The Influence of Early Architectural Decisions on Modular Platform Development

A Framework Supporting Early-Stage Decision-Making

Master's Thesis in Product Development

ANNA JÖNSSON
JOSEF LOMAN

DEPARTMENT OF TECHNOLOGY MANAGEMENT AND ECONOMICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

The Influence of Early Architectural Decisions on Modular Platform Development

A Framework Supporting Early-Stage Decision-Making

ANNA JÖNSSON
JOSEF LOMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Technology Management and Economics
Division of Innovation and R&D Management
Chalmers University of Technology
Gothenburg, Sweden 2026

The Influence of Early Architectural Decisions on Modular Platform Development
A Framework Supporting Early-Stage Decision-Making
ANNA JÖNSSON
JOSEF LOMAN

Supervisor: Magnus Persson, Technology Management and Economics
Examiner: Magnus Persson, Technology Management and Economics

Master's Thesis 2026
Department of Technology Management and Economics
Division of Innovation and R&D Management
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Influence of Early Architectural Decisions on Modular Platform Development
A Framework Supporting Early-Stage Decision-Making
ANNA JÖNSSON
JOSEF LOMAN
Department of Technology Management and Economics
Chalmers University of Technology

Abstract

Platform-based product development is a central strategy for dealing with complexity, enabling reuse and supporting product variation within large-scale organizations, especially within original equipment manufacturing (OEM). Architectural decisions within early phases are, however, greatly affected by high uncertainty, leading to challenges within the organization. This thesis examines one organization in particular, researching ways of working and conditions needed to navigate uncertainty and improving decision-making during early phases of modular platform development.

Distinctive for the study is a qualitative and iterative research where a literature study is combined with empirical data gathered from employees within the organization. The data gathered consist of contextual observations, a survey based on 25 respondents, as well as twelve expert interviews with selected employees within the organization.

The empirical results show a lack of common definitions considering core concepts such as "architecture" and "platform," creating semantic noise and misunderstanding between departments. Furthermore, path dependency is identified where the first project tends to unintentionally dictate the long-term rules and limit the platform's future flexibility. This is amplified by unclear decision mandates, a lack of "platform-first thinking," as well as short-term goals.

To deal with these challenges, the thesis proposes a framework supporting decision-making in early phases structured around five key areas: (1) establishing a common language, (2) ensuring requirements-based definitions of common solutions, (3) conducting platform impact assessments, (4) clarifying cross-functional decision mandate and ownership, as well as (5) enforcing the traceability of decision rationales. The framework is designed as a flexible checklist giving the organization a structured method for dealing with uncertainty in early phases, supporting conscious decisions, while not compromising the platform's long-term flexibility.

Keywords: Product development, decision-making, early-phase, platform development, architectural decisions, uncertainty, systemic learning, shared language, path dependency.

Acknowledgements

We would like to express our sincere gratitude to our supervisor and examiner, Magnus Persson, for his guidance, feedback, and support throughout this thesis. His input has been valuable in shaping the direction of the work and improving the quality of the report.

We would also like to thank the organization where this study was conducted for giving us the opportunity to carry out our master's thesis in close connection with their work. A special thanks is directed to our supervisors at the company for their continuous support, valuable insights, and for helping us navigate the organizational context during the thesis process. We are also grateful to the employees who participated in interviews, answered the survey, and shared their experiences and perspectives. Their openness and insights were essential for the development of this thesis.

Finally, we would like to thank everyone who supported us during the process, both academically and personally.

Anna Jönsson, Josef Loman, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

AI	Artificial Intelligence
BEV	Battery Electric Vehicle
CCB	Configuration Control Board
DSM	Design Structure Matrix
GVWR	Gross Vehicle Weight Rating
OEM	Original Equipment Manufacturer
PBD	Point-Based Design
PLM	Product Lifecycle Management
R&D	Research and Development
RACI	Responsible, Accountable, Consulted, Informed
SBCE	Set-Based Concurrent Engineering
ToC	Trade-off Curves

Contents

1
Introduction

Introduced is the foundation for this project, focused on improving decision-making in early stages within platform-based product development. Initially, a background establishing both theoretical and practical challenges within large-scale product development is introduced. Followed by a description of the organizational context about one of the world's leading original equipment manufacturers (OEM). Furthermore, the chapter defines the thesis purpose, research questions, and limitations, all to provide a clear structure and focus for the project.

1.1 Background

The collaboration between platform architecture and specific module construction requires close contact and a shared vision. The focus of this thesis is targeted on a company that uses an organizational framework based on sharing technology between different products and configurations. Product platforms are used to manage complexity and enable product variety by allowing several products or variants to share a common technical foundation. This foundation can include shared components, modules, subsystems, interfaces, technologies, and design principles that are reused across different products. By using platforms, companies can reduce development time and cost, increase reuse, and support economies of scale, while still offering variation for different customer and market needs (Muffatto and Roveda, 2000; Otto et al., 2016; Li et al., 2016). However, these benefits depend on early architectural decisions regarding modules, interfaces, and shared solutions, since such decisions may influence several future products and functions over time (Li et al., 2016; Muffatto and Roveda, 2000; Merati et al., 2024). Given the size of the company's portfolio and growing needs from customers, many products are designed concurrently.

This thesis will investigate the decision-making process within the organization with extra focus placed on the department of platform architecture, playing a vital part, having a holistic perspective over decision-making and development of platforms.
The organization as a whole is currently reorganizing, giving possibilities for improvements while not disregarding certain matters that did work well in the old organization. This in particular is key and of great interest in investigating the organization. By systematically pinpointing certain ways of working and looking into methods and tools used, important information about how the organization initiates product development in an early stage can be gathered

In large-scale product development, challenges are not only technical but also organizational. Information between departments needs to be accessible and well defined in order to achieve efficient product development (Ulrich, 1995). A common issue is that insufficient communication and a lack of shared vision between organizational departments can lead to delayed design changes. According to the principle of *front-loading*, the cost of correcting design issues increases exponentially as the project progresses. (Tan et al., 2017). Therefore, a need to identify more effective methods in terms of decision-making for how a department initiates its work in early stages, ensuring *correctness from the start,* is identified.

The freedom to change a design drastically reduces as the project progresses (Tan et al., 2017). One of the many reasons why large organizations are working with a platform-based architecture is enabling product variety with the help of modules, reducing complexity and uncertainty while maintaining customer satisfaction in some cases, and offering tailor-made configurations for specific needs (Baldwin et al., 2003). It is easy to imagine how the foundation (read: platform) of these products needs to be planned in every detail regarding interfaces to connecting modules, the lifespan of the platform, the targeted market, and trying to foresee how the industry will look in the next 10 years or more. This creates a high level of uncertainty and sometimes the need to correct decisions taken in an early stage based on new information (Robertson and Ulrich, 1998). This is closely related to the 80–20 rule, which suggests that approximately 80% of product costs are determined by early design decisions (Tan et al., 2017). Studies within the aerospace industry show that as the design matures, the opportunities for life cycle cost savings reduce, further showing that early decision-making is at a crucial stage (Tan et al., 2017).

An additional challenge that large organizations face is the tendency to work in so-called *organizational silos* (Eppinger and Ulrich, 2020). According to Eppinger and Ulrich (2020), a strictly functional organization can lead to organizations optimizing their own (local) strategic goals rather than the overall goal of the organization. This creates an *over-the-wall* mentality, where information is sent between different departments without a foundational understanding and dialogue regarding the next step in the chain. Resulting in a fragmented knowledge stream where critical information risks being lost.
To motivate the risks with fragmented knowledge streams and inefficiency, companies adapt *front-loading*, as a strategy to improve development performance by moving the identification and solving of design problems to earlier phases in the product development process where more flexibility exists (Thomke and Fujimoto, 2000).

Thomke and Fujimoto (2000) describe that effective *front-loading* can be achieved through different approaches, such as transferring lessons learned from previous projects and adapting advanced technologies for early verification, such as digital mock-ups. By systematically transferring knowledge from previous projects, the organization can decrease the number of unplanned changes in an early stage. This proactive setting has shown itself to be a critical success factor, where leading companies such as Toyota have reported that *front-loading* can solve 80% of technical problems before a product even exists, which in the long term can reduce development costs and lead times by 30-40% (Thomke and Fujimoto, 2000). Mastering this is not only a technical advantage but also a core strategic

ability to reach *the correct conclusion from start.*

Despite the evidence of the clear influence of early-stage decision-making, there remains a gap in understanding what practices are most vital at the early phases of a project. To minimize rework and optimize efficient platform development, it is crucial to identify aspects that relate to collaboration between departments to integrate modular development seamlessly into platform architecture.

1.2 Purpose & Research Questions

The foundation for this thesis is stated in one research question: ***What practices should be adopted in early-stage decision-making for modular platform development to navigate high uncertainty?***

This report will often mention "important decisions in early phases," something that the authors will refer to as *decisions that outlast the components*, meaning that focus will be put on specific decisions concerning the holistic and long-term perspective of platform development.

This study investigates organizational decision-making in an early phase with an emphasis on product development and management, where focus will be kept on improving strategies and ways of working. The core is to gain valuable knowledge from the personnel that has been involved in decision-making processes, utilizing this knowledge by comparing it to proven methods based on academic research.

Simplified, the following stated goals are:
1. Identify and analyze internal recurring challenges in cross-functional collaboration during early-stage decision-making.
2. Research and examine methods for decision-making strategies and methods for managing high uncertainty in early phases.
3. Deliver a recommended work process or checklist that minimizes the risk of late changes (rework).

1.3 Limitations

This study focused mainly on hardware development and the specific phase for project initiation. The work does not assess detailed software coding or production logistics but stays at the technical construction and its interfaces.

This work was ongoing for about twenty weeks, meaning that there was a time constraint. This constraint ultimately forced the thesis to narrow its scope to become as qualitative as possible.

The project did not focus on developing a completely new strategy of working but instead researching proven methods and finding alternative ways for the organization that could improve the current situation. Again, time constraints forced the thesis to be hypothetical regarding this matter, since further investigation and testing of new methodology would require more time.

2
Literature Review

This chapter presents the literature acting as a foundation for this study. The purpose of the literature review is to give a deeper understanding of the challenges and strategies that characterize early decision-making within modular platform development under high uncertainty.
The chapter is structured in a holistic way, treating both technical and organizational aspects within these early phases. Initially, sections 2.1 and 2.2 establish the foundational concepts for product platforms, modular architecture, and the ultimate consequences of architectural decisions. After that, sections 2.3 and 2.4 examine strategies to handle uncertainty with a focus on front-loading, path dependency, and set-based concurrent engineering.
Since architectural decisions are deeply connected to organizational structures, sections 2.5 and 2.6 dive deeper into organizational theory and platform governance, including clear frameworks for responsibility. Finally, section 2.7 highlights the importance of organizational communication and traceability of decisions to ensure that the reasoning behind decisions is made clear and stored over time.

2.1 Product Platforms and Modular Product Architecture

Platform-based product development is a central approach for managing complexity, enabling reuse, and supporting product variety in large-scale engineering organizations. Since this thesis focuses on early architectural decisions in modular platform development, it is necessary to clarify the relationship between product platforms, modularity, product architecture, and interfaces. Product platforms provide the overarching logic for reusing shared assets across multiple products, modularity describes how products can be decomposed into reusable building blocks, product architecture defines how functions and physical elements are structured, and interfaces specify how modules interact. Together, these concepts form the foundation for understanding how early architectural decisions influence later module development.

2.1.1 Product Platforms

Product platforms have emerged as a key strategy for managing complexity, enabling product variety, and improving efficiency in product development. A product platform can be understood as a set of shared components, modules, or subsystems that form a common structure from which multiple product variants can be developed (Muffatto and

Roveda, 2000; Otto et al., 2016). This approach allows a company to reuse core elements across products while still offering differentiation to meet diverse customer needs. In this context, product families represent the set of related product variants offered to the market, while product platforms define the underlying structure that enables their efficient development.

A modular product platform can be understood as a platform architecture built on standardized modules and interfaces, enabling the development of multiple product variants through combinations of shared and variable elements (Li et al., 2016; Liu et al., 2010). Platform-based development is therefore not only a technical solution but also a strategic approach for coordinating product development over time. Since platform decisions affect several products, future variants, and multiple functions, they require long-term alignment between product strategy, technical architecture, and organizational governance.

A central characteristic of product platforms is their hierarchical structure. Platforms are typically organized across several levels, including the overall platform, modules, components, and associated parameters. Within this structure, modules can often be categorized as standardized modules, which are reused across many product variants, and flexible modules, which allow adaptation and differentiation. This distinction enables a balance between commonality and variety, where standardized elements support efficiency and scalability, while flexible elements enable responsiveness to specific market or technical requirements (Li et al., 2016).

The use of modular product platforms provides several advantages. It can reduce development time and cost by enabling reuse of existing modules rather than designing each product from scratch. It can also support mass customization by allowing different product configurations to be created through combinations of predefined modules. In addition, modular platforms can improve maintainability and lifecycle management, since individual modules can be updated, replaced, or upgraded without redesigning the entire system (Williamsson and Sellgren, 2021; Shamsuzzoha, 2011).

However, platform design also involves important trade-offs. Increasing commonality across products can improve efficiency and reduce cost but may limit the ability to differentiate products. At the same time, increasing flexibility and customization can improve market responsiveness but may introduce higher complexity and reduce economies of scale. Platform development therefore requires careful consideration of the balance between standardization and flexibility (Muffatto and Roveda, 2000; Merati et al., 2024).

In industrial contexts, product platforms are often implemented through structured approaches such as modular product portfolios. In heavy-duty vehicle development, for example, platforms are commonly built around standardized interfaces between modules, enabling different module variants to be combined into customized product configurations. These standardized interfaces ensure compatibility between modules while allowing variation in performance and functionality (Williamsson and Sellgren, 2021; Kadaba Jayaprakash et al., 2025).

2.1.2 Modularity and Product Families

In platform-based development, modularity is one of the main principles through which product platforms can be realized. Modularity enables products to be decomposed into distinct modules with defined interfaces, allowing shared elements to be reused across several product variants (Eppinger and Ulrich, 2020; Shamsuzzoha, 2011). Rather than focusing only on individual components, modularity supports structuring products in a way that reduces interdependencies and enables reuse across different products and product generations.

A key implication of modularity is its role in enabling product families. Instead of developing products as standalone systems, firms can design a set of related products that share common components while allowing variation in specific features. In practice, most products are part of product families, where modules and technical solutions are reused across multiple products and generations (Sankowski et al., 2021; Otto et al., 2016). This approach allows companies to leverage existing product knowledge and reduce development effort while still offering differentiated products.

The use of modular architectures in product families introduces a fundamental trade-off between commonality and variety. High levels of commonality, achieved through shared modules, enable cost reductions, economies of scale, and faster development cycles. At the same time, product variety is necessary to meet diverse customer requirements and adapt to changing market conditions (Merati et al., 2024; Liu et al., 2010). As a result, companies must balance internal efficiency with external responsiveness when designing modular systems.

This balance is not insignificant, as modularity does not remove complexity entirely. Instead, it often changes where complexity must be managed. While modularization can reduce dependencies within modules, it increases the importance of managing interfaces, module boundaries, and coordination between modules (Sankowski et al., 2021; Williamsson and Sellgren, 2021). If product variety is not carefully managed, it can lead to increased costs, reduced operational efficiency, and additional coordination requirements (Merati et al., 2024).

From a product development perspective, modularity therefore serves both as an enabler and a challenge. It allows companies to manage complexity and support product variety, but it also requires careful architectural decisions to avoid unintended consequences. These trade-offs become particularly important when defining product families and structuring systems for reuse across multiple products.

2.1.3 Product Architecture

Underlying both product platforms and modularity is the concept of product architecture. Product architecture describes how a product is structured and how its parts work together. It includes three main elements: the functional elements of the product, how these functions are assigned to physical components, and how the components interact through interfaces (Ulrich, 1995). This means that architecture is not only about dividing

a product into parts but also about understanding how these parts are connected and how they collectively fulfill system-level functions.

