European Journal of Information Systems*, vol. 25, no. 4, pp.289–302

Mumford, E. (2006). The Story of Socio-Technical Design: Reflections on Its Successes, Failures and Potential, *Information Systems Journal*, vol. 16, no. 4, pp.317–342

Myers, M. D., & Newman, M. (2007). The Qualitative Interview in IS Research: Examining the Craft, *Information and Organization*, vol. 17, no. 1, pp.2–26

Petersen, C., & Seymour, L. (2021). Explaining the Network of Factors That Influence the Timing of and Decision to Upgrade Enterprise Systems, in D. Dennehy, A. Griva, N. Pouloudi, Y. K. Dwivedi, I. Pappas, & M. Mäntymäki (eds), *Responsible AI and Analytics for an Ethical and Inclusive Digitized Society: 20th IFIP WG 6.11 Conference on e-Business, e-Services and e-Society, I3E 2021, Galway, Ireland, September 1–3,*

2021, Proceedings, Vol. 12896, [e-book] Cham: Springer International Publishing, pp.506–519

Recker, J. (2013). Scientific Research in Information Systems: A Beginner's Guide, Berlin, Heidelberg: Springer

Ren, Y., Kiesler, S., & Fussell, S. R. (2008). Multiple Group Coordination in Complex and Dynamic Task Environments: Interruptions, Coping Mechanisms, and Technology Recommendations, *Journal of Management Information Systems*, vol. 25, no. 1, pp.105–130

Retana, G. F., Forman, C., Narasimhan, S., Niculescu, M. F., & Wu, D. J. (2018). Technology Support and Post-Adoption IT Service Use: Evidence from the Cloud, *MIS Quarterly*, vol. 42, no. 3, pp.961–978

Rydén Sonesson, T., Johansson, J., & Cedergren, A. (2021). Governance and Interdependencies of Critical Infrastructures: Exploring Mechanisms for Cross-Sector Resilience, *Safety Science*, vol. 142, p.105383

Sarker, S., Chatterjee, S., Xiao, X., & Elbanna, A. (2019). The Sociotechnical Axis of Cohesion for the Is Discipline: Its Historical Legacy and Its Continued Relevance, *Management Information Systems Quarterly*, vol. 43, no. 3, pp.695–719

Saxena, D., & McDonagh, J. (2022). Communication Breakdowns during Business Process Change Projects – Insights from a Sociotechnical Case Study, *International Journal of Project Management*, vol. 40, no. 3, pp.181–191

Silver, M. S. (1991). Decisional Guidance for Computer-Based Decision Support, *MIS Quarterly*, vol. 15, no. 1, pp.105–122

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., Mcdermid, J., & Paige, R. (2012). Large-Scale Complex IT Systems, *Communications of the ACM*, vol. 55, no. 7, pp.71–77

Stahl, B. C., Eden, G., Jirotka, M., & Coeckelbergh, M. (2014). From Computer Ethics to Responsible Research and Innovation in ICT: The Transition of Reference Discourses Informing Ethics-Related Research in Information Systems, *Information & Management*, vol. 51, no. 6, pp.810–818

Strong, D. M., & Volkoff, O. (2010). Understanding Organization—Enterprise System Fit: A Path to Theorizing the Information Technology Artifact, *Management Information Systems Quarterly*, vol. 34, no. 4, pp.731–756

Sunyaev, A., Dehling, T., Strahringer, S., Da Xu, L., Heinig, M., Perscheid, M., Alt, R. &, Rossi, M. (2023). The Future of Enterprise Information Systems, *Business & Information Systems Engineering*, vol. 65, no. 6, pp.731–751

Tamanna, M., Anwar, M., & Stephens, J. D. W. (2025). Security Implications of User Non-Compliance Behavior to Software Updates: A Risk Assessment Study, *Journal of Information Security and Applications*, vol. 93, p.104152

Techakriengkrai, W., Techatassanasoontorn, A. A., & Tan, F. B. (2021). Examining Post-Adoptive Change of Enterprise System Implementations: A Socio-Technical Perspective, *Australasian Journal of Information Systems*, [e-journal] vol. 25

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics, *Information Systems Research*, vol. 21, no. 4, pp.675–687

Uludağ, Ö., Kleehaus, M., Erçelik, S., & Matthes, F. (2019). Using Social Network Analysis to Investigate the Collaboration Between Architects and Agile Teams: A Case Study of a Large-Scale Agile Development Program in a German Consumer Electronics Company, in P. Kruchten, S. Fraser, & F. Coallier (eds), *Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montréal, QC, Canada, May 21–25, 2019, Proceedings*, Vol. 355, [e-book] Cham: Springer International Publishing, pp.137–153

Tsang, E. W. K. (2014). Generalizing from Research Findings: The Merits of Case Studies, *International Journal of Management Reviews*, vol. 16, no. 4, pp.369–383

Venkatesh, V., Brown, S. A., & Bala, H. (2013). Bridging the Qualitative–Quantitative Divide: Guidelines for Conducting Mixed Methods Research in Information Systems1, *Management Information Systems Quarterly*, vol. 37, no. 1, pp.21–54

Venkatesh, V., Brown, S., & Sullivan, Y. (2016). Guidelines for Conducting Mixed-Methods Research: An Extension and Illustration, *Journal of the Association for Information Systems*, vol. 17, no. 7, pp.435–494

Vert, M., & Sharpanskykh, A. (2026). The Resilience of Complex Sociotechnical Systems: A Meta-Review of Conceptualisations, *Systems*, vol. 14, no. 1, p.71

Walsham, G. (2006). Doing Interpretive Research, *European Journal of Information Systems*, vol. 15, no. 3, pp.320–330

Wenrich, K., & Ahmad, N. (2009). Lessons Learned During a Decade of ERP Experience: A Case Study, *International Journal of Enterprise Information Systems*, vol. 5, no. 1, pp.55–73

Yin, R. K. (2018). Case Study Research and Applications: Design and Methods, 6th edn, Los Angeles: SAGE

Yoo, Y., Boland, R. J., Lyytinen, K., & Majchrzak, A. (2012). Organizing for Innovation in the Digitized World, *Organization Science*, vol. 23, no. 5, pp.1398–1408

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research, *Information Systems Research*, vol. 21, no. 4, pp.724–735

Zhang, Y., Yang, J., Jin, Z., Sethi, U., Rodrigues, K., Lu, S., & Yuan, D. (2021). Understanding and Detecting Software Upgrade Failures in Distributed Systems, in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP ’21:

ACM SIGOPS 28th Symposium on Operating Systems Principles, 26 October 2021, Virtual Event Germany: ACM, pp.116–131