A key distinction within product architecture is between modular and integral architectures. In a modular architecture, there is a relatively clear mapping between functions and components, often close to a one-to-one relationship, and the interfaces between components are standardized and well-defined. This allows components to be developed and modified more independently, which can simplify both development and future changes. In contrast, integral architectures involve more complex relationships between functions and components, where multiple functions may be implemented in the same component or distributed across several components. In such cases, interfaces are tightly coupled, and changes to one part of the system often affect other parts (Ulrich, 1995; Eppinger and Ulrich, 2020). In practice, most products are not purely modular or integral, but rather a combination of both. The degree of modularity depends on how strongly components are connected and how clearly interfaces are defined. A more modular architecture typically reduces dependencies between components, while a more integral architecture increases interdependencies and system complexity (Eppinger and Ulrich, 2020; Williamsson and Sellgren, 2021).

Product architecture also has an important role as a shared representation of the system. In cross-functional development, different functions may interpret system boundaries, interfaces, and responsibilities differently. A common understanding of the product architecture is therefore important for aligning decisions across functions, especially in early phases where architectural choices define constraints for later module development (Kadaba Jayaprakash et al., 2025). When the architecture is not commonly understood, coordination between functions becomes more difficult, and decisions risk being interpreted differently depending on role or technical discipline.

Overall, product architecture provides the structural foundation for platform-based development. It defines how functions, physical elements, and interfaces are arranged and thereby influences how modularity, reuse, and variety can be managed across a product family.

2.1.4 Interfaces in Product Architecture

Interfaces are a fundamental element of product architecture, as they define how components and modules interact within a system. In the context of product architecture, interfaces specify the physical, informational, or functional connections between components, determining how elements exchange energy, material, or information (Ulrich, 1995; Eppinger and Ulrich, 2020).

In modular product architectures, interfaces play a critical role in enabling the decomposition of systems into independent modules. Well-defined and standardized interfaces reduce dependencies between modules, allowing components to be developed, modified, or replaced without significantly affecting the rest of the system (Eppinger and Ulrich, 2020). This supports the creation of flexible and scalable product structures, where mod-

ules can be reused across different products and configurations.

Within product platforms, interfaces are particularly important because they enable compatibility between shared and variable modules. By defining stable interfaces, companies can combine standardized modules with flexible elements to create a wide range of product variants. Interfaces therefore act as a key enabler of modular product platforms, supporting both commonality and variety (Li et al., 2016).

Another important aspect of interfaces is their stability over time. In platform-based development, interfaces are often defined early and remain relatively stable throughout the product lifetime. Stable interfaces provide a consistent structure that allows modules to evolve independently while still ensuring system compatibility. In contrast, unstable or poorly defined interfaces can lead to increased system complexity, as changes in one part of the system may propagate and require adjustments in other components (Li et al., 2016; Schuh et al., 2017).

Interfaces also have an organizational role. They define where responsibilities meet and where coordination between teams is required. In this sense, interfaces can be understood not only as technical connections but also as coordination mechanisms that enable parallel work, clarify dependencies, and support communication between functions. This is consistent with the view of interfaces as contracts between modules, defining both physical orientation and flows of information, matter, and energy (Williamsson and Sellgren, 2021).

In summary, interfaces serve as a central mechanism for structuring product architectures. By defining how modules interact, they enable modularity, support product platforms, and provide a foundation for managing both technical and organizational complexity in product development.

2.2 Architectural Decisions

Within the development of complex products, such as within the OEM industry, architectural decisions provide the foundation that defines the physical topology, its primary energy, and the core structural platforms that ultimately will dictate the future products for a 10- to 20-year product life cycle (Ulrich, 1995). These decisions are foundational since they dictate the vehicle's capabilities and limitations. Since production of heavy vehicles demands heavy investments in terms of production tools, stamping dies, crash testing, etc., the reversibility of these decisions often comes with an economic burden.

To separate architectural decisions from technical design at the component level, an analysis of the decision's impact on the system as a whole is required. A central architectural element is the definition of the vehicles' so-called "hardpoints"; these points determine the physical space claims, for example, the cooling systems' dimensions and positioning, dictating the placement of the engine and vice versa, ultimately affecting the cab's positioning and the vehicle's aerodynamic properties(Magnusson and Pasche, 2014). The architectural decisions establish the framework for module development. Compared to the development of personal vehicles, where they are commonly created for one single

purpose, commercial vehicles, on the other hand, are used in a wide range of purposes, creating a need for variety. A well-defined architecture enables this with common interfaces that allow developers to create variety for different applications (Sköld and Karlsson, 2013).

In addition to the physical aspects, the architecture directly affects the weight distribution and its technical total weight (Gross Vehicle Weight Rating, GVWR). These parameters are strictly regulated by national and international road traffic regulations, making certain architectural decisions related to this of great importance, since the wrong decision could in reality lead to not meeting legal requirements in the long terms. A unique aspect for the architecture of a truck is the bodybuilding aspect, where external firms mounts their respective equipment such as a crane or a garbage collector. This leads to specific demands on the architecture where standardized interfaces (mechanical or electrical) needs to be accounted for in order to be competitive and maintain flexibility in the market.

2.3 Early Decision-Making

Within the OEM industry, the most critical phases of the product's lifecycle is during concept generation, product planning and product definition (Tan et al., 2017),(Li et al., 2013). The importance of these early decisions are often illustrated by the "80-20 rule", that implies that the main part of a products costs are locked in way before the manufacturing even begins (Tan et al., 2017). Studies have been given broad support, showing that 80% of a product's manufacturing cost, and at least 75% of the total lifecycle cost is determined already during concept generation (Tan et al., 2017). As the development process continues, the freedom of changing hardware design drastically decreases, as well as the possibility to have an influence on cost reductions regarding the lifecycle of the product quickly vanishes (Tan et al., 2017).

The financial consequences of misinterpretations of the future market made in the early phases of development can be greatly negative (Tan et al., 2017),(Verganti, 1997). Research shows that the frequency of these misinterpreted decisions is of natural reasons at its highest during early conceptual phases, with a fault frequency at 59% before concept freeze (Tan et al., 2017). This is due to the high degree of uncertainty, while only 22% fault frequency occurs in later stages (Tan et al., 2017). In cases when decisions of this nature need to be changed at a later stage, the cost of rework is exponential (Verganti, 1997). Decisions that need to be corrected due to an early design decision demand thirteen times more rework than changes made after concept freeze (Tan et al., 2017). This is heavily affected due to the reason that early architectural decisions often cannot be verified fully until a later stage where testing occurs.

To handle the enormous complexity, while also promoting variety within the truck-manufacturing industry, platforms made from several interlinked modules are used (Li et al., 2013). An effective hardware architecture ensures that components within a specific module show strong internal coupling, while the relations between module to module remain intentionally weak and decoupled, allowing for adaptability (Li et al., 2013).

This decoupling is achieved by establishing standardized interfaces that govern the connections between modules, making it possible for the overarching platform to main-

tain a stable architecture while also adapting to variety in customer demands (Li et al., 2013).
During early phases, an organization should consider categorizing their overall modules in specific module types; base modules, that provide the foundational functionality with relatively static structures (Li et al., 2013), flexible modules, that can be adjusted to deliver adaptable performance depending on needed variety depending on customer demands. Finally individualized modules that provide specific, differentiated market demands (Li et al., 2013). Establishing these modules and their standardized interfaces early in process is vital for creating a flexible platform, that supports further development and design work, without requiring total redesign of the underlying hardware (Li et al., 2013).

2.3.1 Frontloading and Systemic Learning

To effectively deal with the high stakes of early architectural mapping, successful OEM companies utilize "front-loading" (feed-forward planning or left-shifting) (Verganti, 1997). Front-loading is the active anticipation of downstream constraint and possibilities, such as manufacturability, ease of assembly and shifts within technology. Weight is put on doing this as early as possible in the process (Verganti, 1997). By bringing downstream information into the concept generation phase, the development teams ensures that the initial architectural decisions already take into account the realities of the future product lifecycle, which could lead to less rework and changes in late stages (Verganti, 1997).

The primary issue with front-loading within hardware development is the enormous uncertainty that is in the initial stages of a project (Verganti, 1997). To hinder that front-loading only becomes a sterile practice where overspecification is done due to uncertain parameters, OEM companies needs to rely on systemic learning (Verganti, 1997). Systemic learning is the capability for pre-project teams to leverage knowledge from previous projects, to as exact as possible predict an early architectural decision such as how a specific chassi layout or an interface standard will affect the whole downstream lifecycle (Verganti, 1997).

Furthermore, because not every downstream constraint can be anticipated, planned flexibility needs to be embedded in early decisions (Li et al., 2013). By deliberately designing a modular structure with standardized interfaces, OEM companies ensures that unpredictability with late adjustments or where technical innovations are demanded, can be isolated to a specific module and thereby not affecting the system as a whole (Verganti, 1997). This planned flexibility improves the companies chances of coping with inevitable late changes, decreasing the consequences and successfully bridging the gap between early decisions and final production (Verganti, 1997).

To successfully implement systemic learning, the company must realize that effective front-loading to a high degree is dependent of systemic knowledge, that specific individuals in pre-project teams possesses (Verganti, 1997). Systemic knowledge is defined as an individual's specific ability to early detect how their own decisions will affect the following

phases in the product life cycle. Consequently, the early concept and planning phases does not require generalists, but rather specialists who possess a deep understanding of the systemic impacts of their choices (Verganti, 1997). While cross-functional collaboration is often emphasized within product developments, a simple increase in the team's size without these knowledgeable specialists can turn early involvement into a fruitless and sterile exercise.

Crucially, systemic knowledge is not built solely by developing one single product, but has to grow consciously through structured learning from previous knowledge. Successful companies foster this capability by ensuring that the same experts who make the initial architectural decisions are also actively involved in the projects closing and evaluation phases. By participating in these phases, specialists can compare their initial forecasts with the actual results, analyze any unforeseen problems and learn from the deviations. This continuous cycle of evaluating previous decisions against end results is how systemic knowledge is being refined, ensuring that architectural decisions are based on expertise rather than guesses (Verganti, 1997).

2.3.2 Path Dependency

Initial decisions under uncertainty could lead to path dependency. This is characterized by the process of being less flexible until a change to another option is close to impossible (Camuffo et al., 2025). When an organization begins the development of a new product platform, usually it's the first project that dictates the playing rules for the future platform. Despite the intention of creating a common architecture for a broad family of derivatives, the initial team tends to optimize the design for their own short-term goals and time plans (Camuffo et al., 2025).

One of the most famous cases of path dependency is the QWERTY keyboard. When Christopher Sholes developed his early typewriter in the 1860s, it needed some major corrections, most notable was that mechanical type arms adjacent to each other would get stuck when the user wrote too quickly. To solve this issue, the QWERTY layout was designed, specifically to decrease the probability of mechanical type arms to get stuck. This early architectural decision was fully rational according to that time's technical limitations but quickly dictated a long-lasting standard. It became definite once the users, such as schools, started to standardize their education based on the QWERTY layout, which drastically increased the cost of changing system. When the technical development later eliminated the problem with type arms getting stuck, the solution was already "locked in", due to an infrastructure that made it impossible for the market to change to a quicker more ergonomic layout (Camuffo et al., 2025).

This example highlights a central issue where decisions in the initial phases sets the conditions for a long time ahead. The creation of the first iteration of a platform, such as in the case with Scholes typewriter often have short term goals and incentives that differs to the teams that later on will continue maintaining and developing (MacCormack and Sturtevant, 2016). As Verganti suggests, it's crucial that decisions in early phases takes in account for future possibilities and limitations, otherwise the organization is forced into massive amounts of rework once the issues with the platform becomes apparent in later

stages (Verganti, 1997).

2.3.3 Design Change Propagation

In complex product development, design changes are rarely isolated to a single component or module. Since products consist of interconnected systems, a change in one part of the product may create a need for additional changes in other parts. This phenomenon is commonly described as "design change propagation" (Clarkson et al., 2004). In complex products, a single modification can therefore develop into a chain of subsequent changes that propagate through different parts of the system (Clarkson et al., 2004).

The extent of change propagation depends on the structure of the product and the dependencies between its elements. Functional couplings between components, subsystems, and interfaces can create ripple effects, where a change in one system element forces changes in other functionally connected elements (Wessels and Pretorius, 2013). Such ripple effects can have negative consequences for development cost and schedule, especially when they occur late in the development process or during system integration (Wessels and Pretorius, 2013). Therefore, the architecture of the system plays an important role in limiting the impact of future changes (Wessels and Pretorius, 2013).

Change propagation is particularly relevant in modular platform development, since modularity aims to manage complexity by defining module boundaries and interfaces. Well-defined interfaces can help contain changes within specific modules, while poorly defined or highly coupled interfaces may allow changes to spread across the system (Clarkson et al., 2004; Wessels and Pretorius, 2013).

Since engineering changes cannot be fully eliminated, the goal is not to avoid all future changes but to design architectures that reduce the likelihood and impact of change propagation (Hamraz et al., 2012). This implies that early platform decisions should consider system dependencies, interface stability, and areas where flexibility may be needed over time.

2.3.4 Requirements as a Basis for Modular Platform Decisions

In modular platform development, early architectural decisions should be based on an understanding of customer needs, market requirements, and expected product variety. Platform architecting can be described as a process moving from market segments and customer needs through system requirements toward module boundaries, common platform elements, and architectural down selection (Hölttä-Otto et al., 2013). This means that the intended variety of the product family should be understood before defining which modules or solutions should become common across the platform (Hölttä-Otto et al., 2013). In other words, common solutions should be derived from common or compatible requirements, rather than from the needs of a single initial product variant.

This is important because modular product architectures are used to balance product differentiation with economies of scale through standardization (Schuh et al., 2014). A

structured development process should therefore begin by identifying requirements from both external conditions, such as market and customer needs, and internal conditions, such as product structure, production processes, and complexity drivers, (Schuh et al., 2014). These requirements can then be translated into a requirement specification that supports decisions about standards, modularity, and interfaces (Schuh et al., 2014). In this way, the architecture can include stable common elements while still leaving flexible areas for future product development (Schuh et al., 2014).

However, standardization must be evaluated against the variation the platform needs to support. Requirements that are stable and shared across several product variants can provide a basis for common platform solutions, while requirements that differ between markets, applications, regulations, or customer use cases may need to be handled through flexible or variant-specific modules (Hölttä-Otto et al., 2013; Schuh et al., 2014). Since requirements also may change over time due to technological development or changing customer needs, premature standardization can reduce the organization's ability to respond to future variation (Magnusson and Pasche, 2014). Therefore, early platform decisions need to distinguish between what should be common, what should remain flexible, and what should be developed as variant-specific.

2.4 Set Based Concurrent Engineering (SBCE)

The traditional product development often builds upon Point-Based Design (PBD), where engineers early in the process chooses one main concept being iterated and refined all the way up to production (Kerga et al., 2014). This way of working easily creates an issue (path dependency), where the first project quickly determines the architecture, before requirements and physical limitations are known (Verganti, 1997). Locking one solution to early creates a risk for faulty assumptions, often identified late in the development process leading to a costly rework (Tan et al., 2017). Research and other industrial studies shows that design decisions forced before the concept is mature for freezing (concept freeze), tends to be incorrect and demands on average 13 times more rework (Tan et al., 2017). The fact is that late corrections in an already locked architecture can stand for up to 86 percent of the total rework cost in a project (Tan et al., 2017). If unforeseen issues or new customer demands occurs in a locked PBD, the consequences quickly spreads to adjacent systems resulting in delays, incompatibility and poor product quality (Rebentisch and Genta, 2017).

To handle the uncertainty and avoid the risks with an early freeze, Set-Based Concurrent Engineering (SBCE) is adopted. Where multiple design alternatives (sets) are developed and explored in parallel (Raudberget, 2010). Instead of trying to choose the winning concept, the design space is being kept open, and the teams work in a process where they systematically eliminate worse and impractical alternatives, based on continuous tests and data (Raudberget, 2010). This principle, often formulated as establishing feasibility before commitment, minimizes project risk significantly (Kerga et al., 2014). The consequences of wrongly eliminating the worst alternative are very low in comparison with building the whole architecture on an unevaluated "best" concept that later crashes (Raudberget, 2010).

One of the most central tools to evaluate and understand the different sets of architectures, without having to build complete prototypes, is the use of Trade-off Curves (ToC) (Araci et al., 2022). These curves visualize the inevitable compromises and relations between critical design parameters, such as the tradeoff between cost, weight, lifespan, and performance (Kerga et al., 2014; Araci et al., 2022). Using these curves, decisions can be based on boundary curves highlighting historical data, simulations, and physical tests, making it easier to define what is possible for a specific architecture. This front-loads the process with a solid knowledge base that helps engineers to clearly see where the boundaries are, making decisions based on facts and knowledge rather than guesses (Araci, 2017).

The ability to delay decisions and keep several solutions open as long as possible has a strong relevance to modularity and the development of platform architectures (Verganti, 1997), (Toche et al., 2020). SBCE enables the creation of flexible modules, which are defined by modules that are based on a parametric and customizable architecture made to manage and integrate several unique performance steps and technical solutions on the same platform (Li et al., 2013). By early and systematically identifying which interfaces need to handle uncertainty and thereby defining these interfaces with margins that hold several different sets, the company establishes a robustness in their different concepts (Raudberget, 2010). This in practice means that if a technical issue suddenly arises or if new information about the market occurs, a specific module can be adjusted or swapped against an alternative solution without creating a cascade of rework that forces an expensive redesign of the whole architecture (Verganti, 1997). Thus, SBCE identifies the interfaces flexible enough to withstand future, not yet established, configurations (Toche et al., 2020).

2.5 Cross-Functional Collaboration and Coordination

In the industry, there are certain technical disciplines needed to achieve a functioning and coherent holistic perspective (Eppinger and Ulrich, 2020), (Hirshorn et al., 2017). This means that the architectural decisions concerning a product, for example, the degree of modularity and the overarching structure of the organization, are deeply connected and dependent on one another (Eppinger and Ulrich, 2020).

Historically, manufacturers have relied on functional organizational structures, where specialized teams are grouped within their specific discipline (Eppinger and Ulrich, 2020). Such a structure tends to create strong organizational silos, where a lack of concurrent engineering and a shared vision exists. Departments focus on their own priorities and goals in front of the whole project or the product's success (Eppinger and Ulrich, 2020; Verganti, 1997). A direct and possibly devastating consequence of this is "over the wall engineering." This means working in an extremely reactive way, where an upstream department (e.g., construction) finishes their design job and "throws it over" to the next downstream department (e.g., manufacturing) to realize their vision, without any previous coordination (Verganti, 1997), (Eppinger and Ulrich, 2020). Out of this, the result could be that potential issues are discovered in a late stage, forcing costly and time-consuming reconstructions, ultimately delaying market introduction (Verganti, 1997).

To counteract silo thinking and instead work in a proactive (front-loading) way, integrated cross-functional cooperation is needed. This is best achieved by early involvement of different departments such as marketing, design, production, and purchasing (Verganti, 1997). Since platform and module development often cause organizational tensions and demand a high degree of coordination (Boer and Persson, 2015), the cross-functional collaboration is often strengthened by physical co-location. This accelerates problem-solving cycles, eliminates information barriers, and facilitates the important transferring of silent knowledge (tacit knowledge) between engineers (Muffatto and Roveda, 2000). However, management needs to balance this in their everyday work. Keeping the beneficial aspects of a coordinated project focus and still not disabling the engineers from using their expertise by staying too long in isolated project teams (Muffatto and Roveda, 2000).

A fundamental theoretical way of dealing with cross-functional teams is to treat product development as an information processing system, where departments continuously process input data to final design specifications (Eppinger and Ulrich, 2020). To mathematically and visually optimize the flow of information, models such as the design structure matrix (DSM) is being used. This model maps the information flow between tasks and differentiates activities that are sequential, parallel, or coupled (Eppinger and Ulrich, 2020). Coupled tasks mean mutual dependencies, demanding frequent interaction and rapid iteration between engineers (Eppinger and Ulrich, 2020). Through an early and thorough specification of interfaces, the possibility to restructure the platform still exists. The tasks can therefore be decoupled, making it possible for different departments and teams to do their work both autonomously and parallel to other departments/teams. This could drastically shorten the development time (Eppinger and Ulrich, 2020; Boer and Persson, 2015).

Furthermore, PLM systems (Product Lifecycle Management) and visual interface diagrams enable engineers from different domains to have an overall understanding of the system in order to quickly determine where conflicts in the design can occur (Bruun et al., 2015).
In environments where several brands work collaboratively, information exchange is not sufficient, since platform development makes enormous demands on knowledge integration. In multi-branded OEM structures (several brands share the same platform), architectural knowledge is usually brand-specific to a large extent and embedded in previous ways of working (Sköld and Karlsson, 2007). To avoid conflicts with the platform, not only learning is demanded, but also "unlearning" from previous product-specific experiences that lack relevance in an integrated multi-brand perspective (Sköld and Karlsson, 2007). Successful transferring of knowledge heavily builds on systemic learning, meaning the ability to gather unstructured feedback and learnings and proactively feed this information into the generation of new platform concepts (Verganti, 1997).

For all these dimensions to work coherently, strongly formalized decision structures are demanded. Systems engineering provides the holistic framework necessary to make informed decisions so that neither department is allowed to dominate the end product (Hirshorn et al., 2017). In the extended matrix organization, this is realized through formal gov-

ernance bodies such as Configuration Control Boards (CCBs) which assess requirements changes, evaluate the effects of design changes, and deal with technical risks (Hirshorn et al., 2017).
The transfer to modular architectures and centralized PLM environments creates a shift in power in the organization's decision-making structure. Decisions concerning the design of components gradually transition away from classical, decentralized engineering domains and are moved into the hands of module managers, who enforce strict adherence to standard interfaces and design rules (Bruun et al., 2015).

2.6 Accountability and Transparency

This section reviews platform governance, which is a critical area dealing with the complexity that arise when technical decisions crosses over several function- and product boundaries. Governing this often concerns balancing the need for a standardized architecture with market demands on unique functions, which puts high demands on the organizations structure as well as role distribution. The section is divided into two subsubsections, highlighting governance from different perspectives. First in section 2.6.1 weight is put on clear role distribution in cross-functional teams and discusses responsibility matrices with a focus on the RACI-framework. Then, section 2.6.2 examines the need for strong leadership structures and centralized governance at high level platform development, where traditional project leader roles is contrasted against product owners with mandate and authority to make strategic decisions, protecting the long term integrity of the platform.

2.6.1 RACI and Responsibility Matrices

In project-based and cross-functional organizations, complexity often arises due to double ways of reporting or unclear roles (Koivisto, 2025). This complexity can lead to uncertainty among employees regarding responsibilities and about who really owns a specific decision. Research emphasizes that when roles are not clearly defined, the organization invites unnecessary and unproductive conflicts as a consequence of the ambiguity that occurs (Koivisto, 2025).
One of the primary problems in cross-functional teams is role confusion. This confusion is often founded in an imbalance between three aspects of a role: role conception (what the person thinks their job is), role expectation (what others in the organization expect the person to be responsible for), and role behavior (what the person actually does in his/her daily work) (Smith et al., 2005). When these three do not match with one another, the organization often shows symptoms such as delayed decisions, an imbalanced workload, a "we against them" attitude, and employees blaming each other when tasks are not done (Smith et al., 2005).

To handle these problems and systematically identify ambiguity in processes, responsibility matrices are often used, where RACI (Responsible, Accountable, Consulted, Informed) is the most established one. The RACI framework forces the organization to openly discuss and clarify expectations and decision rights (Smith et al., 2005). The core in this

model and the most critical for cross-functional teams is the strict separation between operational execution and formal decision ownership (Koivisto, 2025).

Responsible
One or several individuals acting as "the doer" that actually execute the task or implement the decision (Smith et al., 2005). This form of responsibility is an obligation to finish a task and can be shared with several team members to promote collaboration and joint implementation.

Accountable
This role is often described with the expression "The buck stops here" (Smith et al., 2005). The one who's accountable and ultimately responsible for the activity's result or when decisions are made. In this "yes" or "no" authority is included as well as the right of veto (Smith et al., 2005). In contrast to the operative execution, this accountability can never be shared, it has to be one individual who carries the responsibility for when a decision is successful or a failure.

When the border between responsible and accountable is erased in cross-functional teams, delays and inefficiency often occur, since shared responsibility in decision-making tends to mean that no one takes responsibility ("Joint responsibility ends up being no one's responsibility") (Koivisto, 2025). By consequently using responsibility models and clarifying who has an accountable mandate, organizations can limit the time-consuming uncertainty that decision-making often has (Koivisto, 2025). In addition, research shows that in newly formed and complex teams, where team members do not know each other, clearly defined roles and mandated work build trust, helping the team to perform more effectively (Koivisto, 2025).

2.6.2 Heavyweight Product Managers

Decision-making within platform development is well known to be more complex than traditional individual product development since the decisions cut over several function and product boundaries (Magnusson and Pasche, 2014). Development of platforms consistently consists of maintaining a standardized architecture and accommodating customers' demands on unique functions. In traditional and functionally governed "weak" matrix organizations, projects are often led by coordinated project leaders called "lightweight project managers" that firsthand act as coordinators without any formal power to force decisions across function boundaries (Muffatto and Roveda, 2000). This often leads to crucial cross-functional platform decisions being delayed, sub optimized, or entirely driven by the local interests of the individual functional departments.

To successfully drive platform development, a strong decision and leadership structure is needed. Research shows that the demands on leadership and organizational changes drastically depend on the extent of the platform development(Sköld and Karlsson, 2013). While incremental or limited modular development often can be handled by lightweight structures, instead, more profound platform development demands so-called "heavyweight teams." A heavyweight product manager does not only have a strong cross-functional

team but also the formal mandate and the technical authority to make strategic platform decisions.
Research shows that strong platform ownership to a high degree needs to be driven top-down through centralized governance and the active involvement of senior management (Magnusson and Pasche, 2014). Since platform decisions involve complex strategic trade-offs between different business areas, the scope needs to be lifted from the project level to a more holistic level to avoid expensive chain effects caused by unplanned changes in the architecture. By assigning strong technical product leaders with a full mandate, the company ensures that decisions are moved closer to the actual engineering work while maintaining the integrity of the platform when solving isolated, short-term projects.

2.7 Organizational Communication

2.7.1 Semantic Noise

In complex organizational environments, where parties use common terminology but hold fundamentally different understandings of those terms, it acts as a significant barrier to coordination and integration. When terms are used loosely without a common definition, an environment is open for what researchers call "merely verbal" disputes (McGrath and Whitty, 2015). These are conflicts arising not due to any particular disagreement but due to the parties unknowingly using ambiguous terms or phrases in different senses. Philosophers such as John Locke have long argued that a big part of the world's conflicts comes from the meaning of words, and if these words were defined more clearly, the conflicts would vanish (McGrath and Whitty, 2015).
In professional communication, the lack of a "common meaning" between the sender and receiver of the message acts as a semantic noise disrupting the flow of information (Salsabilla et al., 2025). This definitional confusion is often amplified by the diversity of backgrounds in cross-functional environments. Differences in cultural background, level of education, age, and communication habits all affect to a high degree how different individuals interpret the professional language (Salsabilla et al., 2025). This imbalance increases the risk for common concepts being misunderstood and leads to messages being interpreted in several different and sometimes contradictory ways.

When conceptual discrepancies arise, foundational terms lose function as effective coordination mechanisms. Instead of creating consensus, the frequent use of ambiguous or multi-interpretable language negatively impacts message understanding, team coordination, and decision-making processes (Salsabilla et al., 2025). This lack of consensus can lead to delayed completion of tasks, double work, failures in negotiations, and unnecessary internal conflicts that only originate in faulty assumptions.
In the end, there is not only language practice to eliminate confusion but also a strategic necessity for an organization's management (Salsabilla et al., 2025). Reaching clarity and a common understanding of core terms makes it possible for the organization to avoid spending time, resources, and money unnecessarily, instead promoting a more connected, productive, and cooperative work culture.

2.7.2 Decision Traceability and Design Rationale

In complex product development, decisions are not only important because of the solution that is selected but also because of the reasoning behind that selection. Design rationale can be understood as the explanation of why a product, system, component, or part of a design is developed in a certain way, including the reasoning, trade-offs, and decision-making that shaped the design (Regli et al., 2000). In engineering design, this rationale includes reasons behind a design decision, the justification for the decision, the alternatives considered, and the argumentation that led to the final decision (Rockwell et al., 2010). This means that decision documentation should not only capture the final outcome but also the basis on which the decision was made.

This is particularly important in platform-based product development, where decisions may affect several product variants, future projects, and later redesign activities. If only the final decision is documented, future teams may know what was decided but not why it was decided. This can make it difficult to understand whether the original reasoning is still valid when requirements, technologies, market needs, or system constraints change. For this reason, design rationale can support knowledge reuse by making previous decisions understandable and by helping later designers access the assumptions, criteria, alternatives, and evaluations that shaped earlier choices (Rockwell et al., 2010).

Design rationale can be captured from several perspectives. Shipman and McCall distinguish between argumentation, documentation, and communication perspectives. The documentation perspective focuses on recording what decisions were made, when they were made, who made them, and why, while the communication perspective focuses on capturing naturally occurring communication between project members (Shipman III and McCall, 1997). This distinction is relevant because design communication may occur in meetings, e-mails, discussions, or informal exchanges, but such communication is not necessarily structured or easily retrievable later. Therefore, communication alone does not ensure decision traceability unless the relevant rationale is captured and organized in a way that can be retrieved and understood over time.

A central challenge is therefore to balance capture and retrieval. More informal communication is often easier to capture, but harder to search and reuse, while more structured documentation can improve retrieval but may require additional effort from designers. Design rationale systems therefore need to support both the capture of design reasoning and its later retrieval, for example, by linking decisions to issues, alternatives, criteria, evaluations, and supporting information (Regli et al., 2000). Rockwell et al. similarly argue that design decisions can be documented through concepts such as design issues, alternatives, criteria, evaluation information, preferences, and outcomes, which makes the rationale behind decisions more explicit and reusable (Rockwell et al., 2010).

However, design rationale should not be understood as a requirement to document everything. Capturing rationale can create cognitive, capture, retrieval, usage, and organizational limitations, especially if the documentation process becomes too intrusive or if the benefits mainly appear for future users rather than the current design team (Horner and Atwood, 2006). Therefore, the purpose of decision traceability should be to make

critical decisions understandable over time, rather than to create exhaustive documentation of all design discussions. In early architectural decision-making, this suggests that particular attention should be given to decisions with high platform impact, including the assumptions, trade-offs, rejected alternatives, accepted constraints, and expected future implications behind them.

3
Methodology

This study followed a qualitative and iterative research approach, supported by a semi-systematic literature review and empirical data collected through three sequential phases: contextual observations, a survey, and semi-structured expert interviews. The methodology was designed to progressively build an understanding of early-stage architectural decision-making in large-scale platform-based product development. The literature review established the theoretical foundation for the study, while the empirical data collection moved from contextual exploration to a broader survey of organizational perceptions and finally to targeted interviews with respondents holding relevant knowledge of architectural work, technical decision-making, and cross-functional collaboration. Together, these components enabled the study to connect theoretical perspectives with empirical insights from the organizational context.

3.1 Literature Review

The literature review in this study followed a semi-systematic approach, inspired by established review methodology guidelines (Snyder, 2019; Jahan et al., 2016). The purpose of the review was not to conduct a fully systematic review but to identify, synthesize, and critically assess theoretical perspectives relevant to early-stage decision-making and platform-based product development.
A semi-systematic approach is considered appropriate when the research field is conceptually broad and interdisciplinary, involving contributions from engineering design, product development, and strategic management research (Snyder, 2019). In this study, the literature review served two primary functions. First, it established the theoretical foundation regarding early decision-making and front-loading in product development. Second, it supported the development of interview protocols by identifying central theoretical concepts and potential areas of organizational misalignment.

The selection of these areas was guided by the overall purpose of the study, which is to understand which decisions and information are important when modular platform development is initiated under uncertainty. Therefore, the review needed to cover both the technical foundations of platform-based development and the organizational conditions that shape how early decisions are made, coordinated, and communicated.

The review focused on these areas:

1. **Early-stage decision-making and front-loading**, including research on cost commitment, design freedom over time, and proactive problem-solving in product development (Tan et al., 2017; Thomke and Fujimoto, 2000).
2. **Early integration of downstream knowledge**, addressing how technical and manufacturing considerations can be incorporated early to reduce late design changes.
3. **Platform-based and modular product development**, including governance, architectural interfaces, and strategic coordination challenges in large organizations (Baldwin et al., 2003; Robertson and Ulrich, 1998)

A structured but iterative search strategy was applied using academic databases such as Scopus, Web of Science, and Google Scholar. Search terms included combinations of "front-loading," "early decision-making," "design for manufacturing," "platform development," "modular platform architecture," and "cross-functional collaboration." "Boolean operators (AND/OR) were used to refine the search process in accordance with recommended search procedures (Jahan et al., 2016).

Inclusion criteria consisted of:

- Peer-reviewed journal articles
- Studies within engineering design, product development, and strategic management
- Research addressing early development phases, platform governance or modular integration

Exclusion criteria included:

- Studies focused solely on software development processes
- Purely operational logistics studies without connection to early development decisions
- Non-academic opinion-based publications

The literature review process was iterative, meaning that initial findings led to refinement of search terms and thematic focus, consistent with semi-systematic review principles (Snyder, 2019). The emerging theoretical themes were subsequently used to support the survey design, structure the interview guide, and guide the interpretation of the empirical findings.

3.2 Iterative Data Collection

The empirical data collection followed a phased and iterative qualitative approach. Rather than collecting all data simultaneously, the process was structured in three consecutive phases: contextual groundwork and observations, a survey, and semi-structured expert interviews. Insights from each phase informed the next, enabling progressive refinement of the research focus, survey design, interview guide, and sampling strategy. This is consistent with qualitative research design principles, where data collection and analysis may develop iteratively as understanding of the studied context deepens (Creswell, 2009).

The phased design supported both contextual understanding and analytical depth, allowing the study to move from broad exploration of the organizational setting to a wider mapping of perceptions and finally to targeted expert inquiry.

3.2.1 Phase 1: Contextual Groundwork and Observations

The first phase consisted of informal observations and participation in internal meetings, educational sessions, and recurring discussions related to platform development activities. The purpose of this phase was to develop contextual familiarity with the organizational setting, terminology, ongoing initiatives, and decision-making processes before conducting the survey and formal interviews.

During the initial phase of the study, the authors participated in three recurring weekly meetings related to platform architecture work: a way-of-working meeting, a technical meeting, and a module interface development meeting. Participation in these meetings varied between approximately three and eight weeks, depending on the meeting content and relevance to the study. In addition, weekly meetings with the company supervisors were held throughout the full thesis period. The authors also participated in internal educational sessions during the first four weeks, covering topics such as data documentation, front-loading, and the product development process. Informal discussions with colleagues in the department also took place throughout the study period.

The observations were mainly passive, where the authors followed discussions to understand how platform architecture work was carried out in practice. In some meetings, the thesis work was included as a discussion point, for example, to receive input on how survey questions could be formulated to be understandable and relevant for the organization. The observed settings included internal platform architecture meetings, technical discussions, project-related meetings across organizational boundaries, educational sessions, and informal conversations.

Through field observations and informal discussions, preliminary insights were gathered regarding:

- Organizational structure and roles
- Internal communication patterns
- Ongoing development initiatives
- Terminology and process logic

Meeting notes were taken during the observations and shared between the authors. These notes were used to document recurring topics, questions, and potential problem areas. The observations also helped the authors identify areas that were relevant to explore further in the survey and interviews. These included how key terms were used in practice, how technical decisions were discussed and communicated, and how different roles became involved in early architectural work. However, the observations were not used as a separate empirical result category. Instead, they served as contextual background and informed the design of the survey and interview guide, while also supporting the inter-

pretation of the later empirical findings.

Qualitative research emphasizes the importance of building familiarity with the organizational context, which supports more focused and relevant data collection in later stages (Creswell, 2009).

3.2.2 Phase 2: Survey

Following the initial observations, a survey was conducted to obtain a broader understanding of how employees perceived key concepts related to platform-based product development. The survey focused on how respondents understood architecture, platform, and architectural strategy as well as how they experienced cross-functional collaboration and decision-making in early development phases. In this sense, the survey had both an exploratory and descriptive role, as it was used to map organizational perceptions while also preparing for the subsequent interview phase (Slattery et al., 2011).

The questionnaire was developed based on insights from the initial observations, relevant literature, and discussions with the company supervisor. This is in line with survey design principles, where the objective of the survey should guide the structure, type of data collected, and subsequent analysis (Slattery et al., 2011). The questionnaire consisted of approximately 20 questions and included background questions, open questions regarding the definitions of architecture and platform, multiple-choice questions, Likert-scale statements, and open-ended questions. Closed questions were used to enable comparison across respondents, while open-ended questions allowed respondents to elaborate on specific experiences or provide examples that could not be captured through predefined alternatives (Slattery et al., 2011).

Several statements were assessed using a six-point Likert scale ranging from strong disagreement to strong agreement. The scale was chosen to encourage respondents to take a position rather than selecting a neutral midpoint. The main themes covered in the survey were understanding of architecture and platform, awareness of architectural strategy, decision-making and governance, cross-functional collaboration, and causes of delays in technical decisions.

The survey was distributed through Microsoft Forms by the company supervisor to approximately 120 employees in departments and roles connected to, or collaborating with, the platform architecture team, of which 25 respondents completed the survey. The sampling approach can be described as convenience sampling, although the distribution aimed to include respondents from different roles, functions, levels of experience, and geographical locations, including Europe, North America, and Brazil. Sampling is an important aspect of survey design, as the interpretation and applicability of survey results depend on who responds and how the sample is selected (Slattery et al., 2011).

Participation was voluntary, and respondents were informed about the purpose of the study and that the results would be used in the thesis. The responses were treated anonymously. Respondents were also given the opportunity to indicate whether they

were willing to participate in a follow-up interview, which supported the selection of interviewees for Phase 3.

Before distribution, the questionnaire was reviewed and tested together with representatives from the department and the company supervisor. Based on this review, some questions were reformulated, and additional questions were added to improve clarity and relevance. The estimated response time was also tested. Pilot testing is commonly used to identify unclear questions, improve the questionnaire, and reduce the risk of misunderstanding before the final survey is distributed (Aithal and Aithal, 2020).

The quantitative survey responses were analyzed using descriptive statistics, primarily through frequencies, mean values, and visual summaries. The open-ended responses were reviewed thematically to identify recurring patterns and examples related to the main themes of the study. The survey results were used both as an empirical source in itself and as input for preparing the subsequent interviews. However, due to the limited number of responses and the non-random sampling approach, the survey was not intended to provide statistically generalizable results. Instead, it served as a way to identify organizational patterns, support the interpretation of observations, and guide the next phase of data collection.

3.2.3 Phase 3: Semi-structured Expert Interviews

Following the survey, semi-structured expert interviews were conducted to gain a deeper understanding of how architectural decisions, collaboration, and decision-making processes are experienced in practice. While the survey provided a broader overview of organizational perceptions, the interviews enabled more detailed explanations and examples from individuals involved in, or affected by, platform-related decision-making. Qualitative interviews are particularly useful when the aim is to understand participants' experiences, interpretations, and the meaning they assign to a specific organizational context (Sutton and Austin, 2015).

The interviews were designed as semi-structured expert interviews. This format was considered suitable because respondents had specific knowledge of platform development, architectural work, technical decision-making, or cross-functional collaboration. Expert interviews are commonly used to explore a specific field or action through respondents with specialized knowledge, organizational positions, or experience in relevant decision-making processes (Döringer, 2021). The semi-structured format also allowed the interviews to follow a common thematic structure while still leaving room for respondents to elaborate freely and for the interviewers to ask follow-up questions. This aligns with interview approaches where a guide is used to support open but thematically structured interviewing (Döringer, 2021). The full interview guide can be found in the appendix.

The interview guide was developed based on the research question, the theoretical framework found through the literature review, the initial observations, and the survey results. It covered themes related to the respondents' role and professional background, understanding of architecture and platform, architectural strategy and alignment, communi-

cation of decisions, decision ownership and process, causes of delays, cross-functional collaboration, and potential improvement areas. The interviews typically started out with broad questions, allowing respondents to answer freely before more specific follow-up questions were asked. This structure was chosen to capture both comparable data across interviews and role-specific insights from each respondent.

Interviewees were selected among survey respondents who had voluntarily indicated willingness to participate in a follow-up interview. From this group, respondents were selected to achieve a broad spread across departments, functions, and geographical locations. This was done to capture different perspectives on how architectural decisions are initiated, communicated, and coordinated across the organization. In total, twelve interviews were conducted.

The interviews were conducted either online through Microsoft Teams or in person, depending on the respondent's location and availability. The interview language was adapted to the respondent, and Swedish interview material was later translated into English for the thesis. All interviews were recorded with the respondents' consent and subsequently transcribed. Interviews conducted through Microsoft Teams were transcribed using the Teams transcription function, while other recordings were transcribed afterwards using OneNote. The transcripts were reviewed for accuracy and anonymized before analysis. Interview excerpts used in the results chapter were translated into English when needed and lightly edited for readability, while preserving the original meaning of the respondents' statements. Transcription and checking of interview material are important steps in qualitative research, as they support reliable interpretation and help protect participant identity (Sutton and Austin, 2015).

The respondents were informed that participation was voluntary, that the interview material would be treated anonymously, and that the data would only be used for research purposes. This was particularly important since the interviews concerned internal organizational processes and collaboration across functions.

3.3 Data Analysis

The data analysis combined descriptive survey analysis with an iterative thematic coding process of the qualitative material. The empirical material consisted of survey results, open-ended survey responses, interview transcripts, interview notes, and observations from the initial contextual phase. The survey and interview material formed the main empirical basis, while the observations supported contextual understanding and helped interpret patterns identified in the structured data.

The closed-ended survey questions were analyzed descriptively using frequencies, mean values, and visual summaries. This provided an overview of organizational perceptions related to architecture, platform, architectural strategy, decision ownership, cross-functional collaboration, and causes of delays. The open-ended survey responses were used to support the interpretation of the quantitative answers and to provide examples of how respondents explained their perceptions.

The interview transcripts were analyzed through an iterative thematic coding process. Qualitative data analysis commonly involves preparing the data, reading through the material, coding relevant text segments, and developing broader categories or themes (Creswell, 2009). The initial coding structure followed the main themes of the interview guide, while recurring patterns within each theme were grouped into subthemes. This allowed the analysis to combine predefined themes with patterns emerging from the respondents' answers, consistent with an iterative qualitative analysis approach (Creswell, 2009).
The coding was structured in Excel to enable comparison across respondents, roles, and functions. One researcher conducted the initial coding, after which both researchers discussed and refined the categories to reduce the risk of individual misinterpretation. Particular attention was given to similarities and differences between functions, especially regarding decision ownership, handovers, communication, and involvement in early architectural decisions.

Finally, the survey and interview findings were compared and interpreted together. The survey provided a broader snapshot of how the organization perceived the studied issues, while the interviews provided deeper explanations of why these patterns occurred and how they appeared in practice. The final analysis therefore combined quantitative patterns, qualitative explanations, and contextual observations to develop an understanding of how early architectural decisions are initiated, coordinated, and experienced in practice.

3.4 Use of Generative AI Tools

Generative AI tools, including ChatGPT, Microsoft Copilot, and Gemini, were used during the writing process to support language refinement and improve the readability of selected parts of the report. These tools were also used as support in handling large amounts of qualitative material, for example, by helping to summarize and condense interview transcripts into more manageable text segments.

The use of AI was limited to support functions such as wording, structure, clarity, summarization, and organization of material. The interpretation of the empirical data, coding decisions, analysis, and conclusions were conducted and critically reviewed by the authors. All AI-generated suggestions and summaries were manually checked against the original material before being used in the thesis.

No empirical data, final analysis, or conclusions were generated independently by AI. The authors remained responsible for all final content, including the interpretation of interview and survey results.

3.5 Validation of Method

To strengthen the credibility of the study, multiple sources of empirical data were used, including observations, a survey, and semi-structured interviews. This enabled the findings to be compared across different types of data and respondent perspectives. The observations provided contextual understanding, the survey offered a broader overview of the organizational perceptions, and the interviews enabled deeper exploration of the patterns identified in earlier phases. Such triangulation can support the credibility of qualitative research by allowing findings to be examined from several perspectives (Creswell, 2009).

The reliability of the study was supported by documenting the research process, using a structured survey and interview guide, recording and transcribing the interviews, and organizing the interview material through a systematic coding process. However, as the coding and interpretation of qualitative data involve researcher judgment, the analysis cannot be considered fully independent of the researchers' interpretation. To reduce this risk, codes and themes were discussed between the authors and checked against empirical material.

The study was conducted within a specific organizational and industrial context, which limits the possibility of statistical generalization. The survey was also based on a limited number of responses, and the interviewees were selected from respondents who voluntarily indicated willingness to participate. Therefore, the findings should not be interpreted as representing the entire organization or all platform development contexts. Instead, the results should be understood as context-specific insights that may be transferable to similar large-scale engineering organizations working with modular platforms, early architectural decisions, and cross-functional product development.

3.6 Reflection of Methods & Limitations

This study was formed with an iterative and qualitative approach to investigate organizational decision-making processes and early stage platform development. Even if the methods gave valuable results and in depth insights, there were several limitations connected to the study's scope, data collection, and time frame that need to be addressed.

One of the most obvious obstacles for the study was the time frame of only 20 weeks, forcing a narrow delimitation towards qualitative methodology. The study focused mainly on the initial phases of hardware development and excluded software development, this was due to both the author's own area of expertise and time pressure. Furthermore, the time frame also meant that no practical testing was done to fully prove theoretical recommendations.

The semi systematic literature study worked well to map a broad field. However, unlike a fully systematic review, this involves a risk that specific and relevant research may have been overlooked, as the search strategy was adapted based on the authors' emerging understanding.

The survey was distributed to around 120 employees, of whom a number of 25 persons chose to respond. This constitutes a convenient sample, and combined with the low response frequency, the quantitative result from the survey is not statistically generalizable for the whole organization. The primary value was instead in the survey's exploration function, where it effectively pointed at overarching patterns and differences of opinions that later on could be explored deeper during the interviews.

The twelve semi structured expert interviews were crucial to explore the previous answers from the survey. However, since the selection of employees for the interviews was voluntary from the survey study, there was a risk for selection bias. The respondents that chose to attend may have been more engaged, and may have had stronger opinions, or may have had special agendas regarding the current decision-making process.

Some interviews were made in Swedish and were later translated to English for the report. Despite the transcriptions being thoroughly examined, there is always a risk that some technical expressions and linguistic nuances get lost in translation.

4
Results

This chapter presents the study's empirical results. The chapter is divided into two sub-chapters: survey results and interview results. The first chapter compiles the qualitative and quantitative answers from a survey distributed among experienced employees within the organization. The other chapter presents insights from twelve semi-structured expert interviews that highlight platform development and decision-making in practice. Through the chapter, focus is put on highlighting organizational patterns and challenges such as terminology, communication, roles, and cooperation structures in early phases.

4.1 Survey Results

This section presents the results from the survey conducted among experienced staff involved in product development within the studied organization. The results include both quantitative answers based on 1-to-6 scaled answers and qualitative answers in the form of open questions. This section is structured in a way that highlights the main outcomes from the answers.

4.1.1 Respondent Profile

A total of 25 answers were gathered. Respondents represented a broad spectrum of roles related to product development, including system and platform architecture, vehicle and component engineering, product strategy, manufacturing and service engineering, as well as software-related roles.
The respondents reported a broad variation of professional experience with a mean value at 19 years at the current organization, where most respondents have had several different positions during these years.
As it is a large organization, it was crucial to gather information from several geographical locations. Answers were therefore gathered from both North America and Europe.

4.1.2 Understanding of Architecture & Platform

The respondents were asked to describe and define the two terms "architecture" and "platform" in the context of their own area of work. In the answers, architecture was normally defined in relation to system boundaries, interfaces, and interaction rules. Many highlighted the architecture's role in defining how systems or modules interact, how functions are allocated between components, and what limitations that govern the following development of new products (figure 4.1). Others named aspects such as scalability, reusability,

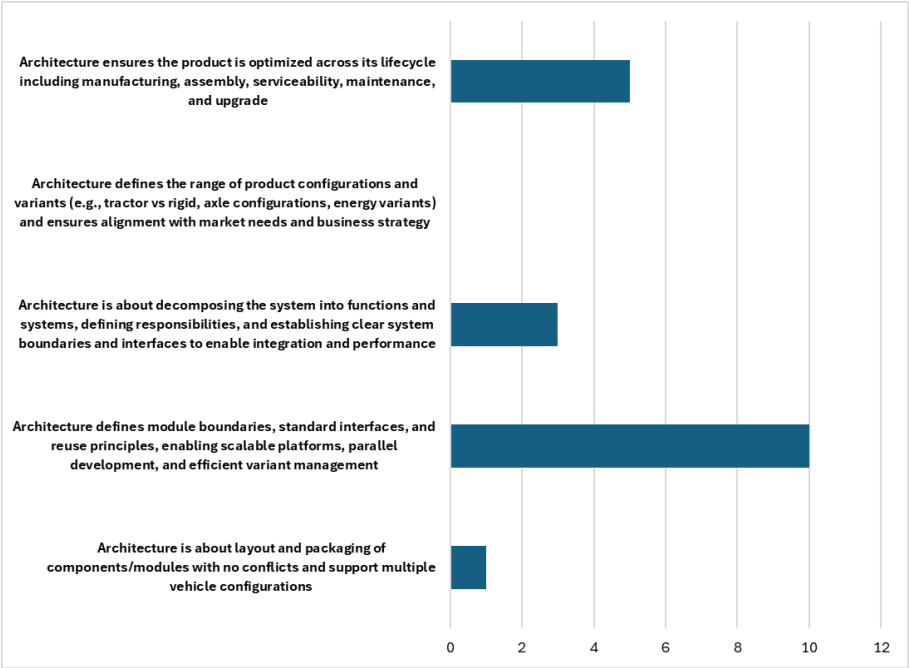

Figure 4.1: Description of Architecture

and the possibility of concurrent engineering. In the open answers, architecture was described as a framework that structures the integration of several subsystems, often with reference to physical layout and interface definition along with life cycle aspects such as manufacturing, serviceability, and upgrading. Architecture was also described as applicable at multiple levels, ranging from individual subsystems to the vehicle level.

The respondents were also asked to describe their understanding of the term "platform" through open answers. This term was often referred to as a collection of reusable modules, components, or systems, usually connected to common interfaces and defined performance steps. Several respondents referred to platforms as building blocks, from which different products or variants can be derived, and also as a catalog of solutions.
Some of the answers showed variation in how the term "platform" was being formulated. Variations were considered regarding both the intended system level and the platform's extent and role in product development. Some respondents described platforms at a component level, through technical solutions or modules that can be integrated with different products. Others described the platform from a more holistic perspective, where it considers vehicles as a whole that acts as a base for several product families.

Another variation was shown in the relation between how architecture and platform were described. In some cases, the platform was thought of as closely related to a specific architecture, while others meant that a platform consists of several architectures or is a framework reaching over several architectural solutions. This includes examples where the platform is viewed as a collection of components or solutions that could be combined in several ways depending on application.
Several respondents noted that the term "platform" was used differently depending on context. In some cases the platform was described as a technical term, while in other

cases it was viewed as a more organizational or strategic way to handle reusability and variations.

4.1.3 Awareness and Understanding of the Architectural Strategy

Several questions in the survey were created to investigate the understanding and the awareness of the overarching architectural strategy within the company. The respondents were asked to estimate their degree of agreement on a Likert scale ranging from 1 to 6 (from strongly disagree to strongly agree) for statements concerning not only the existence of an architectural strategy but also how well that strategy is connected to their own daily work.
The results shown in figure 4.2 show that several persons who took the survey state a relatively high consciousness about the fact that there is an architectural strategy within the organization. On the contrary, the answers vary more when it comes to the understanding of the essence of the strategy and how it relates to their own field of work.

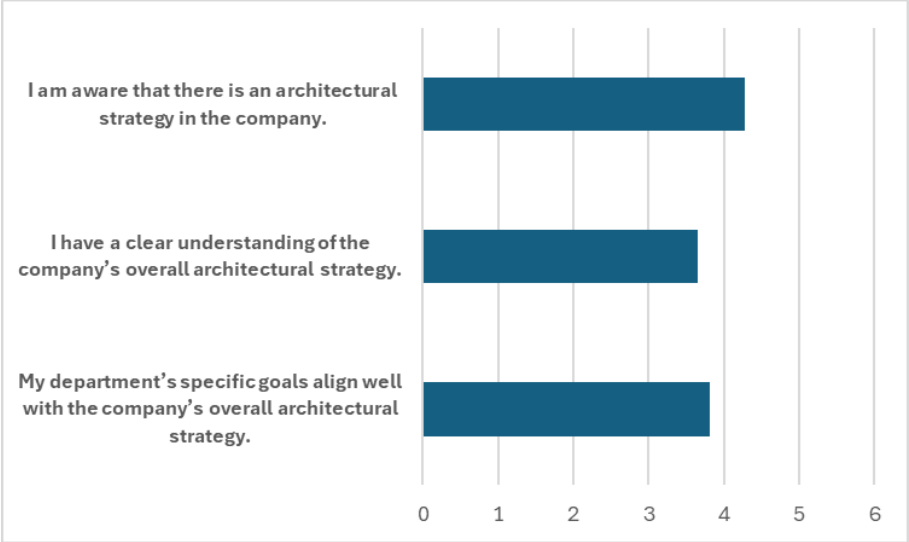

Figure 4.2: Architectural Strategy

4.1.4 Decision-Making and Governance

One part of the survey focused on how technical decision-making and governance are considered in practice, particularly in situations with cross-functional collaboration. Respondents were asked, as in earlier cases, to put a number on a Likert scale covering ownership of decisions, effectiveness in decision-making, and to what extent concerned teams were involved early enough in the process. The answers indicated that the question of "who" the final decision-maker is was not perceived the same way among all respondents. Also, escalation paths when this is unclear are considered differently depending on the respondent. These aspects are shown in Figure 4.3.

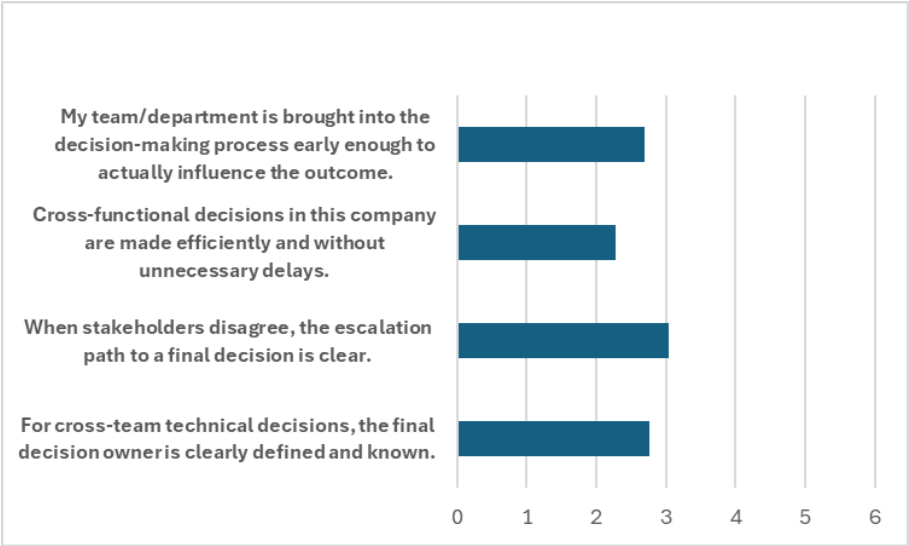

Figure 4.3: Cross-Functional Collaboration

Considering the early involvement of concerned functions and the effectiveness of decision-making, the numbers are generally low. The result highlights that most respondents believe that there is an unnecessary delay when taking decisions concerning cross-functional collaboration as well as being involved too late in decision-making processes. These results are also reported in figure 4.3. In addition to the Likert-scale questions, the respondents were asked to state what factors they believe are the most common causes of delays in technical decisions. Several answers could be selected. The most common occurring alternatives are concluded in figure 4.4.

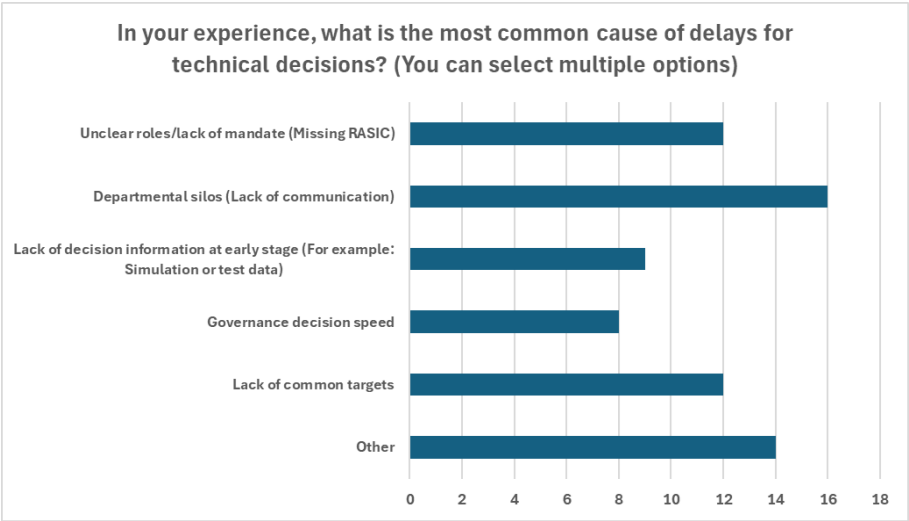

Figure 4.4: Delays in Technical Decisions

4.1.5 Insights from Open Answers

In the survey's open answers it occurs that the first project often sets the agenda for a new architecture, which can create a lock in effect for future needs. Respondents mean that solutions are sometimes named as "common solutions" while not taking into concern demands that are critical for other applications or markets. Some respondents also highlighted that the organization often pushes for a common technical solution without first identifying if there are common requirements. This also corresponds with some answers focusing on the fact that the end product often tends to reflect internal silos and a lack of overall responsibility for the vehicle. This is confirmed by departmental silos being identified as the most common cause for delays in technical decisions. One specific observation is the lack of clear product ownership on a product level, leading to a collection of individual parts rather than an optimized system.

4.2 Interview Results

This section presents the findings from the semi-structured expert interviews. The interviews were conducted to build on the survey results and provide a deeper understanding of how architecture, platform development, decision-making, and cross-functional collaboration are experienced in practice. While the survey provided a broader overview of organizational perceptions, the interviews made it possible to explore the underlying reasons behind these patterns and capture role-specific perspectives from respondents involved in or affected by early architectural decisions.

The results are structured according to the main themes of the interview guide. First, the respondents' profiles are presented to describe the range of roles and perspectives included in the interview study. The following sections then present the findings related to understanding of architecture and platform, architectural strategy and alignment, causes of delays, communication of decisions, cross-functional collaboration, decision-making processes, and improvements and future states.

4.2.1 Respondent Profile

Twelve persons were interviewed across the organization, ranging in roles within vehicle development. These individuals were all aware of the subject, being selected from the previous survey. Interviewed employees within platform architecture, product managers, product planners, manufacturing architects, and digital development specialists.

Among the respondents, involvement within decision-making varied between their respective roles. Some had strong influence over central architectural decisions, such as vehicle layout, functional topology, and system integration. Others contribute with analytical support through evaluation of different options or paths and creation of documentation for decision forums or other analyses. Several respondents participated mainly through coordination and guidance, rather than taking technical decisions.

4.2.2 Understanding of Architecture and Platform

The results from the interviews show that the understanding of the terms "architecture" and "platform" varies significantly within the organization and is often influenced by the specific roles of the respondents. The definition of architecture is described as abstract frameworks or more physical layouts. Several respondents describe architecture as the product's foundational structure, which determines the physical layout, system boundaries, and primary functional properties. Another recurring definition is that architecture constitutes standardized interfaces and works as a set of rules that governs how modules interact and how functions are allocated to specific components. Furthermore, architecture is described as a flexible structural framework that supports different market demands as well as a way to divide the system into functional building blocks to ensure long-term flexibility and scalability. One respondent described this broader view by stating that architecture is about *"breaking systems into functional building blocks that can be combined, moved, and evolved to support long-term direction and flexibility."*

The platform is generally described as something more tangible than architecture. One respondent described it as a "three-dimensional extension of the architecture," consisting of specific technical solutions, performance steps, and rules regulating allowable combinations. The platform is often viewed as the foundation for product variety, where common interfaces and compatible building blocks enable the reuse of components across different product families. For some, the term "platform" is more physical and related to the product's supporting structures, such as the physical structure on which other systems are based. At the same time, another respondent explained that the platform can be seen as *"the technical capability that enables combinations,"* but that this is often confused with what is actually offered to customers.

When respondents were asked if these terms are understood the same way throughout the organization, they were in complete agreement that the terms are not. The understanding is often based on technical discipline, which is noticed in the way of interpreting system-level platforms compared to component-level platforms. One recurring problem that was highlighted during the interviews was that architecture is often misunderstood as only geometrical packaging, leading to critical functional and legal aspects being overlooked. Most respondents pointed out that terms are often used loosely and without a shared definition, creating uncertainty, misunderstanding, and frustration in cross-functional discussions. This was illustrated by one respondent who described that these differences can lead to people *"talking past each other,"* creating frustration and inefficiency.

This lack of shared vision on fundamental terms has practical consequences in development work. Since platforms sometimes are viewed as static and predefined solutions, the architectural work tends to be involved late in projects. This limits the possibility of influencing foundational architectural decisions at an early stage. One example was that design solutions aimed at one specific project were sometimes marked as platform solutions, leading to potential issues at a later stage, where reuse and adaptation were difficult. As one respondent explained, *"project-specific solutions are sometimes treated as platform solutions,"* creating challenges for reuse and adaptation. These differences in interpretation also create a risk of global solutions being forced on markets with different

demands, leading to inefficiency, higher costs, and time-consuming rework.

4.2.3 Architectural Strategy and Alignment

The overarching architectural strategy was, to a broad extent, perceived as vague, unclear, or unstable, especially in relation to battery electric vehicles (BEV). Several respondents described the strategy as emerging, weakly defined, or reactive rather than proactive, making it less usable in daily work. One respondent described the architectural strategy as *"unclear and unstable"* due to changing directives and the lack of a shared long-term goal.

Connecting the architectural strategy to daily decision-making was difficult when short-term project deliveries, time to market, and cost pressures focused on short-term gain rather than long-term goals. In several cases, respondents described that daily work was driven more by near-term project execution than by long-term platform development.

Some respondents also highlighted that technical direction was at times influenced by intuition or strong individual opinions rather than a structured, data-driven decision process. They emphasized the need for greater use of simulations and staged concept evaluations to support more objective decision-making. There is also a noticeable difference in how the strategy is perceived within the organization. While the strategy seems clearer within architecture-related teams, the understanding appears weaker in downstream design teams. This contributes to an environment where the work linked to the architectural strategy of other departments is considered an obstacle rather than an asset. A critical insight found was the lack of "platform-first" thinking, where currently the analysis of the platform often happens after the different projects have already started. This was reflected by one respondent who stated that a *"platform-first approach with upfront analysis before projects"* is missing.

To make the strategy more useful, stronger and clearer leadership is called for, with a mandate to drive architectural questions on a higher level. Several respondents emphasized the need for clearer leadership, stronger decision mandates, and holistic competence to guide architectural and platform-related work across functions.

It was also noted that a common language and early alignment between product development and manufacturing are missing, something that also makes it difficult to implement a cohesive strategy.

4.2.4 Causes of Delays

The results from the interviews show that delays in technical decisions are caused by several connected factors rather than one single reason. A recurring cause described by the respondents was slow and unclear decision-making processes. Several respondents mentioned unclear mandates, long decision lead times, complex decision chains, and uncertainty regarding who has the final decision authority. In some cases, decisions were described as being delayed because they had to pass through several forums or organiza-

tional levels before being concluded.

Another cause of delays was the lack of sufficient information early in the process. Respondents mentioned that decisions are often dependent on data, simulations, market requirements, cost estimations, or technical analyses that are not always available or mature enough in the early phases. This makes it difficult to lock in architectural decisions with confidence. Late changes in requirements were also described as a cause of delay, since new or changed demands during a project can force previous decisions to be reconsidered.

Unclear roles and lack of ownership were also highlighted as important contributors to delays. Several respondents described that when it is unclear who owns a decision, issues tend to be escalated rather than solved directly. This can lead to incomplete information, repeated alignment meetings, and indecision. The lack of common targets was also mentioned as a reason why functions may interpret priorities differently, which can slow down decision-making.

Silos were described as one of the most important organizational causes of delays. Several respondents explained that functions sometimes prioritize their own targets rather than the overall system or vehicle level. This was connected to misalignment in roadmaps, conflicting project requirements, and limited information sharing between departments. The practical effect of this was illustrated by one respondent who described how silos, unclear roles, and lack of common targets limit *"transparency and access to information,"* leading to late changes and inefficiencies. Some respondents also described a lack of system-level ownership, where no single function clearly owns the full product perspective. As a result, trade-offs between functions can become difficult to resolve.

At the same time, some respondents mentioned that collaboration has improved in certain areas, for example, through increased transparency and more structured pre-study work. However, the overall pattern from the interviews indicates that delays are still strongly connected to unclear decision ownership, insufficient early information, silo-based priorities, and weak cross-functional alignment.

4.2.5 Communication of Decisions

The results from the interviews show that communication of major technical decisions varies depending on the type of decision, organizational level, and the respondent's involvement in decision forums. Respondents who were closely involved in project or pre-study forums generally described project-level communication as more structured, often taking place through established project gates, meetings, and documentation. However, communication related to platform direction, long-term architectural intent, and decisions outside specific projects was described as less clear.

Several respondents described that the reasoning behind decisions is not always communicated in a sufficiently clear or traceable way. This was especially highlighted by respondents working with architecture, system-level coordination, and decision support, who

often depend on understanding the assumptions, trade-offs, and constraints behind decisions. In several cases, respondents stated that decisions may be communicated verbally or through informal channels, without a clear written explanation of what was decided and why. This makes it difficult to trace the rationale behind decisions afterwards, especially when decisions affect future or architectural work. The importance of this was reflected in one interview where the missing element was described as clear documentation of *"assumptions, trade-offs, and consequences of decisions."*

The interviews also show that communication is partly dependent on informal networks and personal involvement. Some respondents described that information reaches teams through chance, personal contacts, or established relationships rather than through robust and structured communication channels. This was particularly problematic for respondents who were not directly present in the main decision forums, as they could experience limited visibility of decisions and reduced understanding of the reasoning behind them. One interview illustrated this issue by describing communication as *"inconsistent and often informal,"* where information reaches teams *"by chance rather than through structured communication channels."*

Another recurring pattern was that communication sometimes occurs too late for affected functions to influence the outcome. Several respondents emphasized that early visibility of upcoming decisions is important, as functions need time to analyze consequences, provide input, and understand how decisions affect their area. When decisions are communicated after they have already been locked, the opportunity for bottom-up input and cross-functional alignment is reduced. This was especially relevant for respondents working with regional market needs, manufacturing considerations, or system-level integration.
Transparency was also described as varying across organizational levels. Some respondents stated that decision rationales are available but often require active follow-up to understand. Others described that communication becomes less transparent higher up in the organization or when decisions are cascaded through several organizational layers. As a result, the interpretation of decisions may change or weaken before reaching the teams that need to act on them.

Overall, the interviews indicate that communication works best when decisions are documented, communicated early, and connected to a clear rationale. The main challenges are related to inconsistent communication channels, limited traceability, reliance on informal networks, and unclear communication of long-term platform direction. These challenges affect respondents differently depending on their role, where those closer to project execution often describe clearer communication than those depending on broader platform-level or cross-functional alignment.

4.2.6 Cross-Functional Collaboration

Concerning cross-functional collaboration, a mixed view among the respondents is shared, where several answers given stated that the cooperation works well under specific circumstances while having some flaws on a structural level. A recurring observation is that the effectiveness in the cooperation often is based on individuals and lacks solid structural

support. When engineers with great expertise working together have a shared trust, the open dialog works well, but this is driven by informal coordination rather than clear processes.

Several positive aspects were highlighted during the interviews. For example, it was described that the cooperation has bigger chances of succeeding in early phases when all stakeholders are involved in pre-studies. On an engineering level, the operational cooperation often works well, especially when there is a common language and a practical understanding of the challenges.

At the same time, several challenges were identified that hinder an effective cross-functional collaboration. A central issue is that information shared in common forums often is too shallow. The focus is often aimed at what is being changed, rather than focusing on why it is changed. This was illustrated in one interview where cross-functional forums were described as existing but where the information shared was often *"superficial, focusing on what changes rather than why."* This creates a lack of deep understanding of the consequences of different decisions. Physical distance and cross-site collaboration are also considered challenging, where lack of co-location increases the effort needed to keep the teams aligned, and the cooperation is often reduced to pure information sharing.

Further on, time pressure and an overload of meetings are highlighted as factors that contribute to important input being missed. One risk that was mentioned is that when short-term regional projects are allowed to govern, a tunnel vision effect is being created that counteracts a holistic view on product development and could sometimes lead to "silos" being built.

4.2.7 Decision-making Process

The results from the interviews show that the decision-making process for technical and architectural decisions is perceived as both formal and informal. Several respondents described that there are established forums, project gates, pre-studies, and governance structures where decisions are intended to be prepared and taken. However, the actual decision process was often described as less clear in practice, especially when decisions involve several functions, long-term platform considerations, or different organizational levels.

A recurring issue was unclear decision ownership. Several respondents described that it is not always clear who has the final mandate to make a decision, even when responsibilities are theoretically defined. In some cases, ownership was described as split between different engineering functions, platform-related roles, project-level roles, and management forums. This could lead to uncertainty regarding where decisions should be made, who should provide input, and who has the authority to enforce final direction. Some respondents also described that decision ownership can depend on informal knowledge of who is involved rather than on clearly documented responsibility.

The interviews also showed that technical decisions often follow different paths depending

on the type of decision and the organizational context. Some respondents described a more structured process, where decisions in early-phase studies or advanced engineering work or internal roadmaps before being reviewed in cross-functional forums and later transferred to project execution. Others described the process as more informal and experience-based, where decisions are made by smaller groups through escalation when formal ownership is unclear. This indicates that the process may exist at a formal level but that the practical flow varies depending on timing, urgency, and the people involved. Several respondents described early architectural decisions as being based on a combination of facts, analysis, and educated guesses. In some cases, decisions were supported by simulations, cost estimates, business cases, technical evaluations, or structured comparison of alternatives. However, many respondents also highlighted that early decisions are often made under uncertainty, where requirements, technology maturity, and market conditions are not yet fully known. As a result, expert judgment and previous experience become important, especially when structured data is limited or not yet available.

The main bottlenecks in the decision-making process were described as occurring around unclear mandates, excessive coordination, and transitions between early phases and later development phases. Several respondents mentioned that decisions can be delayed when too many review steps, steering groups, or approval layers are involved. Others described bottlenecks between early exploration and project start, where unclear scope, immature requirements, budget constraints, or unresolved technical uncertainty make it difficult to commit to a direction. In some cases decisions intended to be made in early phases were described as shifting into later project phases, reducing the ability to influence the architecture before it becomes locked.

Timing was therefore described as critical for influence. Respondents who were involved early in pre-studies, road-mapping, or early technical evaluations generally described greater opportunities to shape concepts and architectural direction. In contrast, respondents who were involved after project start often described their influence as limited to reviews, mitigation, or redesign. Late involvement was also connected to increased cost, complexity, and rework, particularly when cross-system dependencies were discovered after architectural decisions had already been made.

Overall, the interviews indicate that decision-making in early architectural work is not only a question of formal process but also of mandate, timing, information maturity, and cross-functional involvement. The process works best when ownership is clear, relevant functions are involved early, and decisions are supported by sufficient analysis and documented rationale. The main challenges are related to unclear authority, informal decision paths, late involvement, and the need to make early commitments despite incomplete information.

4.2.8 Improvement and Future State

When the respondents were asked what they would like to change regarding the current decision-making process, a recurring theme was the need for clearer decision ownership and stronger mandates. Several respondents called for strong technical product leaders

with a full mandate in order to avoid decision-making being governed mainly by short-term project needs. To speed up the process, increased local autonomy was proposed, together with fewer approval layers and a reduction of decision forums. The respondents also emphasized that decisions need to be moved closer to the actual engineering work and that technical experts should be given greater influence. This pattern was reflected in descriptions of a need for stronger product teams and more direct decision-making close to engineering work. Additionally, several respondents highlighted the importance of ensuring that decisions are not only made in time but also followed within the organization. Strong arguments were also made that architecture and platform decisions should be decoupled from project execution and instead be made earlier in the process.

To avoid locking into suboptimal solutions, several respondents suggested a higher use of parallel concept evaluation in early phases. This requires a cross-functional and holistic perspective before specific development projects begin. One recurring improvement area was therefore to strengthen early architectural work through dedicated cross-functional expert teams that continuously work with pre-studies, architectural foresight, and evaluation of future needs.

To strengthen cross-functional collaboration, both structural and cultural changes were proposed. One suggested improvement was increased co-location or shared working cycles in order to improve understanding and reduce conflicts between departments. A broader implementation of agile ways of working across functions was also mentioned as a possible way to improve coordination and dependency management.

Finally, a more data-driven, systematic, and transparent work environment was requested. Respondents highlighted the need for clearer roadmaps, stronger use of previous lessons learned, better early mapping of stakeholders and requirements, and clearer criteria for assessing technical maturity. To create a clearer long-term direction, the respondents also emphasized the importance of common development roadmaps and stronger alignment between strategy, platform direction, and project execution.

5

Discussion

This chapter discusses the findings in relation to the literature review and the purpose of the thesis. The first part synthesizes the empirical findings to understand how early architectural decisions in modular platform development are shaped by shared understanding, requirements, project influence, decision ownership, and traceability. The chapter aims to clarify both what the findings suggest and how the research design affects their interpretation.

5.1 Synthesis of Findings

The empirical findings reveal a chain of interconnected challenges in early-stage modular platform development. Rather than treating the findings as separate issues, this chapter synthesizes them as a broader pattern that influences how architectural decisions are understood, made, communicated, and carried forward over time. This is relevant to the purpose of the thesis, as the study aims to identify which decisions and information are crucial when modular platform development is initiated under uncertainty.

The synthesis starts from the observed lack of shared understanding of central concepts such as architecture and platform. This is then connected to the risk that common solutions are discussed before common requirements have been clarified. When such solutions are shaped by the first project in a new architecture, they may influence future variants and contribute to path dependency. The discussion then moves to the organizational conditions needed to manage this risk, including a clear decision mandate, relevant expertise, and system-level ownership. Finally, it discusses how decision traceability and documented rationale are needed to make early architectural decisions understandable, reusable, and challengeable over time.

5.1.1 Architecture & Platform as Potential Barriers to Integration

The results from the study clearly show that there is a missing common definition of the core terms "architecture" and "platform" within the organization. The empirical study shows that employees define architecture in completely different ways; some see it as abstract interface rules and system boundaries, while others interpret it as a physical layout or a geometrical packaging. The view of what a platform is also varies in a similar way, from being viewed as a physical frame (component) to a broader, more strategic capability or a catalog of solutions. From a theoretical perspective, this confusion of terms can be

explained as "semantic noise" disrupting the flow of information. The literature highlights that when individuals with different disciplinary backgrounds use the same terminology but assign fundamentally different meaning to the words, a hotbed for misunderstandings and disputes is created (McGrath and Whitty, 2015). These conflicts are not due to genuine disagreements but to the parties talking past each other.

The primary consequence of the conceptual discrepancy is that terms lose their critical function as effective coordination mechanisms in daily work. The literature emphasizes that a product architecture is not only a technical structure but also works as a shared representation of the system (Kadaba Jayaprakash et al., 2025). This is crucial for cross-functional decisions to be aligned, especially in early phases. When a common understanding is missing, the coordination between functions is being made more difficult than necessary, creating a risk for decisions being interpreted differently depending on the specific roles of the individuals.

In practice this creates serious consequences for the effectiveness concerning platform development. Interviews confirm that the vague terminology leads to uncertainty, frustration, and misunderstandings in cross-functional discussions. As the literature about organizational communication warns, the lack of consensus results in delayed tasks, double work, and unnecessary internal conflicts that are only based on incorrect assumptions (Salsabilla et al., 2025).

Ultimately, this could result in inefficiency and costly rework in late stages. To avoid these consequences, studies show that establishing a common terminology could be a strategic necessity for architecture and platforms to work integrated within the organization.

5.1.2 From Common Requirements to Common Solutions

The results show a recurring pattern where common technical solutions are sometimes discussed before the underlying common requirements have been fully clarified. This appears both in the survey and in the interviews. In the open survey answers, respondents describe that the first project can set the agenda for a new architecture and that solutions may be named as "common solutions" without fully considering requirements that are critical for other markets or applications. Some respondents also describe that the organization may push for a common technical solution before identifying if there are common requirements. This indicates that the issue is not only whether solutions can technically be reused but also whether the basis for reuse has been sufficiently defined.

This pattern is closely related to the conceptual discrepancy discussed in the previous section. If architecture and platform are understood differently across functions, the meaning of a "common solution" may also vary depending on role, system level, or project perspective. A solution may therefore be perceived as common because it can be reused within a certain technical or project context, while its suitability across broader platform needs may be less explicit. In this way, unclear terminology can make it more difficult to distinguish between a project solution, a reusable module, and a true platform solution.

This finding can be interpreted through the broader theory on product platforms and modular product architectures. Product platforms create value by enabling reuse of shared assets across several products, which can support commonality, economies of scale, and more efficient development. However, the literature also emphasizes that commonality must be balanced against differentiation and variety, since different markets, applications, and customer needs may require different solutions or levels of performance (Muffatto and Roveda, 2000; Liu et al., 2010). Therefore, the value of a common solution depends not only on whether it can be reused but also on whether it supports the required balance between standardization and variation.

The more specific theory on requirements as a basis for platform decisions strengthens this interpretation. Platform architecting is described as a process that moves from market segments and customer needs, through system requirements, toward module boundaries and common platform elements (Hölttä-Otto et al., 2013). This implies that the intended variety or the product family should be understood before deciding which modules or solutions should become common (Hölttä-Otto et al., 2013). Similarly, modular product architecture development should begin by identifying requirements from both external conditions, such as market and customer needs, and internal conditions, such as product structure, production processes, and complexity drivers, before defining standards and modularity. variety, and interfaces (Schuh et al., 2014). From this perspective, commonality should be treated as a requirements-based decision, rather than only as a technical reuse opportunity.

The empirical findings indicate that this logic is not always clearly reflected in practice. Several respondents point to situations where a solution is treated as common while still being strongly influenced by a specific initial project. This does not mean that decisions are necessarily wrong. In early development, projects often need to move forward despite uncertainty, limited information, and time pressure. However, the findings indicate that if the common requirement base is not clearly defined, project-specific needs may influence platform decisions more strongly than intended.

This matters because premature commonality may affect the platform's ability to handle future variety. Modular architectures do not remove complexity entirely but often shift complexity toward module boundaries, interfaces, and coordination between modules. A common solution therefore also affects where future variation can be absorbed in the architecture. If a solution is standardized mainly because it fits an early project, rather than because it reflects shared or compatible requirements across several applications, the platform may become optimized for a narrower scope than intended. In such cases, standardization may limit flexibility, especially when the platform later needs to handle variation in customer use cases, regulation, technical performance, or future product generations (Magnusson and Pasche, 2014).

The findings therefore suggest that the critical information needed in early architectural decision-making is broader than technical solution proposals. Before a solution is treated as common, it may be valuable to clarify which requirements differ between markets or applications, which future needs are still uncertain, and which interfaces need to remain flexible. This would make the decision to standardize more deliberate and reduce the risk

that commonality is created on a narrow or incomplete basis.

Consequently, early architectural decisions may need to consider not only whether a solution can be reused, but also under which conditions reuse creates value for the platform as a whole. Common solutions appear most robust when they are supported by a clear understanding of shared or compatible requirements, while areas of expected variation may need to remain flexible or variant-specific (Hölttä-Otto et al., 2013; Magnusson and Pasche, 2014). This connects to the following section, where the potential long-term effects of early decisions are discussed in relation to path dependency and future platform flexibility.

5.1.3 First Project Locking Future Flexibility

The results show that the first project in a new architecture can have a strong influence on how the platform develops over time. In the open survey answers, respondents describe that the first project may set the agenda for new architecture and create a lock-in effect for future needs. This becomes particularly relevant when early project-specific solutions are later treated as common or platform solutions. Building on the previous section, the issue is therefore not only whether common requirements have been clarified but also how early solutions may shape the decision space for future variants.

This could be understood through the theory of path dependency. Path dependency describes how initial decisions under uncertainty can gradually reduce flexibility, as later decisions are made within the constraints created by earlier choices (Camuffo et al., 2025). In modular platform development, such constraints may take the form of mixed module boundaries, standardized interfaces, packaging assumptions, hardpoints, or technical solutions that become difficult to change once other parts of the system have been developed around them. This is particularly relevant in heavy vehicle development, where architectural decisions define physical topology, interfaces, weight distribution, and long-term product capabilities (Ulrich, 1995; Sköld and Karlsson, 2013).

The concern is not only that the first project influences later projects but also that its architectural assumptions may be difficult to reverse. Theory on early decision-making emphasizes that early architectural choices are made when uncertainty is high, while the freedom to change decreases as the development process progresses (Tan et al., 2017). If early decisions need to be changed later, this can lead to significant rework, cost, and delays (Tan et al., 2017; Verganti, 1997). This makes the first project especially important in platform development, since its solutions may shape the boundaries within which future module development takes place.

The design propagation perspective further explains why such early lock-in can become problematic. In complex products, later changes are rarely isolated to one component or module. Instead, changes can propagate through functional couplings, interfaces, and system dependencies, creating ripple effects across the architecture (Clarkson et al., 2004; Wessels and Pretorius, 2013). This means that if an early project solution becomes embedded in the platform and later proves insufficient for another market, application,

regulation, or product variant, changing it may require adjustments in several connected modules. The issue is therefore not only that an early decision may be difficult to reverse but that reversing it may trigger additional redesign work across the system.

This also connects to frontloading and systemic learning. Since not all future requirements can be known in early phases, the aim is not to eliminate uncertainty completely. Rather, the aim is to bring relevant downstream information, previous project experience, and expected future constraints into the early decision process (Verganti, 1997). In the studied context, this means that the first project in a new architecture may need to be evaluated not only against its own delivery targets but also against possible future variants, market needs, regulatory requirements, and interface flexibility. Such evaluation can help clarify which decisions are safe to standardize and which areas should remain flexible until more information is available.

The findings also relate to the logic of set-based concurrent engineering. SBCE suggests that keeping several alternatives open in early phases can reduce the risk of committing too early to a solution that later proves unsuitable (Raudberget, 2010; Toche et al., 2020). For modular platform development, this does not necessarily mean that all alternatives should remain open for a long time. Rather, it suggests that architectural decisions with high platform impact should be tested against several future scenarios before being locked. This is especially relevant for interfaces and module boundaries, since these determine where future variation can be absorbed without requiring redesign of the entire architecture.

At the same time, the influence of the first project should be interpreted with caution. The first project can also provide necessary learning, technical validation, and concrete direction for the platform. In this sense, it can function as an important reference case. The potential issue arises when the first project becomes a platform reference without this being an explicit architectural decision. In such cases, project-specific assumptions may shape future flexibility stronger than intended.

Consequently, the results suggest that the first project in a new architecture may need a more explicit evaluation against the long-term platform flexibility. Before their solutions are reused as common platform solutions, it may be valuable to assess which requirements they represent, which future variants they support, which interfaces they constrain, and what would be difficult to change later. This would remove uncertainty from early platform development, but it could make the long-term consequences of early architectural choices more visible. In this way, the first project can function as an important learning case without unintentionally limiting future architectural flexibility.

5.1.4 Decision Mandate and Expertise

As discussed in chapter 5.1.3, the choices made during the first platform project tend to create a strong path dependency. This "lock-in" effect is particularly challenging since early architectural decisions are by nature taken under an extreme level of uncertainty. In these phases, future market demands, technical readiness, and the long-term consequences

are rarely fully established. The results from the study highlight the problems from this clearly, when respondents mean that early decisions often need to be based on a combination of facts and educated guesses. To avoid that short-term projects permanently dictate the architecture, expertise from specific employees and their deep understanding on a holistic level is crucial.

By systematically taking advantage of this knowledge is the core of what the literature is calling "systemic learning" (Verganti, 1997). However, it's not enough that experiences from previous projects exists passively within the organization. Specialists must actively be used in pre-studies to predict how early decisions will affect the whole product life cycle. The empirical part does show a current risk. When the uncertainty becomes to big, critical decisions are sometimes pushed to the future, limiting the possibility to strategically form the architecture before locked in by the first project.

Beyond the technical uncertainty, the study shows that there's a clear discrepancy between the need for cross functional expertize and how the formal power is distributed. Decision mandates and ownership is in practice often not defined, creating challenges. Interviews emphasize that there is an uncertainty regarding who's owning a specific decision, especially when the responsibility is experienced as shared between engineering functions, roles and forums. If we analyze this through the RACI framework a clear lack of a defined "accountable" role appears. In complex organizations it's critical to separate between who's contributing with input (consulted) or performs the work (responsible), and the individual who carries the final responsibility (accountable).

When a clear decision owner is missing, questions tends to unnecessarily escalate in the hierarchy, which the empirical part shows leads to insufficient information, repeated meetings, indecisiveness and in the end serious delays. In worst cases, the absence of formal mandates could lead to decision paths being governed disproportionately by strong individuals and informal networks, increasing the risk for local interest, rather than the long term platform strategy governing the decisions.

To handle these challenges and specifically to proactively counteract the path dependency discussed in previous chapter, it's not sufficient to only add more technical expertise in early phases. The organization must establish a much stronger connection between this expertise and a formal decision mandate. Architectural decisions always includes complex compromises that crosses over functions- and module boundaries. In the study, the respondents call for a stronger mandate to avoid decision-making being regulated by short term project goals. This need mirrors the established research about Heavyweight Product Managers. While incremental development often can be handled by more coordinating "lightweight" structures, more in depth architectural platform development requires leaders that does not only have the technical understanding but also the formal authority to make crucial cross functional decisions. Without this clear system ownership, functional silos within the organization risks prioritizing their own local goals, not having a holistic perspective. As the empirical part shows, this could lead to the organization developing a collection of individual, suboptimal parts rather than an integrated an optimized system.

To summarize, successful early architectural decisions requires a combination of in depth expertise and an unquestionable decision mandate. To ensure that the overarching platform strategy is maintained and not unintentionally becomes locked in by short term needs in certain projects, must the decision-making power be moved closer to the actual engineering work but with a clear holistic responsibility. The organization must establish roles and forums with a clear accountability that carries the whole platform perspective. Only then can experts systemic knowledge actually be translated to strategical decisions.

5.1.5 Decision Traceability and Communication of Rationale

The results indicate that decisions are often communicated, but that the rationale behind them is not equally clear or traceable. Several respondents describe that decisions may be shared through meetings, informal networks, or verbal exchanges, while the underlying reasoning is not always documented in a structured way. This means that the organization may know what was decided but not always why the decision was made, what assumptions it was based on, or what alternatives were considered.

This is particularly important for early architectural decisions, since their consequences often appear later, when other teams, projects, or product generations are affected. Design rationale theory emphasizes that documentation should not only capture the final outcome but also the reasoning, trade-offs, alternatives, criteria, and justifications behind the decision (Regli et al., 2000). In the studied context, this means that architectural decisions should ideally be traceable to the requirements, assumptions, rejected alternatives, accepted constraints, and expected future implications that shaped them.

The findings also show that communication and traceability are not the same. A decision may be communicated in a meeting, but still be difficult to retrieve or understand later if the rationale is not captured. This is consistent with design rationale literature, which distinguishes between communication and documentation and highlights that naturally occurring communication is often difficult to structure and reuse over time (Shipman III and McCall, 1997). As a result, relying mainly on informal communication may make platform decisions dependent on personal networks or individual memory.

Limited traceability may also reinforce path dependency. If earlier architectural decisions remain in place without a clear explanation of why they were made, later teams may continue to follow them because they are already embedded in the architecture, rather than because the original rationale still applies. The same issue can occur when an initiative, solution, or project is stopped without documenting why. If the reason for not proceeding was, for example, a technical limitation or lack of maturity, future teams may later repeat the same analysis instead of starting from the point where the previous work ended. Documenting rejected or postponed alternatives therefore helps future teams understand whether the original barrier still applies.

At the same time, decision traceability should not mean documenting every discussion or minor technical choice. Capturing rationale can create cognitive, retrieval, usage, and organizational limitations if the process becomes too intrusive (Horner and Atwood, 2006).

The main need is therefore to document the rationale behind decisions with high architectural or platform impact, including decisions that affect interfaces, module boundaries, common solutions, future variants, or long-term flexibility.

Consequently, decision traceability is important for making early architectural decisions reusable, challengeable, and understandable over time. By preserving not only what was decided, but also why it was considered valid at the time, the organization can reduce dependence on informal networks, avoid repeating already performed analyses, and support learning between projects.

6
Conclusion

This chapter brings together the main findings of the thesis and answers the research question. The chapter first summarizes the conclusions drawn from the study, focusing on what practices can support early-stage decision-making in modular platform development when uncertainty is high. It then presents the proposed framework, which translates the main findings into five key areas for supporting early architectural decisions. The chapter also describes how the framework can be used more practically and ends by outlining directions for future work.

6.1 Concluding Remarks

This thesis set out to answer the research question: ***What practices should be adopted in early-stage decision-making for modular platform development to navigate high uncertainty?*** This study shows that early architectural decision-making under uncertainty is shaped by both technical and organizational conditions. Therefore, the decision-making process needs to be supported not only by technical analysis but also by shared terminology, requirement clarity, platform-level evaluation, clear decision ownership, early expert involvement, and traceable decision rationale.

The main conclusion is that organizations working with modular platform development can benefit from a more structured decision-support framework in the early phases. Such a framework can help ensure that key concepts, such as architecture and platform, are understood consistently across functions, that common solutions are based on common or compatible requirements, and that project-specific solutions are evaluated against long-term platform needs before being treated as common platform solutions. This is especially important since early decisions may influence future variants, module boundaries, interfaces, and the flexibility of the platform over time.

The findings also indicate that early architectural decisions are easier to manage when decision ownership and expert involvement are made explicit. In a cross-functional setting, several perspectives are needed, but the final accountability for architectural decisions with platform-level consequences should be clear. This can reduce uncertainty in the decision-making process and support a stronger balance between short-term project needs and long-term platform considerations.

Finally, the study highlights the importance of documenting why decisions are made, not only what has been decided. Since early decisions are often made under uncertainty,

capturing assumptions, trade-offs, rejected alternatives, and expected implications can help future teams understand, reuse, challenge, or revise decisions when new information becomes available.

In summary, the answer to the research question is that early-stage decision-making in modular platform development should adopt practices that support shared understanding, requirements-based definition of common solutions, platform impact assessment, clear decision governance, and documented decision rationale. These practices can support more deliberate architectural decisions and reduce the risk of premature lock-in when uncertainty is high.

6.2 Framework for Early Architectural Decision-Making in Modular Platform Development

The framework consists of five key areas that should be considered when making early architectural decisions in modular platform development. These areas do not necessarily need to be addressed in a strict sequence. Instead, they should be understood as decision checkpoints that help ensure that the organization has sufficient clarity before decisions with platform-level consequences are locked.

In this thesis, architectural decisions are understood as decisions that influence the structure, interfaces, module boundaries, physical constraints, or long-term flexibility of the platform. These decisions are particularly relevant because their consequences may extend beyond a single component, project, or product variant.

The purpose of the framework is to support decision-making when information is incomplete by making key aspects explicit: Whether the terminology is understood, whether common solutions are based on common requirements, whether long-term platform consequences have been assessed, whether decision ownership is clear, and whether the rationale behind the decision is traceable. In this way, the framework can be used flexibly in pre-studies, platform reviews, project initiation, or architectural decision-making forums.

6.2.1 Establish a Shared Language

One area of the framework is to establish a shared language among the functions involved in early architectural decision-making. The findings show that central concepts such as architecture and platform can be interpreted differently depending on role, function, and system level. These differences are not necessarily problematic in themselves, but they can create uncertainty when decisions need to be discussed, coordinated, and communicated across functions.

Therefore, architectural decisions with platform-level consequences should be supported by an explicit alignment of the key terms used in the decision. This includes concepts as architecture, platform, module, interface, common solution, and variant-specific solution. The purpose is not to create rigid definitions that cover every possible situation, but to

ensure that the people involved in a specific decision have a sufficient shared understanding of what is being discussed.

A shared architectural language can make cross-functional discussions more precise and reduce the risk that different functions interpret the same discussion in different ways. It can also help distinguish between a project-specific solution, a reusable module, and a platform-level decision. In this way, terminology becomes a practical support for early architectural decision-making, rather than only a matter of communication.

6.2.2 Requirements-Based Definition

Another area of the framework is to ensure that common solutions are defined based on common or compatible requirements. The findings indicate that technical solutions may sometimes be treated as common because they fit an initial project, rather than because they have been evaluated against the broader requirement of the platform. In modular platform development, this can create a risk that the project-specific needs influence what later becomes perceived as a platform solution.
Therefore, before a solution is classified as common, the underlying requirement base should be clarified. This includes identifying which requirements are shared across markets, applications, product variants, and internal functions, as well as which requirements differ or remain uncertain. The purpose is to make sure that commonality is based on an understanding of the product variety the platform needs to support, rather than only on technical reuse potential.

A requirements-based definition of common solutions can help distinguish between what should become common, what should remain flexible, and what should be handled as variant-specific. In this way, commonality becomes a deliberate architectural decision, rather than a consequence of the first project or the most immediate development need.

6.2.3 Platform Impact Assessment

Another area of the framework is to assess the platform impact of solutions that are considered common. The findings indicate that early project solutions can influence future platform development, especially when a solution from the first project becomes treated as a common platform solution. While such solutions may be technically valid for the initial project, they may also create constraints for future variants if their broader platform consequences are not evaluated.

Therefore, before a solution is treated as common across the platform, it should be assessed against potential impact on future product variants, module boundaries, interfaces, packaging constraints, and areas where flexibility may be needed. This assessment should clarify what the solution enables, what it limits, and what may become difficult to change later. The purpose is not to predict all future needs but to make the possible consequences of commonality more visible before the solution is adopted across the platform.

A platform impact assessment can support more deliberate decisions about whether a solution should become common, remain flexible, or be treated as variant-specific. In this way, the organization can reduce the risk that short-term project needs unintentionally limit long-term platform flexibility.

6.2.4 Decision Ownership and Accountability

Another area of the framework is to clarify decision ownership while ensuring that relevant cross-functional expertise is involved. The findings indicate that early architectural decisions often require input from several functions, since they may affect interfaces, module boundaries, production, market needs, and future platform flexibility. However, broad involvement also needs to be supported by a clear understanding of who has the mandate to make the final decision.

Therefore, architectural decisions that affect interfaces, module boundaries, or long-term platform flexibility should include an explicit clarification of roles. This means defining who is responsible for preparing the decision basis, who should be consulted for expert input, who needs to be informed, and who is accountable for the final decision. The purpose is not to reduce cross-functional involvement but to avoid situations where shared responsibility leads to unclear accountability.

By combining clear decision ownership with relevant expert input, architectural decisions can be made closer to the technical knowledge while still reflecting the long-term platform perspective. This can support more efficient decision-making and reduce the risk that the decisions are delayed, escalated unnecessarily, or shaped mainly by short-term project priorities.

6.2.5 Traceability of Decision Rationale

The final area of the framework concerns the traceability of decision rationale. The findings indicate that architectural decisions may be communicated through meetings, forums, or informal discussions, but that the reasoning behind the decisions is not always equally traceable over time. This can make it difficult for future teams to understand why decisions were made, especially when requirements, technologies, or market conditions change.

Therefore, architectural decisions with high platform impact should be documented in a way that captures not only what was decided but also why it was decided. This includes the assumptions behind decisions, the trade-offs that were accepted, the alternatives that were considered but rejected, and the expected future implications. The purpose is not to document every discussion, but to make architectural decisions with high platform impact understandable and possible to revisit.

By improving the traceability of decision rationale, the organization can support learn-

ing between projects and reduce dependence on individual memory or informal networks. This makes it easier for future teams to reuse, challenge, or revise earlier decisions when new information becomes available.

6.3 Practical Application of the Framework

The framework is intended to be used as a flexible decision-support tool rather than a strict sequential process. The five areas do not necessarily need to be addressed in a fixed order, but they should be considered before decisions with platform-level consequences are locked. In this way, the framework can support pre-studies, project initiation, platform reviews, and architectural decision forums by making key decision aspects explicit.

To make this framework practically applicable, the five areas can be translated into guiding questions, as shown in Table 6.1. These questions can be used as a checklist to ensure that the organization has sufficient clarity regarding terminology, requirements, platform consequences, decision ownership, and decision rationale.

Table 6.1: Framework

Framework Area	Goal	Control Question(s)
Establish a Shared Language	Ensure that involved functions have a shared understanding of the key concepts.	☐ Are the key terms used in the decision understood in the same way by the involved functions?
Requirements-Based Definition	Ensure that solutions are defined based on requirements.	☐ Have the unique requirements of other applications, regions, or future variants been mapped to ensure this "shared" solution does not penalize them?
Platform Impact Assessment	Make the long-term consequences visible before a solution is locked into the platform.	☐ How may the decision affect future variants, interfaces, module boundaries, and platform flexibility?
Decision Ownership and Accountability	Clarify who owns the decision while ensuring that relevant expertise is involved.	Decide and communicate: ☐ Who is accountable for the decision? ☐ Who prepares the basis? ☐ Who should be consulted? ☐ Who needs to be informed?
Traceability of Decision Rationale	Make the reasoning behind architectural decisions understandable and possible to revisit over time.	Has the rationale been documented including: ☐ Assumptions ☐ Trade-offs ☐ Rejected alternatives ☐ Future implications

By applying the framework in this way, early architectural decisions can become more explicit, better anchored across functions, and easier to revisit over time. The framework

does not remove uncertainty from early modular platform development, but it provides a structure for handling uncertainty more deliberately. It can help distinguish between what should become common, what should remain flexible, who should own the decision, and how the reasoning behind the decision should be preserved.

6.4 Future Work

Future work should focus on validating and further developing the proposed framework in practice. Since this study was limited in time and did not include practical testing of the framework, an important next step would be to apply it in an actual pre-study, platform review, or early project initiation phase. This would make it possible to evaluate whether the framework supports clearer terminology, better requirement alignment, more explicit decision ownership, and improved traceability of decision rationale.

Another relevant direction for future work is to develop the framework into a more detailed decision support tool, focusing on tailoring it to a specific department or organization. This could include a more structured checklist, a decision template, or a digital tool connected to existing project, platform, or PLM processes. Such a tool could help clarify how the framework should be used in practice and how decision rationale should be captured without creating unnecessary administrative work.

Future research could also follow selected architectural decisions over time. Since the consequences of early architectural decisions often become visible only later, this could provide a deeper understanding of how early assumptions affect future variants, interfaces, module boundaries, rework, and platform flexibility. This would also make it possible to evaluate whether the framework helps reduce the risk of premature lock-in.

Finally, the framework could be tested in other departments, product areas, or organizations working with modular platform development. This would help evaluate whether the framework is transferable beyond the studied context and how it may need to be adapted to different organizational settings.

Bibliography

Aithal, A. and Aithal, P. (2020). Development and validation of survey questionnaire & experimental data–a systematical review-based statistical approach. *International Journal of Management, Technology, and Social Sciences (IJMTS)*, 5(2):233–251.

Araci, Z. C. (2017). *Knowledge creation and visualisation by using trade-off curves to enable set-based concurrent engineering applications*. PhD thesis.

Araci, Z. C., Al-Ashaab, A., and Garcia Almeida, C. (2022). Physics-based trade-off curves to develop a control access product in set-based concurrent engineering environment. *International Journal of Lean Six Sigma*, 13(4):824–846.

Baldwin, C. Y., Clark, K. B., et al. (2003). Managing in an age of modularity. *Managing in the modular age: Architectures, networks, and organizations*, 149(1):84–93.

Boer, H. E. E. and Persson, M. (2015). Contrasting platform thinking and product modularization: A survey of swedish development practices. In *22nd Innovation & Product Development Management Conference*.

Bruun, H. P. L., Mortensen, N. H., Harlou, U., Wörösch, M., and Proschowsky, M. (2015). Plm system support for modular product development. *Computers in Industry*, 67:97–111.

Camuffo, A., Gambardella, A., and Kazemi, S. (2025). A theory based analysis of path dependence. *Economics of Innovation and New Technology*, pages 1–24.

Clarkson, P. J., Simons, C., and Eckert, C. (2004). Predicting change propagation in complex design. *J. Mech. Des.*, 126(5):788–797.

Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, Inc., 3rd edition.

Döringer, S. (2021). 'the problem-centred expert interview'. combining qualitative interviewing approaches for investigating implicit expert knowledge. *International journal of social research methodology*, 24(3):265–278.

Eppinger, S. D. and Ulrich, K. (2020). *Product design and development Seventh Edition*. McGraw-Hill New York.

Hamraz, B., Caldwell, N. H., and John Clarkson, P. (2012). A multidomain engineering change propagation model to support uncertainty reduction and risk management in design.

Hirshorn, S. R., Voss, L. D., and Bromley, L. K. (2017). Nasa systems engineering handbook. Technical report.

Hölttä-Otto, K., Otto, K. N., and Simpson, T. W. (2013). Defining modules for platforms: An overview of the architecting process. *Advances in product family and product platform design: methods & applications*, pages 323–341.

Horner, J. and Atwood, M. E. (2006). Design rationale: the rationale and the barriers. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, pages 341–350.

Jahan, N., Naveed, S., Zeshan, M., and Tahir, M. A. (2016). How to conduct a systematic review: a narrative literature review. *Cureus*, 8(11).

Kadaba Jayaprakash, R., Bergseth, E., Törngren, M., and Williamsson, D. (2025). Overcoming challenges in the transition towards battery electric and software-intensive modular heavy-duty vehicles. *Systems*, 14(1):24.

Kerga, E., Rossi, M., Taisch, M., and Terzi, S. (2014). A serious game for introducing set-based concurrent engineering in industrial practices. *Concurrent Engineering*, 22(4):333–346.

Koivisto, E. (2025). Improving role and responsibility clarity through communication in construction project teams.

Li, Z., Cheng, Z., Feng, Y., and Yang, J. (2013). An integrated method for flexible platform modular architecture design. *Journal of Engineering Design*, 24(1):25–44.

Li, Z., Pehlken, A., Qian, H., and Hong, Z. (2016). A systematic adaptable platform architecture design methodology for early product development. *Journal of Engineering Design*, 27(1-3):93–117.

Liu, Z., Wong, Y. S., and Lee, K. S. (2010). Modularity analysis and commonality design: a framework for the top-down platform and product family design. *International journal of production research*, 48(12):3657–3680.

MacCormack, A. and Sturtevant, D. J. (2016). Technical debt and system architecture: The impact of coupling on defect-related activity. *Journal of Systems and Software*, 120:170–182.

Magnusson, M. and Pasche, M. (2014). A contingency-based approach to the use of product platforms and modules in new product development. *Journal of Product Innovation Management*, 31(3):434–450.

McGrath, S. K. and Whitty, S. J. (2015). Redefining governance: From confusion to certainty and clarity. *International Journal of Managing Projects in Business*, 8(4):755–787.

Merati, M., Karbasian, M., Toloie Ashlaghi, A., and Hale, H. (2024). Formulating a design model for product platform architecture adopting integrated design approach considering variety, cost, and accessibility criteria. *Production Engineering*, 18(3):693–708.

Muffatto, M. and Roveda, M. (2000). Developing product platforms:: analysis of the development process. *Technovation*, 20(11):617–630.

Otto, K., Hölttä-Otto, K., Simpson, T. W., Krause, D., Ripperda, S., and Ki Moon, S. (2016). Global views on modular design research: linking alternative methods to support modular product family concept development. *Journal of Mechanical Design*, 138(7):071101.

Raudberget, D. (2010). Practical applications of set-based concurrent engineering in industry. *Journal of Mechanical Engineering*, 56(11):685–695.

Rebentisch, E. and Genta, J. (2017). Applying principles of set-based design to improve ship acquisition.

Regli, W. C., Hu, X., Atwood, M., and Sun, W. (2000). A survey of design rationale systems: approaches, representation, capture and retrieval. *Engineering with computers*, 16(3):209–235.

Robertson, D. and Ulrich, K. (1998). Planning for product platforms. *Sloan management review*, 39(266):19–31.

Rockwell, J. A., Grosse, I. R., Krishnamurty, S., and Wileden, J. C. (2010). A semantic information model for capturing and communicating design decisions.

Salsabilla, R. N. A., Choiriyah, C., Aravik, H., Fadilla, F., Noviani, D., and Sari, E. (2025). Effective communication strategies in business: The use of ambiguous sentences to avoid communication. *Jurnal Ekonomi Bisnis, Manajemen dan Akuntansi*, 4(1):139–152.

Sankowski, O., Küchenhof, J., Dambietz, F. M., Züfle, M., Wallisch, A., Krause, D., and Paetzold, K. (2021). Challenges in early phase of product family development processes. *Procedia CIRP*, 100:840–845.

Schuh, G., Riesener, M., and Breunig, S. (2017). Design for changeability: Incorporating change propagation analysis in modular product platform design. *Procedia Cirp*, 61:63–68.

Schuh, G., Rudolf, S., and Vogels, T. (2014). Development of modular product architectures. *Procedia CIRP*, 20:120–125.

Shamsuzzoha, A. (2011). Modular product architecture for productivity enhancement. *Business Process Management Journal*, 17(1):21–41.

Shipman III, F. M. and McCall, R. J. (1997). Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *AI EDAM*, 11(2):141–154.

Sköld, M. and Karlsson, C. (2007). Multibranded platform development: a corporate strategy with multimanagerial challenges. *Journal of Product Innovation Management*, 24(6):554–566.

Sköld, M. and Karlsson, C. (2013). Stratifying the development of product platforms: Requirements for resources, organization, and management styles. *Journal of Product Innovation Management*, 30:62–76.

Slattery, E. L., Voelker, C. C., Nussenbaum, B., Rich, J. T., Paniello, R. C., and Neely, J. G. (2011). A practical guide to surveys and questionnaires. *Otolaryngology–Head and Neck Surgery*, 144(6):831–837.

Smith, M. L., Erwin, J., and Diaferio, S. (2005). Role & responsibility charting (raci). In *Project Management Forum (PMForum)*, volume 5, page 12.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of business research*, 104:333–339.

Sutton, J. and Austin, Z. (2015). Qualitative research: Data collection, analysis, and management. *The Canadian journal of hospital pharmacy*, 68(3):226.

Tan, J., Otto, K., Wood, K., et al. (2017). A comparison of design decisions made early and late in development. In *DS 87-2 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 2: Design Processes, Design Organisation and Management, Vancouver, Canada, 21-25.08. 2017*, pages 041–050.

Thomke, S. and Fujimoto, T. (2000). The effect of "front-loading" problem-solving on product development performance. *Journal of Product Innovation Management: An International Publication of the Product Development & Management Association*, 17(2):128–142.

Toche, B., Pellerin, R., and Fortin, C. (2020). Set-based design: a review and new directions. *Design Science*, 6:e18.

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research policy*, 24(3):419–440.

Verganti, R. (1997). Leveraging on systemic learning to manage the early phases of product innovation projects. *R&D Management*, 27(4):377–392.

Wessels, A. and Pretorius, L. (2013). The impact of a design change in an engineering development environment: A case study. In *2013 IEEE International Conference on Industrial Technology (ICIT)*, pages 1517–1523. IEEE.

Williamsson, D. and Sellgren, U. (2021). Integrated modularization methodology and process for heavy-duty trucks.

A

Appendix 1

Interview Guide

1. Introduction

The purpose of this interview is to deepen the understanding of how architecture, decision-making, and collaboration work in the organization. The interview builds on patterns from the previous survey, focusing on general experiences.

Consent

The interview is anonymous and will only be used for research purposes.

2. Background

- Can you briefly describe your role and area of responsibility?
- How are you involved in architecture or technical decision-making in your work? What type of decisions are you involved in?

3. Understanding of Architecture and Platform

- How do you personally define “architecture” in your work?
- How do you define “platform”?
- Do you think these concepts are understood in the same way across the organization, or do they vary?
- What are the practical consequences of these differences?
- Have you experienced situations where different interpretations created problems or misunderstandings?

4. Architectural Strategy and Alignment

- How clear do you perceive the overall architectural strategy to be?
- How is the strategy connected to your daily work?
- Could you give an example on specific information (e.g. simulation data, market demands) that is crucial for you to “lock” an architecture?
- To what extent do you feel that your department’s goals align with the overall strategy?
- What is missing to make the strategy clearer or more useful?

5. Communication of Decisions

- How do you experience how major technical decisions are communicated today?
- How clear is the reasoning behind decisions?
- What works well in the communication?
- Is there anything within communication that you believe is missing? Could you give an example?
- Can you describe a concrete example where communication worked well or poorly?

6. **Decision-Making Process**

- How clear is it in practice who owns a decision?
- What happens when it is unclear?

Process and Flow

- Can you describe how a typical technical decision is made, from start to finish?
- Do you feel like decisions taken early are based on facts or rather on educated guesses? What type of information could be needed to move these decisions to an earlier stage?
- Where in the process do challenges or bottlenecks occur? Why do you think these challenges/bottlenecks oc cur?
- How do you use data or knowledge from previous projects to make decisions at an early stage? Are there any methods for this?

Timing and Involvement

- When is your department typically involved in decisions?
- How does the timing affect your ability to influence the outcome?

7. **Causes of Delays**

- What do you perceive as the most common causes of delays in technical decisions?
- To what extent do factors such as silos, unclear roles, or lack of common targets play a role?
- How do these challenges manifest in practice?

8. **Cross-Functional Collaboration**

- How does cross-functional collaboration work today?
- What works well and what are the main challenges in cross-functional work?

9. **Improvement and Future State**

- If you could change one thing about how decisions are made today, what would it be?
- What concrete changes do you think would have the biggest impact?
- Are there any ways of working or structures that you think work particularly well and sho uld be used more?

10. **Closing**

- Is there anything we haven't asked about that you think is important?

DEPARTMENT OF TECHNOLOGY MANAGEMENT AND ECONOMICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY