Universität der Bundeswehr München
Institut für Technische Produktentwicklung

A coordination perspective of agility in automotive product development.

Julian Immanuel Schrof, M.Sc.

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

genehmigten Dissertation.

Gutachterin / Gutachter:

1. Univ.-Prof. Dr. Ing. Kristin Paetzold
2. Prof. Dr. Ing. Torgeir Dingsøyr

Die Dissertation wurde am 07.06.2022 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 07.11.2022 angenommen. Die mündliche Prüfung fand am 21.11.2022 statt.

Acknowledgement

I would like to start by acknowledging that there have been many people who have helped me to finish my Ph.D. thesis with inspiration and motivation. I would like to express my deepest gratitude to all of you.

First, I am deeply grateful for my wonderful wife Christina who has been my foundation throughout this project. I also would like to thank my parents and my sisters for their belief in me and for introducing me early to the fascinating world of science.

I would like to express my sincere gratitude to my advisors Prof. Dr. -Ing. Kristin Paetzold and Prof Dr. -Ing Torgeir Dingsøyr for their guidance, motivation, and trust. Both invested many hours to discuss theories and review findings which I am deeply grateful for. Kristin gave me the opportunity to conduct this industrial research project with scientific rigor at her chair. Torgeir inspired me to connect the novel research field agile automotive development with the established field large scale agile software development. From our exchange emerged the idea to employ coordination theory as the theoretic lens for this thesis.

I would like to extend my sincere thanks to Dr. rer. nat. Sabine Rittmann, Dr. rer. nat. Florian Fischer and Dr. -Ing. Nicolai Martin who were my mentors at the BMW Group. Sabine encouraged me with both here ease and proficiency throughout the complete PhD project and has become a dear friend. Florian's farsighted vision of agile automotive design laid the groundworks for my research and our exchange was a prominent orientation. Nicolai encouraged me to overcome obstacles and find my own path by sharing his own experiences from top level management. My sincere thanks go to my colleagues at the BMW Group, who believed in me and gave me the chance to validate theoretical constructs in practical projects. I would like to highlight the welcoming cooperation with Dr. -Ing. Robert Irlinger, Dr. -Ing. Johannes Staeves and Rainer Rump.

Furthermore, I would like to thank the whole team at the LRT 3 in Munich for the great working atmosphere, the fruitful discussions, and the organizational and technical support. Especially the friendly collaboration with Dr. -Ing. Alexander Atzberger and Dr. -Ing. Tobias Schmidt has been very valuable. I was also lucky to supervise several excellent students during their Bachelor and Master theses. I would like to highlight the great teamwork with Felix Rathert, Franziska Scharold, Jörg Holzer and Andreas Sedlmair. I am sure that I learned at least as much from them as they did from me.

I would also like to thank the whole SINTEF Digital team in Trondheim, Norway where I spend a fruitful research exchange. I appreciate the open exchange, the new perspectives, and the invitation into exciting research projects. Thanks to Anastasiia Tkalich and Tor Sporsem I felt like a part of the team from the very first day. They made my stay most enjoyable.

Abstract

Traditional automotive companies are increasingly object to unfamiliar competition from tech companies. The concomitant speed in new product development and the induced changes in technology overstrain their established product development systems. Alternative, more flexible and customer-oriented design approaches such as agile product development are necessary instead. Agility reflects the continual readiness to create, embrace, react to and learn from change to improve customer value. While agile product development has become a standard in software development its transferability to mechatronic product development in general and cars in specific is yet to be proven. The aim of this research is to explore agility in automotive product development. It is divided into three research objectives. First, to systematize agile product development in respect to design context characteristics based on coordination theory. Second, to evaluate agile methods in the automotive domain and categorize agile constraints. Third, to generate domain specific agile coordination strategies to avoid the experienced constraints.

To accomplish this research aim an Action Research methodology was employed. During a four-year research project agile methods and practices were introduced to a spectrum of automotive development requirements in eleven pilot projects. Change in the form of adjusted agile practices was actively and repeatedly introduced to observe its impact on development dynamics. The methodology allows to iteratively design and evaluate context-specific agile practices in collaboration with affected product designers within their application contexts. The researcher was an active part of the development projects and able to directly collect qualitative data sets. To ensure research rigor participation across projects was varied, data sets were analysed according to a standardized process, and findings were cross-referenced with supplementary qualitative and quantitative data sets from outside the pilot projects.

A coordination reference model is established to provide a comprehensive understanding of agile product development in relation to context characteristics. The findings show that agile methods rely on emergent, self-adjusting coordination strategies based on mutual adjustment coordination modes. The lightweight composition of interlinked coordination mechanisms autonomously adjusts to changing project dynamics. But in the automotive domain agile product development is limited by constraints of scale and physicality. Both cause multiteam development systems and translate into coordination determinants that overstrain original agile coordination strategies. Their lack of inter team coordination mechanisms outbalances the self-adjustability of the coordination system. Three scenarios are presented to avoid this imbalance. Scenario one introduces domain-suitable inter team coordination mechanisms which match automotive coordination determinants. Scenario two applies digital development technologies which enable to develop hardware like software products. Scenario three changes the product structure to realize coordination determinants that suit original agile coordination strategies.

The research improves the applicability of agile product development in the automotive domain. It provides a straightforward tool to adjust agile methods to project specific requirements. Additionally, it allows to estimate realistic benefits of agile product development based on project characteristics. The theoretical contribution of the research includes a model-based understanding of agile system behaviour in different application contexts. This proceed is not limited to automotive development and hence opens opportunities to research agile product development in further domains. Moreover, the comparison of constraints of scale and physicality in the automotive development shows how opposing characteristics of domains cause similar limitations to agility and hence allows to connect both research streams.

Kurzfassung

Traditionelle Automobilkonzerne sind in zunehmender Weise einem aggressiven Wettbewerb durch Tech Companies ausgesetzt. Die gesteigerte Entwicklungsgeschwindigkeit für Produkte und der beschleunigte Technologiewandel überfordern dabei die etablierten Entwicklungsprozesse. Stattdessen sind flexiblere und kundenzentrierte Entwicklungsmethoden wie agile Produktentwicklung notwendig. Agilität umfasst die kontinuierliche Bereitschaft Wandel zu erzeugen, zu akzeptieren, auf Wandel zu reagieren und davon zu lernen, um den Kundenwert zu erhöhen. Obwohl agile Produktentwicklung mittlerweile ein Standardvorgehen in der Softwareentwicklung ist, muss die Übertragbarkeit auf die Entwicklung mechatronischer Produkte wie Autos noch nachgewiesen werden. Das Ziel dieses Forschungsprojekts ist daher die Untersuchung von Agilität in der Automobilentwicklung. Dieses Ziel teilt sich in drei Teilziele auf. Erstens, die Systematisierung agiler Produktentwicklung in Abhängigkeit zu domänenspezifischer Entwicklungsrahmenbedingungen basierend auf der Koordinationstheorie. Zweitens, die Evaluierung bestehender agiler Methoden in der Automobilentwicklung und die Kategorisierung realer Hemmnisse. Drittens, die Entwicklung von kontextspezifischen agilen Koordinationsstrategien, um die analysierten Hemmnisse zu umgehen.

Um dieses Forschungsziel zu erreichen, wurde eine Action Research Methodik verwendet. Im Rahmen eines vierjährigen Forschungsprojekts wurden agile Methoden und Praktiken anhand eines repräsentativen Spektrums von Anforderungen der Automobilentwicklung in elf Entwicklungsprojekten getestet. Wandel in Form von angepassten agilen Praktiken wurde aktiv und wiederholt eingeführt, um die Auswirkungen auf die Entwicklungsdynamik zu bewerten. Die Methodik erlaubt es, in Zusammenarbeit mit den betroffenen Produktentwicklern kontextspezifische agile Praktiken iterativ zu gestalten und zu evaluieren. Der Forscher war dabei ein aktiver Teil der Entwicklungsprojekte und in der Lage, qualitative Daten direkt zu erheben. Um die objektive Aussagefähigkeit der Daten zu gewährleisten, wurde die Beteiligung des Forschenden zwischen den Projekten variiert, die Datensätze nach einem standardisierten Verfahren analysiert und die Ergebnisse mit ergänzenden qualitativen und quantitativen Datensätzen von außerhalb der Pilotprojekte abgeglichen.

Das entwickelte Koordinationsmodell ermöglicht ein umfassendes Verständnis der agilen Produktentwicklung unter Berücksichtigung spezifischer Anwendungskontextmerkmale. Die Ergebnisse zeigen, dass agile Methoden auf emergenten, sich selbst anpassenden Koordinationsstrategien beruhen. Miteinander verknüpfte Koordinationsmechanismen passen sich selbständig an Projektdynamiken an. In der Automobilbranche ist die agile Produktentwicklung jedoch durch Hemmnisse aufgrund der Körperlichkeit des Produkts und der Skalierung des Entwicklungsprozesses begrenzt. Beide bedingen Entwicklungssysteme bestehend aus voneinander abhängigen Teams. Aufgrund fehlender teamübergreifender Koordinationsmechanismen funktioniert die Selbstanpassungs-fähigkeit der agilen Koordinationsstrategien nicht mehr. Es werden drei Szenarien vorgestellt, um dieses Ungleichgewicht zu umgehen. Szenario eins führt teamübergreifende Koordinationsmechanismen ein, die den Koordinationsdeterminanten in der Automobilentwicklung entsprechen. Szenario zwei führt digitale Entwicklungstechnologien ein, die es ermöglichen, mechatronische Produkte ähnlich wie Softwareprodukte zu entwickeln. Szenario drei verändert die Produktstruktur so, dass sich Koordinationsdeterminanten ergeben, die den ursprünglichen agilen Koordinationsstrategien entsprechen.

Die Forschungsergebnisse ermöglichen die Anwendbarkeit agiler Produktentwicklung in der Automobilentwicklung. Sie beinhalten zudem ein einfaches Werkzeug zur Anpassung agiler Methoden an projektspezifische Anforderungen. Darüber hinaus ermöglichen sie eine realistische Abschätzung des Nutzens agiler Produktentwicklung basierend auf realen Entwicklungsbedingungen. Der theoretische Beitrag der Forschung beinhaltet ein modellbasiertes Verständnis des agilen Systemverhaltens in unterschiedlichen Anwendungskontexten. Diese Vorgehensweise ist nicht auf die Automobilentwicklung beschränkt und eröffnet daher Möglichkeiten zur Erforschung agiler Produktentwicklung in weiteren Domänen. Darüber hinaus zeigt der Vergleich der Hemmnisse durch Skalierung und Produktkörperlichkeit in der Automobilentwicklung, wie gegensätzliche Charakteristika von Domänen zu ähnlichen Hemmnissen für Agilität führen und erlaubt es, beide Forschungsgebiete zu verknüpfen.

Contents

1 Introduction

Agility in product design reflects a continual readiness to create, react, embrace, and learn from change in order to improve customer value. Agile product design summarizes interlinked design practices and methods that rely on a shared set of values and principles to realize this agility. While agile product design has become a standard in software development its transferability to physical products is yet to be evaluated. The research focus of the thesis at hand is agile product design in the automotive domain.

The research aim is to comprehend and enable agility in automotive product design. The first research objective is to analyse agile system behaviour based on coordination theory. The second research objective is to collect and categorize agile constraints in automotive and to adjust the theoretical reference model to match the empirical data. The third research objective is to recommend supplementary agile practices to outbalance the identified flaws of agile product design in automotive application contexts.

The contribution of the research includes a theoretical understanding of agile system behaviour in different application contexts and a practical adjustment of agile methods to the automotive domain based on straightforward design practices. This Introduction chapter is divided into problem outline, research strategy and contribution, and structure of the thesis.

1.1 Problem outline, motivation, and relevance of research

The automotive industry is currently object to a set of tendencies that progressively overstrain its established product design systems. Both exogenous and endogenous factors accelerate **the dynamics and relevance of change in automotive product design** (Stelzmann, 2012). The VUCA acronym (volatility, uncertainty, complexity and ambiguity) summarizes both the endogenous and exogenous change factors well (Bennett and Lemoine, 2015). It characterizes the dynamics in automotive design and underlines the urgency to reconfigure automotive design with a focus on flexibility, speed, and customer value.

Regarding **exogenous factors**, automotive OEMs are increasingly challenged by growing and heterogeneous regulatory requirements, unfamiliar competitors, and inexperienced customer behaviour. The ability to manage the product complexity in automotive design has traditionally been the distinguishing capability of OEMs (Schömann, 2011). But faster-changing markets contradict the established long duration product design projects. Time to market is crucial in digital product design (Wedeniwski, 2015) since shorter technology cycles create new customer expectations (Baltes and Selig, 2017). Technological trends in general are a dominant driver for changing customer requirements (Ebel and Hofer, 2014). Large automotive OEMs lack the implementation and adaption speed to cope with the faster development cycles of digital technologies and hence cannot compete with competitors that originate from digital product design. Speed of development has become a key competitive factor (Díaz, 2011). Customers' preferences shift towards digitally connected vehicles and mobility. The availability of mobility replaces personal ownership of vehicles in some markets. Shared mobility is driven by new and aggressive companies such as Uber or Lyft. Relying on well-established premium brands won't suffice anymore in a market that is characterized by a changing concept of personal mobility (Ueding, 2014). Automotive OEMs will have to adapt to the short innovation cycles of the information technology and the consumer's electronics industry to maintain their competitiveness in an increasingly dynamic future (Kortus-Schultes *et al.*, 2014).

Endogenous factors that accelerate change in automotive product design are the growing complexity of the product and the shifting balance between involved disciplines and their cooperation in an integrative design process. Product complexity has increased significantly throughout the last two decades. Automotive OEMs need to integrate up to 10,000 parts per vehicle from 3,000 suppliers throughout the design process under the presumption of production rates of up to 2,000 cars per day (Schömann, 2011). The mechatronic nature of the automotive product requires the connected design activities of mechanical engineering, electrical engineering and software engineering amongst additional disciplines (Lefèvre et al., 2014). The interplay between these disciplines is difficult (Luckel et al., 2000). Electrical engineers and software designers develop function-oriented in fast cycles while mechanical engineers focus on component design based on long-lasting, hardware-intensive verification cycles (Hellenbrand, 2013). The faster pace of electronics and software design contradicts the established, hardware-focused design process in automotive (Eigner, 2021). This imbalance even worsens since the relevance of software and electronics has increased significantly in automotive design throughout the last two decades and continues to do so (Hensel, 2011).

1.2 Agility in automotive design

Software development companies had been object to very similar challenges twenty years ago and established agile product design in response. While independent concepts and individual lightweight methods had been employed since the 1970s (Abbas *et al.*, 2008) the Manifesto for Agile Software Development (Fowler and Highsmith, 2001) officially coined the term Agile and integrated earlier approaches based on a shared set of values and principles in 2001. Since the publication of the manifesto agile design has gained widespread use and has become a standard in software design (VersionOne, 2020), even though it lacks a holistic theoretical explanation (Baham and Hirschheim, 2021; Dingsøyr *et al.*, 2012; Rathor *et al.*, 2016). Agility in product design is characterized by a readiness to create, react, embrace and learn from change to improve customer value (Conboy, 2009). The core concepts are inspect and adapt cycles, incremental and iterative development, collaboration in teams, and continuous customer involvement (Baham and Hirschheim, 2021). Based on these concepts, agile methods are lightweight combinations of design practices that are adapted to different application contexts and use cases.

Numerous publications confirm that agile product design enables design flexibility, higher team productivity, early customer involvement, delivery speed and shorter time to market (Pikkarainen *et al.*, 2008). These characteristics match the presented challenges in automotive design very well. But the ideal application context or sweet spot for agile methods are small, collocated, self-organized teams that design software products being object to medium to high levels of change, and little external dependencies (Boehm, 2002; Kruchten, 2013). The automotive design context differs significantly from these conditions. It relies on a highly interdependent multiteam design system and the enablement of agility in automotive design is therefore not plug and play. Two central characteristics of automotive design diverge from agile sweet spot conditions. First, the scale of the design process which includes hundreds of teams that need to cooperate closely throughout design projects which last several years. Second, the physicality of the product which is fundamentally opposing to the nature of software since it requires the cooperation of more disciplines and additional design steps such as production and logistics that are not relevant in software design.

Both factors have been researched in separated research streams and coined as challenges of scale (Dingsøyr *et al.*, 2014) and constraints of physicality (Ovesen, 2012) to agility in their respective application contexts. Both categories are termed constraints in the thesis at hand to facilitate readability. Constraints of scale for large scale agile software development have been reported and classified by several literature reviews (Dikert *et al.*, 2016; Edison *et al.*, 2021). The scale of the process complicates the practical implementation of the agile core concepts and the original agile methods. Necessary cooperation between autonomous teams is a central contradiction that shapes the constraints of scale category. Determining the right coordination mechanisms to suit both objectives is central to the research of large scale agile (Gustavsson, 2020a). Constraints of physicality are a less mature research stream that has emerged with the first use of agile methods in non-software product design only ten years ago (Ovesen, 2012; Schmidt *et al.*, 2019). Characteristics of the hardware product such as the materialization process or the physical dependencies between components complicate agile product design practices. Original agile principles and methods based on software design do not match the requirements of hardware design (Schrof *et al.*, 2018). To enable agility in automotive design the relevance of both constraints' categories and their mutual influence onto each other must be evaluated.

The existing literature on agile automotive design focuses mostly on experience reports and includes few theory-grounded publications. Rigby et al. describe in a *Harvard Business Review* article agile software design at Bosch and agile design approaches at Tesla with a focus on product modularization (Rigby *et al.*, 2018). Denning describes the agile transformation at Volvo cars with a focus on shared planning meetings based on the scaled method *SAFe* (Denning, 2020). Ekedahl and Berger investigated scaled agile development in mechatronic organizations which several automotive companies and included expected benefits and recommended 26 mechatronic specific agile practices (Eklund and Berger, 2017). Other reports focus on large scale agile software design projects within automotive product design in early product stages (Weber, 2015). Some experience reports spread fascinating success based on easy metaphors and analogies to adjacent fields instead of well-grounded scientific reports and theories which drives the ambiguity of agile design (Janes and Succi, 2012). Completely agile automotive design projects are limited to non-serial and non-standard products in individual design projects (Denning, 2012). Hohl et al. explain the lack of necessary agile design in automotive by the factors inertia, fear, and context. Inertia is caused by an incomplete understanding of agility and its implications, fear is driven by the threat of management losses and context summarizes non-ideal requirements for agile design in automotive (Hohl *et al.*, 2016). Especially the amount of disciplines that need to cooperate is seen as a central hurdle (Poth and Wolf, 2017). These practice-oriented publications reflect the early stage of the research phenomena agile automotive design and open several **research gaps**. First, the research lacks a shared understanding of agility in automotive design. Second, there is no theory-based decomposition of agile automotive design even though several theoretical lenses have been employed to examine agile software development (Strode *et al.*, 2012; Vidgen and Wang, 2009). Third, the literature lacks rich case descriptions, a summary and categorization of agile constraints in automotive design and a domain-specific comparison to the established constraints of scale and physicality categories. Fourth, the experienced problems in agile automotive design lack a comprehensive theoretical grounding. The thesis at hand addresses these research gaps with the aim to realize agility in automotive design.

1.3 Research strategy

To address the identified research gaps **this study aims to investigate and enable agility in automotive product design**. It therefore analyses agile design practices within automotive product design to define and explain the application context specific constraints. Based on this analysis it recommends and evaluates adjustments to both the employed agile methods and the application context respectively. Three research questions (RQs) divide the research aim into research objectives and thus provide a structure to the complete research project.

RQ 1: How to explain agility and its benefits theoretically?
RQ2: What constraints reduce agile design applicability how in automotive design?
RQ3: How to enable agility in automotive product design?

The first research question addresses the need for a theory-based decomposition and explanation of agile product design to analyse its applicability to the automotive context. Coordination theory is chosen as the most suitable design theory to construct a reference model of agile design based on the coordination strategy concept. It allows to analyse and model agile design system behaviour independent of the application context. The second research question focuses on specific constraints of agile design in automotive application contexts. Agile pilot projects are used to collect challenges in real world automotive application contexts. The data is analysed regarding the influence of constraints of scale and physicality. The established coordination-theory specific understanding of agile system behaviour allows to determine how the constraints impact agile design functionality. The third research question builds on the earlier findings and addresses concepts to outbalance or avoid the experienced constraints in automotive application contexts.

Pilot projects were the backbone of the practical implementation of the research strategy. Agile methods and practices were introduced to eleven different development projects to study their impact on automotive design. During the pilot projects change was repeatedly introduced in the form of adjusted agile practices to observe the changing impact on the respective design project. This allowed to iteratively design and evaluate context-specific agile practices in collaboration with product designers within their practical application contexts. Findings were compared and transferred between the sequential pilot projects. Unlike in case studies, the researcher was an active part of the design projects. This **Action Research methodology** is based on an Idealist ontological and a Constructivist epistemological position. It addresses both practical utility and contribution to design theory to ensure research relevance and validity. Social interactions in pilot projects are studied by introducing change and observing the effects. The researcher is part of the pilot projects and discontinuous the objective observer position. Activities of design artefact construction, intervening with the organization and evaluating the impacts are iteratively interwoven (Sein *et al.*, 2011).

To ensure research objectivity the data analysis was conducted according to a research model based on coordination theory. This coordination reference model accurately mirrors central traits of agile design. It is based on the original model of Van de Ven (Ven *et al.*, 1976) and supplemented with the coordination concepts of boundary spanning (Star and Greisemer, 1989) and cognitive coordination (Espinosa *et al.*, 2004) to better reflect agile core concepts. The model allows to analyse the coordination efficiency of interlinked design practices in relation to different application contexts. Initial theory selection and model development was guided by a structured literature review (Okoli and Schabram, 2010) and continuously enriched throughout the pilot projects by a narrative literature review (Boell and Cecez-Kecmanovic, 2015). The research does not address the necessary change management in agile transformations. Detailed comparisons of scaled agile software development methods like the literature review of Edison et al. (Edison *et al.*, 2021) are also out of scope.

1.4 Research contribution

The research conceptualizes agility as an attribute and as a construct to ensure an unambiguous understanding of agile design and avoid the Guru problem (Janes and Succi, 2012). It employs coordination theory as a theoretical lens to decompose and analyse agile methods and explain their empirically proven functionality. Unlike earlier coordination models the coordination reference model combines coordination theories from different fields to best reflect agile design characteristics. It allows to analyse individual agile practices and agile system behaviour in relation to the application context based on their coordination strategy efficiency. To do so the model employs the theoretical concept coordination strategy to connect dynamically coupled application environment characteristics with coordination practices.

The research provides rich case descriptions of agile design in automotive application contexts. The respective data analysis leads to a summary and categorization of problems of agile methods across pilot projects. The data analysis proves the existence of constraints of scale and physicality in automotive design. Constraints of physicality are viewed as an amplification of constraints of scale to simplify problem understanding of the research phenomena since both categories cause inter team coordination problems.

The research also explains the experienced problems by analysing flaws in the respective agile coordination strategies in relation to the application contexts. This approach allows to adjust both agile design practices and or the application context to enable agility in automotive design. It also opens opportunities to expand agile design to further application contexts.

1.5 Structure of the thesis

The structure of the thesis is divided into seven chapters as presented in Figure 1. The following section provides short summaries of the chapters two to seven.

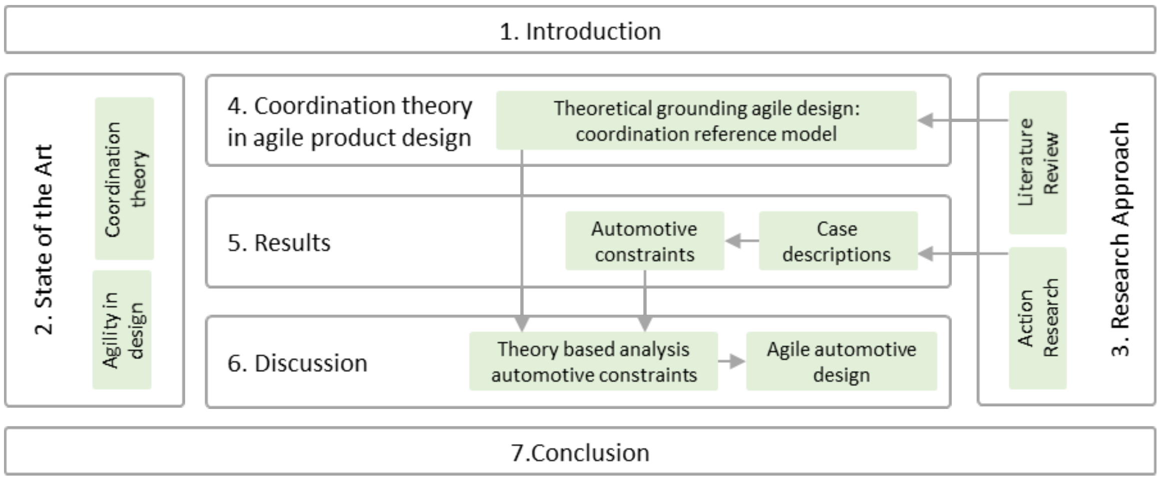


Figure 1: Structure of the thesis. The structure is divided into seven interlinked chapters. The green boxes highlight the most relevant contents, and the arrows reproduce the red line of the research.

Chapter 2: State of the Art. Agility in product design is defined from two viewpoints. Agility as an attribute and agility as a construct. This conceptualization of agility is compared to conventional product development theory in general and automotive design in particular. Known constraints of agility are subdivided into the categories scale and physicality. Coordination theory in product design provides a theoretical background for the thesis at hand.

Chapter 3: Research approach. The scientific strategy is divided into three sections. First, the practical problem is specified, corresponding research questions are deduced, the chosen theoretical lens coordination theory is presented, and the complete research project is summarized. Second, Design Research as a research field is categorized and the respective research challenge between relevance and rigour is outlined. Third, the implementation of the chosen Action Research methodology is presented.

Chapter 4: Coordination perspective of agility in product design. The coordination reference model is constructed to mirror typical agile design and thus coordination structures. Agile coordination strategies of popular agile methods are analysed and mapped to experienced benefits to understand the systematology of agility in product design beyond its straightforward practices.

Chapter 5: Results. The collected data is presented and analysed with a focus on experienced practical problems to the employed agile methods. The section is divided into a bottom-up and a top-down data analysis that cluster the data to reoccurring problems in the first stage and relate them to scale or physicality driven causes in the second stage. The chapter ends with a case-specific differentiation between constraints of physicality and scale in the automotive domain.

Chapter 6: Discussion. Dysfunctionalities of agile coordination strategies in automotive application contexts are deduced and connected to the experienced practical problems. Inter team coordination mechanisms within agile coordination modes are discussed to align agile coordination strategies with automotive design. Furthermore, changes to the product structure and their influences on coordination determinants and agile coordination strategies are depicted. Lastly, the enablement of agile coordination strategies through digital design technologies in automotive design is discussed.

Chapter 7: Conclusion. The main findings in relation to the research questions are summarized. Additionally, the contribution and limitation of the research project are stated, and future research opportunities are outlined.

2 State of the Art

"If I have seen further, it is by standing on the shoulders of giants."
Isaac Newton

This chapter aims to introduce, describe and summarize the state of the art of the relevant design theories and practical knowledge fields necessary for the research project. Agility in product design is defined and compared to conventional design theories. The fundamental product design principles and established methodologies are introduced to establish a scaffolding to delaminate agility in product design. The domain automotive design is introduced, and its characteristics are compared to the requirements of agile design. Especially the impact of the scale of the design process and the physicality of the product on agile design are referenced since they differ from software design. Coordination in product design is the second focus of this chapter. Coordination theories from different fields are presented to provide a broad theoretical understanding of the current research. Practical applications of coordination theories from different research fields related to product design are introduced to show the relevance of the field. Coordination theories that reflect fundamental principles of agile design are connected to create a research model which serves three purposes. First, analyse the established agile methods regarding their empirically proven benefits and provide an explanation of their functionality. Second, analyse the collected data from the agile pilot projects regarding practical constraints due to the automotive domain. Third, establish adjustments to existing agile methods to improve their applicability in automotive design.

The chapter is subdivided into seven subchapters. The first subchapter establishes an understanding of agility from two perspectives. Agility as an attribute defines agility based on its origins, suitable application context, practical interpretations, core concepts and experienced benefits. Agility as a construct defines agility based on the manifesto of agile software development, agile methods and scaled agile frameworks. The second subchapter introduces the established design theory in product design and provides a scaffolding to delaminate agile design from it. The third subchapter describes the domain automotive design and emphasises differences to software design as the ideal application context of agile methods. It includes two subchapters that summarize the literature on agile constraints caused by the scale of the design process and the physicality of the product since both characteristics are unavoidable in automotive design. The last subchapter focuses on coordination theory. It defines coordination and provides perspectives from different adjacent research fields. Based on the coordination strategy concept coordination determinants, coordination modes, mechanisms, and coordination outcomes are connected.

2.1 Agility in product design

“There is nothing permanent except change.“
Heraclitus of Ephesus

Tseng and Lin describe agility as “the business paradigm of the 21st century” (Tseng and Lin, 2011) and Denning claims that “agile is a global movement that is transforming the world of work” (Denning, 2017). Reports from industry underline the relevance of agility as a standard in software development today (Komus and Kuberg, 2020; VersionOne, 2020) and studies show that agility has become the new norm within software development (Gustavsson, 2020b). This success of agility is not limited to software development but continuously expands to other domains (Atzberger, Nicklas, *et al.*, 2020; Komus and Kuberg, 2020; Schmidt *et al.*, 2019).

Despite this undeniable success in practice, it remains ambiguous what exactly agility entails and what it stands for. Even after at least two decades of research agile design is still reported to lack a theoretical core (Baham and Hirschheim, 2021; Rathor et al., 2016). The resulting lack of a unified theoretical understanding of agility has resulted in different views with practitioners and researchers asserting their own understandings, which often differ from each other (Wang et al., 2012). Conboy, therefore, describes agility as a concept that is highly multifaceted and which has been used by different people to refer to different phenomena (Conboy, 2009). In practice, agility is often interpreted as a progressive alternative to traditional, plan-driven product design approaches and not specified comprehensively.

The agile manifesto (Beck and Beedle, 2001) is the most quoted reference to agility. It communicates values and principles like a philosophy and invites to interpret but does not prescribe clear guidelines and practices for product design. Such guidelines are provided in agile methods. But different agile methods focus on different aspects of the manifesto. This divergence within the set of agile methods includes methods that differ clearly (Baham and Hirschheim, 2021) and sometimes contradicting practices (Conboy, 2009). Today a broad set of agile methods exists (Abrahamsson *et al.*, 2002; Edison *et al.*, 2021) with Scrum being the most popular method (VersionOne, 2020). Unfortunately, this variety of methods often results in applications without questioning the method and considering the design context of the applying company. Even worse, self-proclaimed gurus present their methods based on single success stories, easy metaphors or analogies to other fields and not in relation to the existing body of knowledge (Janes and Succi, 2012). Without a sound theoretical foundation new and existing agile methods cannot be evaluated, corrected and or rejected (Strode, 2005). This in turn increases confusion about what agility implies and how to implement it in practice. To resolve this ambiguous understanding of agility, it is described in the following subchapters from two perspectives: agility as an attribute and agility as a construct.

The first perspective defines agility as a characteristic of product design. It starts with the history of agility to depict its origins and the original driver. Definitions of agility across research fields are discussed and a definition for the thesis at hand is chosen based on four essential core concepts. Reported benefits of agility are aligned to the central concepts. At last, agile requirements or “sweet spot conditions” and implications of different application contexts on agility are presented.

The second perspective “agility as a construct” describes principles and practices in product design to enable agility. It describes the Agile Manifesto and connects it to the most relevant agile methods. First, the manifesto is described based on its values. Second, the single team agile methods Scrum and XP are presented and set into relation to the manifesto. Third, characteristics of scaled agile methods and differences to single team agile methods are summarized.

2.1.1 Agility as an attribute

The practice-driven origin of agility has led to a spectrum of approaches, frameworks, and interpretations. This subchapter aims to describe a clear picture of what agility in design is, where it came from and what it implies, based on scientific literature streams. This clear definition is necessary as a point of reference for the following investigation of agility in automotive design. The analysis of different interpretations of agile design and their emergence process also improves understanding of its central characteristics.

The subchapter is divided into five sections. First, the history of agile design is presented and connected to central drivers. The Manifesto for Agile Software Design is put into context of the existing streams in software development and product design. Second, definitions of agility are summarized and compared to find repeating concepts and decide on a common definition for the thesis at hand. Third, based on the definition of agility core concepts are described that further detail the definition. Fourth, change in product design as the essential driver of agility is detailed and explained. Tools to differentiate and concretize the broad concept of change are described and compared. Fifth, based on the established concept of change in product design, ideal conditions for agility are summarized. Sixth, benefits and drawbacks of agile product design from literature are compared and connected to the core concepts and the definition of agility.

2.1.1.1 History of Agility

In February 2001 the Manifesto for Agile Software Development (manifesto) was created by a group of 17 programmers during a two-day workshop in Utah to discuss alternative software development approaches (Beck and Beedle, 2001; Fowler and Highsmith, 2001). The aim of the manifesto was to develop an alternative to the then heavyweight and documentation driven software development approaches. The declaration consists of four values and twelve principles that frame an understanding of agility in product design. These values and principles are discussed in detail in section 2.1.2.1. The essence of the manifesto is the ability of product design to embrace change, to focus on people and direct communication and to continuously focus on customer value. The publication of the manifesto is often referenced as the formal origin of agile product design (Dingsøyr *et al.*, 2012). This view is supported by the facts that since the publication of the manifesto in 2001 agile software development has gained widespread use and has become a standard in software design (Hoda *et al.*, 2018; Venkatesh *et al.*, 2020; VersionOne, 2020).

But the presentation of the manifesto cannot be decoupled from the then already well-advanced discussion about alternative approaches to software design. Abbas et al. provide historical and anecdotal evidence that dissatisfaction with heavyweight software development approaches existed before the manifesto and alternatives were already established (Abbas *et al.*, 2008). Starting in the 1960s large software projects repeatedly overstrained the established software design approaches in complexity and size (Ovesen, 2012). Projects that increased the number of programmers without adapting the structure were called “million monkey approach” since they did not improve project progress, delivery deadlines and quality requirements (van Vliet, 2008). The employed traditional approaches were heavily structured and planned in detail in advance and hence suffered from changing user requirements, changes in technology and environmental uncertainty throughout the project. Abbas et al. question these approaches and point to the contradiction that it was assumed realistic to anticipate complete sets of requirements early in the project lifecycle, even though many changes in requirements and technology occurred throughout the project’s life span (Abbas *et al.*, 2008). The sequential interpretation of the waterfall development approach is often referenced as a negative example to delaminate agility in design from these traditional, linear approaches, which misinterprets the original publication of Royce that already incorporates iterative design cycles (Royce, 1970). In 1968 the term “Software Crisis” was coined to reflect the problems of these approaches with longer and more complex software development projects in highly dynamic application contexts that were an object to continuous change in their product design projects (Kraut and Streeter, 1995).

To answer this crisis alternative development approaches were established by independent practitioners that would later be sub summarized as agile movement and lead to the manifesto. As a result, already in the nineties a number of lightweight frameworks were applied in software design that are known as agile methods today (Dingsøyr *et al.*, 2012). Since these methods were driven individually by experienced practitioners such as Beck, Schwaber and Cockburn their implementations and concept vary largely (Abbas *et al.*, 2008). Nevertheless, these methods have in common that they were based on the idea of product design that accepts and integrates turbulent development environments and change as a delamination to linear approaches (Highsmith and Cockburn, 2001). Although these agile methods were new, the underlying principles and ideas such as incremental, iterative design had been applied before (Abbas *et al.*, 2008). For example, Scrum is based on the publication "The New Product Development Game" of Takeuchi and Nonaka from 1986 long before the manifesto (Takeuchi and Nonaka, 1986). Other early approaches such as the V-model (Rook, 1986) or the Spiral model (Boehm, 1988) integrated similar ideas to overcome shortcomings of linear process models. In summary, agile methods were not as new or revolutionary as often idealized but have origins in older methods (Conboy, 2009).

Nevertheless, since the publication of the manifesto agile product design has changed from an alternative to established design approaches to a standard in software development (VersionOne, 2020). The concept has even been transferred to large scale software projects (Dingsoeyr *et al.*, 2019) and mechatronic product design (Atzberger, Simon, *et al.*, 2020; Komus, 2017). Both transfers are further specified in sections 2.3.1 and 2.3.2 To explain the growing success throughout the last decades despite its ambiguous interpretations the next sections will compare the most relevant definitions of agility and derive central ideas and concepts.

2.1.1.2 Definition of agility

Before defining agility, it is necessary to differentiate the set of terminologies that are connected to it and employed in practice. Several terms such as agile, agile methods and agile product development or design are used to describe different aspects of agility in product design. The attribute agile is described in product design as "an embedded trait or attribute characterized by durability, resilience, speed, flexibility, attunement and preparedness" (Prosci, 2021). The word agile in agile methods was originally spelt in capital letters as the name of the specific group of methods (Gustavsson, 2020b). In the thesis at hand the lower case is chosen to improve readability. Agile methods are a set of frameworks and tools that support and lead to agility in product design through recommended practices. Agile product development or design describes design activities necessary to develop a product according to agile principles and values. Product design refers to the comprehensive activities to realize new products and hence expands the German utilization of the term only in reference to aesthetic design activities. While *development* is more commonly employed in practice, *design* is the more general term in research. For this reason, design is preferred to development in the thesis at hand. Nevertheless, both terms are employed synonymously. Agility as a comprehensive term is further specified in the following paragraphs.

Baham and Hirschberg argue that even though the manifesto of agile software development provides central values and principles to comprehend agile product design it is not a formal definition of agility (Baham and Hirschheim, 2021). It rather provides generic guidelines and value statements instead (Dingsøyr *et al.*, 2012). Strode argues that without a sound definition of agility any author or practitioner can state that their method is an agile method. Aligned to the manifesto none of the most relevant agile methods fulfils all values and principles which clarifies the unsuitability of the manifesto as a practical definition of agility (Strode, 2005). Conboy further underlines this assessment and argues that hardly any two agile methods adopt the same definition of agility. Some agile methods even propose opposing principles such as collective versus individual code ownership (Conboy, 2009). To avoid these unclear and opposing concepts of agility in practice the following section conceptualizes a definition of agility from different research fields (see Table 1) for the thesis at hand. Central aspects of the definitions are combined to synthesize a consistent understanding of agility for the thesis at hand.

Table 1: Definitions of agility across knowledge fields.

"Agility as the ability to both create and respond to change in order to profit in a turbulent business environment."	(Agile Alliance, 2021)
"Agility as the ability of a development team to react rapidly to changes in a dynamic environment."	(Conforto et al., 2016)
"Agility as the ability to balance stability and flexibility."	(Highsmith, 2010)
"Agility as the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment."	(Conboy, 2009)
"Agility as a software development team's ability to anticipate, create, learn from and respond to changes in user requirements through a process of continual readiness."	(Baham and Hirschheim, 2021)

The ability to accommodate change had been the central driver throughout the origins of agility. William and Cockburn state that agile methods and practices were developed to **embrace, rather than reject, change** (Williams and Cockburn, 2003). This early definition of agility emphasises the ability to integrate change. The Agile Alliance further specifies agility as the ability to create and respond to change (Agile Alliance, 2021) and therefore expands Williams and Cockburn's passive concept into an active one.

Conforto et al. define agility in the project management field as the **ability of a development team** to react rapidly to changes in a dynamic environment. (Conforto et al., 2016). This definition specifies the concept of agility to integrate change by means of close teamwork. Baron and Hüttermann underline this team concept of agility and define it as a particular way of thinking and attitude to work, that fosters close collaboration within the team (Baron and Hüttermann, 2010). Lee and Xia combine both aspects of Conforto and Hüttermann and define agility as the software team's capability to efficiently and effectively respond to and incorporate user requirement changes during the project life cycle (Lee and Xia, 2010).

Erickson et al. explain agility as an approach to **strip development methodologies of their heaviness** to enable fast response to changing environments and user requirements (Erickson *et al.*, 2005). Cockburn similarly characterizes agility as being manoeuvrable and fast to respond by means of lightness and effectiveness (Cockburn, 2006). These definitions continue the concept of change integration but emphasize the lightness and manoeuvrability in design especially in contrast to earlier more formal and planning centric design approaches.

Haberfellner and de Weck provide a more overarching perspective and define agility as **the property of a system that can be changed quickly** (Haberfellner and de Weck, 2005). Henderson-Sellers and Serour confirm the system agility concept and extend the agility concept from adjustment to change with the ability to refine and fine-tune development processes as needed (Henderson-Sellers and Serour, 2005). Highsmith also provides a system perspective on agility and defines it as the combination of system flexibility and rapid response (Highsmith, 2009). These definitions expand the idea of agility to a system behaviour that can continuously adjust to answer to dynamic change on a macro scale.

Conboy derives the most exhaustive definition of agility from these definitions (Conboy, 2009). His analysis is based on a structured literature review that integrated agility definitions across related disciplines. This composition of definitions is compared to and delaminated from the concepts of flexibility and leanness in product design. He conceptualizes leanness as the "contribution to perceived customer value through economy, quality and simplicity" and flexibility as "the ability of a ISD method to create change, or proactively, reactively, or inherently embrace change in a timely manner, through its internal components and relationships with its environment" (Conboy, 2009). Combining and going beyond these individual concepts Conboy defines agility as:

"[…] the continual readiness of an […] method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment."

He further emphasizes *"learning from change"* as a central characteristic of agility which has been confirmed by Lyytinen and Rose and further publications (Lyytinen and Rose, 2006). The presented definitions show that the term agility has been employed for a spectrum of design characteristics that are centred around and enable the idea of product design that addresses internal and external change. Central are the ability to not only respond but also create change, the close teamwork, the lightweight product design process and the connected system behaviour. The thesis at hand is based on Conboy's definition of agility. The aspects within the definition of agility allow its delamination from the similar, alternative product design approaches such as *Flexible Product Developmen*t and *Adaptive Product Design*. Flexible Product Development centrally addresses the ability to integrate change throughout the design process but does not account for the quickness and activeness of the reaction (Thomke and Reinertsen, 1998). *Adaptive Product Design* adjusts underlying processes to best suit dynamically changing environments (Meißner and Blessing, 2006) but remains at the system-theoretical level and focuses on processes. Unlike in agile design enabling product or organizational structures are not part of the concepts.

2.1.1.3 Core concepts of agility

Based on Conboy's systematic definition of agility Baham and Hirschheim concentrate the concept of agility as the ability to anticipate, create, learn from and respond to change (Baham and Hirschheim, 2021). They derive four central concepts (see in Table 2) from a review across the most relevant agile methods and frameworks that enable agility in software design. These four concepts are interconnected and reinforcing. This implies that that agile design represents a design system and should not be simplified to the application of individual practices. The core concepts reflect the presented definitions of agility and connect them into understandable conceptualizations. Descriptions of each concept are presented in the following paragraphs.

Table 2: Four core concepts to enable agility based on a review of agile methods and frameworks (Baham and Hirschheim, 2021).

Core concept 1	Inspect and adapt cycles
Core Concept 2	Incremental design and iterative development
Core concept 3	Working collaboratively in close communication
Core concept 4	Continuous customer involvement

Inspect and adapt cycles allow teams to analyse and reflect on design activities and adapt them if necessary. This concept allows to deal with change in design projects. Each design cycle allows to reconfigure assumptions according to results or change in a repetitive learning process (Baham and Hirschheim, 2021). Agile teams rely on inspect and adapt cycles to verify technical implementations and validate their interpretation of user requirements in continuous design cycles. Based on this evaluation design projects adapt their design activities soon and at low costs. The concept of inspect and adapt includes a broad spectrum of use cases ranging from small design activities of individuals to repetitive impersonal automated continuous integration systems to complete design project evaluations. Iterative design accommodates the inspect and adapt cycle concept centrally into design projects as every iteration represents an individual design cycle that provides the opportunity to reflect and adapt progress.

Baham and Hirschheim underline the importance of **incremental and iterative development** and value the concepts to breakdown the development process and product delivery into smaller units as the most fundamental and universal approach to achieving agility (Baham and Hirschheim, 2021). Iterative design divides the design process into continuous, short design cycles to incorporate inspect and adapt cycles within each iteration. Incremental design requires each iteration to result in an increment which presents a part of the product that has costumer value and offers the opportunity to verify and validate the progress of the iteration to a large degree without having to rely on the complete product. Both are essential to the assumption that design progress and requirements cannot be specified completely in advance. Iterations result in increments which represent continuous feedback sources to adapt progress and requirements as design knowledge grows. Short iteration lengths establish quick feedback loops and minimize rework in changing environments (Vidgen and Wang, 2009). Overall, iterative design provides understanding of past issues, sensing of current issues and responding to future issues (Rathor *et al.*, 2016). Teams are enabled to focus on design activities and reduce time spent on interpreting unclear requirements.

Working collaboratively in close communication is the third core concept of Baham and Hirschheim (Baham and Hirschheim, 2021). It paraphrases the teamwork ideal in agile product design. The intensive cooperation in teams realizes open and close communication between designers. Agile teams include the customer as a part of design teams if requirements are dynamic and teams need to readjust often. Agile design teams are self-organized, autonomous and promote mutual participation and teamwork. They collaborate towards a common goal. Teams need to be authorized to make their own decision largely autonomously without complicated management or hierarchy consultation to answer quickly to change. Another important aspect of agile teamwork is the cross-functionality of such teams which further enhances the autonomy of teams (Stray *et al.*, 2018) since team-external input that might block, or slow design activities is minimized.

Continuous customer involvement in design teams is the fourth of Baham and Hirschheim's core concepts (Baham and Hirschheim, 2021). Wood et al. even argue that agile product quality advantages are driven rather by close design team and customer cooperation than better teamwork in design teams (Wood *et al.*, 2013). The intensive exchange between customer and design team in agile design establishes close relations which lead to shared understanding of user requirements. This increases predictability of the product and hence customer acceptance of it (Cohn, 2010). The close connection between designer and customer within a design team allows to directly adjust to changes of both design implementation and the customer requirements. The proven increase in customer satisfaction improves a team's ability to provide business value.

2.1.1.4 Change in design contexts

After having described where agility was established originally, what it stands for, and how it is implemented the concept of change is clearly central to it. This subchapter defines and frames change in product design.

The strengths of agility in product design are most valuable in dynamic design environments. Denning emphasizes the continuous expectance and integration of change in agile design. For him this trait is essential in an increasingly dynamic and unpredictable world (Denning, 2016). Bennet and Lemoine summarize such dynamic product design conditions with the VUCA acronym which emphasizes volatility, uncertainty, complexity and ambiguity in design environments (Bennett and Lemoine, 2015). The VUCA categorization allows to specify the rather general concept of change or dynamic environments in product design that is referenced in the presented agility definitions. Volatility relates to the increasing rate of change and the need for speed to answer to it. Uncertainty underlines the lack of knowledge what kind of change is to be expected and how action might trigger it. Ambiguity implies that cause-effect relations are unclear and therefore impacts of change are not unequivocal. Complexity is defined by the number of elements, their interdependencies and the dynamics of these relations. (Bennett and Lemoine, 2015). To differentiate unpredictable VUCA environments and recommend appropriate actions concepts such as the Cynefin framework or the Stacey matrix have been developed.

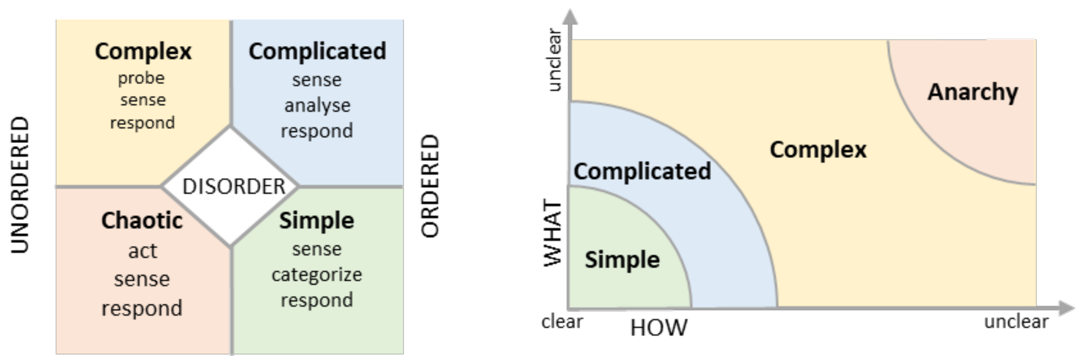


Figure 2: Cynefin framework and Stacey matrix. The Cynefin framework (left) differentiates five problem contexts based on their change dynamic and recommends respective approaches (Kurtz and Snowden, 2003)*. The Stacey matrix (right) differentiates four change situations based on the axes "what" for the problem understanding and "how" for the problem approach* (Stacey, 2007)*.*

To improve problem understanding and categorization within VUCA environments the Cynefin framework (Figure 2, left) differentiates five contexts and recommends specific approaches for each of them (Kurtz and Snowden, 2003). **Simple** and **complicated contexts** are representatives of ordered systems. Within these contexts cause-effect relations are well understood, and existing approaches can be selected or generated based on the given facts. **Complex** and **chaotic contexts** on the other hand represent unordered systems. Complex cause-effect relations are unclear to such a degree that they are understood only in retrospect while chaotic cause-effect relations remain unspecifiable even in retrospect. Both lack transparent and linear cause-effect relations and require an emergent problem-solving strategy. Known and established patterns are applicable to complex contexts. Chaotic contexts require emergent action and cannot apply proven procedures. In both contexts better problem understanding is generated through action and reflection of the corresponding change in relation to existing knowledge. In the complex context, short iterative design cycles are recommended to increase problem understanding incrementally. Chaotic contexts on the other hand are characterized by the need to drive change immediately at large scale and therefore cannot apply structured, iterative design cycles. Besides the ordered and unordered systems, the Cynefin framework also includes a disorder category. This category represents problems which cannot be assigned clearly to one of the four contexts due to opposing or wrong interpretations. Such situations are especially threatening since inadequate action might worsen the unspecifiable situation. The Cynefin framework recommends to break down such situations into constituent parts until individual parts can be categorized to the four known contexts. (Kurtz and Snowden, 2003)

The Stacey matrix (Figure 2, right) also provides guidance on how to approach VUCA contexts (Stacey, 2007). Based on the two dimensions "knowing what" which means clear understanding of requirements and "knowing how" which means clear understanding (and technology) on how to solve a problem the Stacey matrix categorizes situations and recommends suitable, proven approaches. The two dimensions form four situations that require different approaches. Clear definitions of what and how are defined as **simple situations**. Fact-based decision making is applicable and sensible in this context. Within the **complicated situation** either the *what* or *how* dimension are only clarified partially. Cause-effect relations are unclear, and discussions and negotiations are necessary. **Complex situations** are based on increasing uncertainty in both dimensions. Such situations require deviation from existing approaches or initial action to further specify the situation. If both what and how are completely unclear the Stacey matrix defines the **chaotic** (anarchy) **situation**. Conventional approaches are not useful within this area.

Both the Cynefin framework and the Stacey matrix further differentiate the change concept and claim to be able to determine if agile design is suitable. The Stacey matrix emphasizes situation evaluation while the Cynefin framework recommends appropriate action. Their combined categorizations of design situations help to understand whether agile design is suitable and beneficial. In predictable contexts (Cynefin: simple, Stacey: simple) change dynamic is low and agile design is applicable but not needed since more efficient product design methods such as best practices or automatization can be applied. In such contexts the efficacy advantage of agile methods is often outbalanced by their additional cost compared to efficiency focused approaches such as lean development or plan-driven approaches. The complicated and complex contexts are ideal for agile design. Agile design approaches quickly improve problem understanding and parallelly drive solutions in short iterative design cycles that allow fast learning cycles. If problem understanding has been improved to such a degree that it suits the simple contexts, agile methods might be changed to more efficient design approaches. On the contrary, chaotic (Cynefin framework) or anarchic (Stacey matrix) contexts that are completely dominated by change are also unsuitable contexts for agile design since neither the problem nor the solution are understood sufficiently. Instead of iterative design based on learning cycles such environments require direct action and cannot rely on small, agile learning steps.

2.1.1.5 Sweet Spot conditions for agility

Change-driven design environments within the complicated and complex categories of the Cynefin framework are viewed as ideal application contexts for agile design. Another factor is the type of product. Software products present characteristics that suit the agile core concepts ideally. Because of both factors different agile methods emerged from similar contexts in software design that were object to VUCA conditions (Bennett and Lemoine, 2015). Takeuchi, Sutherland and Rigby specify ideal agile project characteristics as complex problems, initially unknown solutions, changing product requirements, modularizable work, direct costumer access and critical time to market (Takeuchi *et al.*, 2016). Even though they avoid a delamination based on the product type most of their characteristics reflect software products. Strode underlines this categorization and states that agile design was developed to cope with change and uncertainty in small teams that de-emphasize traditional coordination mechanisms such as forward planning, specific coordination roles, contracts and extensive documentation, mostly free of pre-defined specified processes (Strode *et al.*, 2012).

The idea of ideal conditions for agile design was first introduced by Barry Boehm (Boehm, 2002; Boehm and Turner, 2004). Kruchten later employs the term "agile sweet spot" which he describes as the ideal conditions of which agile software design practices originated from and where they are most likely to succeed (Kruchten, 2013). This contextualization is based on small, collocated teams around twelve persons and a governance model based on simple rules. He states that little criticality of product safety is necessary for iterative product design and changes even after first customer application. Also, medium or high rates of change are ideal for agile design, since low change rates do not require the high adaptivity of agile design. He continues that in-house business models support the agile principle to maintain the product at the responsibility of a team throughout the design process. Handovers categorically lead to loss of information regarding product, process, and organization. Finally, a stable overall product architecture supports modularization which is necessary to divide larger products into smaller sub products which suit the design capacity of compact teams throughout the complete design process. Changes in the product modularity lead to changes in team responsibility and lead to unspecified interfaces and handovers which result in information losses.

The presented sources clearly show that agility in product design suits dynamic application contexts that are driven by change. Additionally, agile design is most valuable under certain project internal conditions such as small, co-located, collaborating teams, unclear requirements, and stable product architecture. The type of product has a certain influence on both internal and external influences. It therefore also influences the added value of agile design. Especially software products feature characteristics that support agile requirements.

2.1.1.6 Benefits of agility

Benefits of agility in product design have been summarized by several researchers. Campanelli and Parreiras describe in their literature review an increase in quality and enhanced flexibility (Campanelli and Parreiras, 2015). Gustavsson emphasises accelerated time to market, increase in quality and productivity and enhanced flexibility (Gustavsson, 2020b). The 14th annual State of Agile Report says that at least 50% of the respondents value agile design for the following benefits: the ability to manage changing priorities, project visibility, business/IT alignment, delivery speed, team morale, team productivity, project risk reduction and project predictability (VersionOne, 2020). Additionally non-research reports from leading tech companies such as Microsoft (Denning, 2017), Spotify (Kniberg, 2014a, 2014b) or Google (“Google’s Agility”) further underline the competitive edge of agility in product design.

Takeuchi, Sutherland and Rigby derive agile benefits in comparison to traditional management approaches (Takeuchi *et al.*, 2016). They emphasis team productivity and employee satisfaction. They see a reduction in waste caused by redundant meetings, repetitive planning, excessive documentation, and low-value product features. Customer satisfaction increases due to improved visibility, costumer integration and the ability to continuously adapt to changing customer requirements. Agility also improves time to market and predictability of new product design. Mutual trust and respect across organizations are caused by cross-functional teams. Micromanagement is avoided largely which frees management to focus on removing impediments to progress, strategy, organization development and customer exchange.

Schmidt connects agile concepts and central principles to experienced benefits of agile design (Schmidt, 2019). For example, agility in product design allows to adapt quickly to evolving changes through self-organization, team ownership and decentralized decision making. The short and steady iterations enable learning cycles throughout the design process. Stakeholders and designers have a shared understanding of the product due to their close and continuous cooperation. Early costumer value is the result of incremental design since costumer may already profit from individual increments besides the complete product. Team collaboration is significantly increased due to the close team structures. The team stability increases team motivation and team commitment. The cross-functionality of the team allows to concentrate cooperation within and not between teams which simplifies communication structures. Interdisciplinary teams are responsible for a product throughout the design phases and avoid handshakes and communication problems.

Table 3: Summary of benefits of agility in product design across different sources from scientific and popular literature.

	Campanelli and Parreriras 2015	Gustavsson 2020	VersionOne 2020	Schmidt 2019	Takeuchi, Sutherland, Rigby 2016
Design process	Flexibility	Flexibility	Flexibility	Flexibility	Flexibility
		Increased productivity	Visibility	Customer integration	Reduction of waste
			Business, IT alignment	Early customer value	Customer satisfaction
			Project predictability	Improved communication	Trust and respect
Team			Team morale	Team collaboration	Team productivity
			Team productivity	Team motivation, commitment	Employee satisfaction
Product	Increased quality	Increased quality	Risk reduction		
		Time to market	Delivery speed		Time to market

Table 3 gives an overview of the described benefits across the selected literature. Design process flexibility is a benefit across all literature sources. Increases in transparency and productivity have been reported repeatedly and costumer integration and early customer feedback are evaluated very positive. Regarding characteristics of teams team morale or motivation increase and especially collaboration within teams profits from agile design. The product benefits from increases in quality, a faster time to market and reduced risks throughout the design process. These benefits are driven directly by reductions of design complexity and coordination structures both within teams and also between designers and customers. It is important to emphasise that agility in design is not a silver bullet for product development. Its strengths are most evident in change driven design contexts. But central characteristics such as continuous customer integration or weekly team meetings might turn into drawbacks compared to conventional development approaches in well-predictable design contexts.

2.1.2 Agility as a construct

The perspective agility as an attribute emphasized an understanding of the characteristics, the benefits and the system behaviour that are attributed to agile design. The perspective outlined the change in product development that started agile design, and explains how it evolved, what values and principles define it and what core concepts it is based on. In simple terms it provided an exterior view on agile design and depicted the resulting characteristics. Unlike this first perspective the second perspective on agility focuses on structures to realize agility in product design. The aim of this subchapter is to describe how agile values and principles are implemented in rules, practices and frameworks to result in the desired system behaviour. Furthermore, it is presented how these structures are adjusted to each other in reinforcing systems. Put in simple terms again it provides an interior view on agile design. This perspective is necessary to decompose agile product design into smaller structures and their connections which is central for a differentiated analysis of the working principles and adjustments of the systems.

This subchapter first presents the Agile Manifesto for Software Design and analyses its values and principles. It connects these theoretical guidelines with more practice-oriented agile frameworks. The two popular agile methods Scrum and XP are described in detail. The perspective concludes with an outlook on scaled agile methods.

2.1.2.1 Manifesto of Agile Software Development and agile methods

As presented in 2.1.1.1 the manifesto for agile software development is the result of a two-day workshop of 17 designers in Utah (Fowler and Highsmith, 2001). The authors of the manifesto intended to circumscribe and summarize the basic beliefs and philosophy of agile software design in the shape of four values and twelve principles (Beck and Beedle, 2001). The four values listed in Table 4 are basic beliefs that frame agile product design and represent guidelines to desirable project settings. They are designed as tendencies that prefer aspects that reflect agile product design on the left to aspects that represent more conventional product design techniques on the right. The supplement in the manifesto which states "*… while there is value in the items on the right, we value the items on the left more.*" underlines the adjustability of agile product design. Structures are matched to the specific project conditions and should not follow standards blindly.

Table 4: Values of the Manifesto for Agile Software Development (Beck and Beedle, 2001).

Value 1	**Individuals and interactions** over processes and tools
Value 2	**Working software** over comprehensive documentation
Value 3	**Customer collaboration** over contract negotiation
Value 4	**Responding to change** over following a plan

The values reflect the ability to embrace change, the integration of the customer into the design process and the focus on people and communication to improve product design. The twelve principles (Beck and Beedle, 2001) further elaborate the values. They represent more specific working mechanisms and provide guiding rules to implement the values. Gustavsson underlines the importance of the manifesto to create self-organized teams (Gustavsson, 2020b). He states that the manifesto supports team autonomy and trust which enable teams to make the right decisions, solve problems and deliver results in accordance with literature that confirms the value of autonomous teams to the success of agility in product design (Stray *et al.*, 2018).

What sets the manifesto apart is its comprehensive approach. Agility in product design is neither only about responding to change, customer integration or autonomous teams. Instead, it is a systematic approach that combines and connects these aspects. The manifesto summarizes the basic values that create a foundation to continuously develop practices and methods to implement individual aspects of agile design. When new aspects are necessary the manifesto gives guidance how to evaluate and implement them. Partial implementations or shallow adoptions (Gregory *et al.*, 2015) are avoided by the comprehensive approach of the manifesto.

The manifesto hast been discussed repeatedly in literature. Different interpretation have been criticised and specific values and principles questioned (Laanti *et al.*, 2013). While some aspects are very specific other elements remain vague. Dingsøyr et al. regard the manifesto not as a formal definition agility but rather as guidelines for delivering high-quality products in an agile manner (Dingsøyr *et al.*, 2012). They view the manifesto as a foundation for methods and practices that improve customer value in accordance with the core concepts of Baham and Hirschheim (Baham and Hirschheim, 2021).

While the manifesto creates a common ground for agility in product design it does not provide specific practices that can be implemented directly by design teams. Its direct application as rulebook is nonsensical since application contexts differ and practices that are valuable in one context might be harmful in another. Instead, agile product design relies on a set of agile methods (or frameworks) that combine practices and rules for particular goals and specific application contexts. The manifesto describes the agility of a system while agile methods realize this agility by concrete and applicable practices and principles. Dingsøyr et al. state that agile methods need to reflect the core values of the manifesto (Dingsøyr *et al.*, 2012). This does not imply that all agile methods have been designed with the manifesto in mind since some agile methods such as Scrum are actually older than the manifesto.

In contrary, the manifesto has been composed as a summary of the most valuable concepts of agile methods that have been proven empirically. Abbas et al. coin agile methods as an umbrella term for well-defined methods that vary in practice and reflect the manifesto (Abbas *et al.*, 2008). The term method in agile methods is debateable since in conventional product development terminology XP or Scrum would be methodologies and their practices such as the Sprint review would be methods (Pahl and Beitz, 2021). Nevertheless, the popular agile method term is well-established in both practical application and the research community and therefore also employed in the thesis at hand.

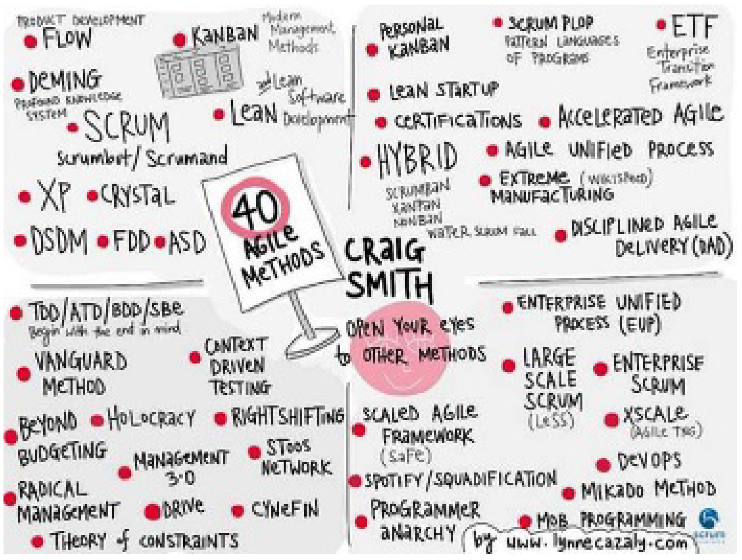


Figure 3: Summary of agile methods (Denning, 2016).

Most agile methods were originally intended for small teams but their popularity has expanded their use to large design projects and beyond software development (Gustavsson, 2020b; Xu, 2009). While agile methods were applied to different application context, they also have been adjusted to them. Additionally, new methods have been introduced that are based on popular agile methods or cannot be traced to existing ones. This evolution has led to a large set of agile methods as shown in Figure 3. The popular agile methods Scrum and eXtreme Programming XP (Baham and Hirschheim, 2021; VersionOne, 2020) are presented in the following sections to provide details how agile methods are structured and applied in practice.

2.1.2.2 Scrum framework

Scrum is by far the most used agile method (VersionOne, 2020). Its origins can be traced to the 1986 article ”*The New New Product Development Game*” (Takeuchi and Nonaka, 1986). Based on empiric data Takeuchi and Nonaka describe successful product development as a result of continuous interaction of designers in small, multidisciplinary teams that are responsible for the complete product development process of a product. The close collaboration of the team is compared to the Scrum of rugby players. In 1995 Sutherland and Schwaber presented the first version of the Scrum Guide at the OOPSLA conference based on their experiences in several product design projects and regular updates have led to the 2020 version (Schwaber and Sutherland, 2020) which is used for the thesis at hand. Scrum is guided by the principles transparency, inspection and adaption and relies on the values commitment, courage, focus, openness and respect (Schwaber and Sutherland, 2020). The framework is based on three roles, five events (meetings) and three artifacts to structure product design.

The **three roles** in a Scrum Team are a team of cross-functional developers, one Scrum Master SM and one Product Owner PO. Developers (designers) are responsible for delivering the increment. This includes responsibilities for creating and adjusting the Sprint Backlog (Sprint Plan), the quality of the increment, and holding each other responsible. The **Product Owner** is accountable for the value of the product. This includes the duty Backlog management in the form of creating, prioritizing and transparently depicting Backlog items which must match customer requirements and reflect stakeholder restrictions. The **Scrum Master** is accountable for the Scrum Team's effectiveness. Her responsibilities are establishing the Scrum framework, coaching the Scrum Team, removing impediments, and ensuring the Scrum events. The Scrum Master serves both the developer team and the Product Owner and facilitates their responsibilities.

The **five events** are the Sprint, the Sprint Planning, the Daily Scrum, the Sprint Review and the Sprint Retrospective. The **Sprint** is a repeating timebox of several weeks with a consistent length during which the Scrum Team generates the product increment. In the **Sprint Planning** the Scrum Team decides on which Backlog items the Scrum Team will work during the next Sprint. It is divides into two parts with and without the Product Owner and results in the Sprint Backlog. The short **Daily Scrum** the team allows to coordinate activities for the following day. The **Sprint Review** allows the Scrum Team to review and discuss the increment after the Sprint together with costumers and stakeholders and collect feedback which get documented in the Backlog. In the **Sprint Retrospective** the Scrum Team reflects the past Sprint and identifies adjustments to the collaboration process.

The **three artifacts** are the Product Backlog, the Sprint Backlog and the Increment. The **Product Backlog** is a prioritized list of items that reflect options to further develop the product according to customer requirements. The **Sprint Backlog** is a subset of the Product Backlog items chosen for a Sprint by the Scrum Team. The **Increment** represents the implemented Backlog items into a subproduct during a Sprint with a value to the customer. The Product Owner decides whether to release it.

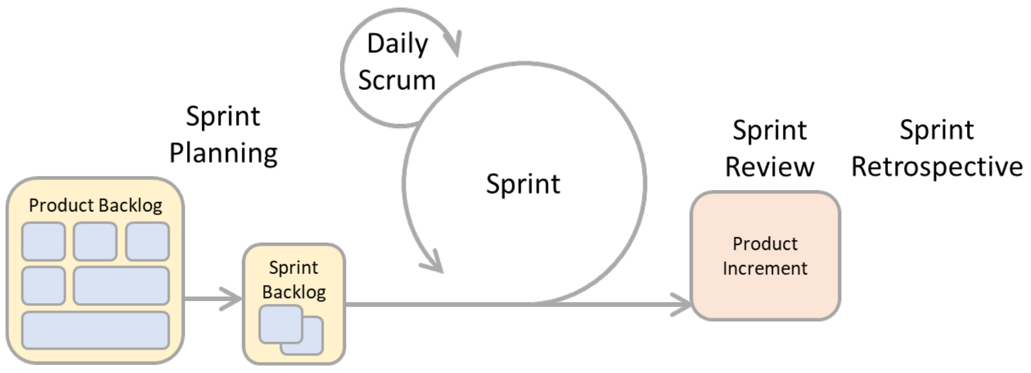


Figure 4: The agile method Scrum. The framework is based on iterative Sprint cycles that connect artifacts and meetings. In the Sprint Planning prioritized items from the Product Backlog are selected for the Sprint Backlog to be developed during the Sprint. Throughout the Sprint Daily Scrum meetings are used to improve cooperation and ensure coordination within the team. The Sprint results in the Product Increment which is presented in the Sprint Review meeting. A Sprint is completed by the Retrospective meeting to continuously question and improve cooperation.

Figure 4 depicts how the presented artifacts, and events interlock during one iteration (Sprint). In the Sprint Planning the development team selects the highest prioritized items from the prepared Backlog into the Sprint Backlog for the upcoming Sprint. During the Sprint the developers focus on the development of the increment while the Product Owner updates the Backlog. The Scrum Team meets every day for the Daily Scrum to coordinate progress of the Sprint Backlog and avoid impediments. At the end of the Sprint the increment is presented and discussed at the Sprint Review amongst the Scrum Team which can also include customers and stakeholders. The increment represents the sum of the selected items of the Sprint Backlog. Feedback and discussion help to adjust the Product Backlog. The Sprint Retrospective represents the end of the Sprint and offers the opportunity to reflect on the past Sprint regarding collaboration and overall progress of the Scrum Team. The Agile Alliance defines Scrum as empirical since teams continuously establish a hypothesis, test it, reflect on the experiment and adjust the product accordingly (AgileAlliance, 2021a).

2.1.2.3 Extreme Programming XP framework

Extreme Programming or XP is almost as popular as Scrum (VersionOne, 2020). It was first applied in the mid 1990's in the Chrysler Comprehensive Compensation program in cooperation with Kent Beck (AgileAlliance, 2021b). Kent Beck also provides a thorough introduction and explanation of the agile method XP in (Beck, 2004). Unlike Scrum XP provides very specific engineering practices and does not detail the overall structure of the design cycle. But it is also based on iterative design cycles that connect coding, testing, listening and designing phases. Close teamwork and the integration of the customer into the design team are like Scrum but role definition is less formal. The engineering practices are adjusted to software development and therefore cannot be transferred to other product contexts without difficulties.

Don Wells recommends XP under dynamically changing requirements, high risks caused by new technologies, small co-located development teams and if automated unit and functional tests are applicable (Wells, 2021). Especially the last condition shows that XP unlike Scrum relies on highly automated IT infrastructure. XP is based on the values communication, simplicity, feedback, courage and respect. The XP practices present the core of the agile methods. Initially twelve practices were published but these practices have been adjusted by Kent Beck in the second edition of his book (Beck, 2004). The descriptions in the thesis at hand are based on practices from both versions.

In XP the customer is expected to be available and part of the design team. The **On-Site Customer** practice reflects this requirement. This enables understanding of customer requirements, direct feedback, customer testing and accelerates necessary decisions for small releases. According to the **System Metaphor** practice parts of the software product get names that are easy understandable by all stakeholders including the customers. This practice is connected to user stories. They significantly improve communication between customer, designers and further stakeholders. The Planning Game practice determines the release and the iteration planning. During the Release Planning the customer presents the desired features and the designers estimate their difficulty. They implement an initially imprecise release plan that is continuously adapted according to project progress. During the Iteration Planning the designers break down features into tasks and estimate their effort based on their experience from past iterations. The planning steps transparently display project and product progress and enable to customer to adjust project steering accordingly. The **Continuous Integration** practice requires code change to be immediately tested when added to a larger code base. It allows to detect integration issues directly. It requires a code integration system, Coding Standards, the Ten-Minute Build and Test First Development to function. The Ten-Minute Build stands for the ability to automatically build the whole system and run all the tests in ten minutes. It requires an automated build process which has to be provided or adjusted by the design team. The short time frame is necessary, since a longer time would result in avoided builds if only small changes are applied. It supports the practices Continuous Integration and Test First Programming. The **Test-First Programming** practice reverses the normal path of software design from develop code, write tests, run tests to write failing automated test, run failing test, develop code to make test pass, run test. It reduces the feedback cycle for designers to find and resolve bugs and therefore improves quality of the product. The **Refactoring** (or Design Improvement) practice focuses on removal of duplication and improving cohesion of the code. It therefore lowers coupling between pieces of codes. It enables simple design for software and therefore improves product quality and product design efficiency. It requires testing practices and continuous integration. The **Collective Code Ownership** practice allows any designer (or pair of designers) to improve any code at any time. This improves code quality and ensures that requires features are put in the right place by the responsible designers. To avoid unreflective code changes the Collective Code Ownership practice relies on the Pair Programming, the Coding Standards and the Testing practices to provide added value. The **Coding Standard** practice results in code throughout the whole systems that could have been written by one competent designer. It does not imply one standard across industries but requires all connected code to look familiar. This also requires a commitment of all responsible designers. The **Pair Programming** practice requires all product software in XP to be coded by two designers, sitting next to each other at the same computer. It guarantees direct review and feedback and results in better design, testing and code quality. Like in Scrum the **Incremental Design** practice divides the whole product in several sub products that can be designed within iteration time lengths. The practice creates the opportunity for customer testing, Small Releases and reduces costs of changes. It requires modular system design and customer integration. The **Small Releases** practice includes the release of small, tested and functional packages to both the costumer and the end users after each iteration. The practice relies on Incremental Design, Continuous Integration and automated testing. It improves product quality through additional end user feedback and improves customer value.

2.1.2.4 Large scale agile methods

Throughout the last two decades a set of large scale agile methods has been developed (Larman and Vodde, 2009). Dingsøyr et al. divide these scaled agile methods into two generations (Dingsøyr *et al.*, 2021). **First generation large-scale agile methods** combine agile methods at team level with traditional project management frameworks such as Prince2 (Bentley, 2005). They connect conventional engineering approaches that provide structure and orientation and might include several phases over long-time spans with agile design practices at the team level. These combinations are often called hybrid frameworks (Bick et al., 2018). These process-centric frameworks often divide work into phases, rely on formal communication and individual roles. **Second generation large-scale agile methods** replace the management frameworks with agile and lean structures. Edison et al. group the most relevant large scale agile development frameworks which include the Disciplined Agile Delivery (DAD) framework, the Large Scale Scrum (LeSS) framework, the Scaled Agile Framework (SAFe), the Scrum-at-Scale framework and the Spotify Model framework (Edison et al., 2021). These second generation frameworks shift the focus from the process towards the product and embrace concepts such as informal communication or an collaboration oriented management style clearly based on agile principles (Baham and Hirschheim, 2021). The focus of the thesis at hand is second generation large scale agile methods.

SAFe is currently the most popular large-scale agile method and employed in software and non-software product design (VersionOne, 2020). Like LeSS it is based on the agile method Scrum and adds further practices, roles and structure to enable cooperation of several teams and connect long-term planning and strategy with agile practices at the team level. The concept of the framework was designed to combine agile software development, lean product development and systems thinking. Origins of SAFe were first presented by Dean Leffingwell in 2007 SAFe was first presented in 2007 (Leffingwell and Kruchten, 2007) and is described in detail at the framework homepage (SAFe, 2021). SAFe has been criticized by researchers (Alqudah and Razali, 2016; Stojanov *et al.*, 2015) and practitioners including Ken Schwaber one of the framework originator of Scrum (Schwaber, 2013) who claim that SAFe minimizes team autonomy and rebuilds bureaucracy similar to conventional product design methodologies like the waterfall model or first generation large-scale agile methods. Regarding this criticism LeSS offers an alternative product design approach for more than one agile team but requires much less structure compared to SAFe. Further information regarding inter team coordination in large scale agile methods is discussed in subchapter 4.4.

2.2 Product development theory

The aim of this subchapter is to supplement a conventional perspective on product development and respective methodologies independent of product type characteristics. It includes different perceptions on how to describe product development or specific aspects of it. This classification of conventional product development understanding is introduced as a scaffolding that allows to analyse agile product design structure in comparison. The reference frame allows to define parallels and differences between agile and conventional approaches to product development and hence increases understanding of agile design. The subsection first introduces a definition of product development and summarizes its elemental characteristics. Thereupon micro and macro logic in product development are divided and the concept of a design project is differentiated from product development. Next, the most relevant process models are described and categorized into linear and iterative (incremental) models. Finally, agile design is categorized according to the presented scaffolding and compared to popular process models.

Ulrich and Eppinger define **product development** as the sum of all necessary activities from sensing a market potential to a product model that is subsequently used for production and sales (Ulrich and Eppinger, 2015). Hammer extends this definition to the product development process which he defines as a series of interrelated activities that give rise to a valuable result for the company (Hammer, 2001). Ehrlenspiel et al. specify these activities as including all operations, from the product idea to the start of the production (Ehrlenspiel and Meerkamm, 2013). Paetzold et al. describe the (mechatronic) product development process by four characteristics (Paetzold et al., 2017). First, development processes are object of uncertain and incomplete data and information. Throughout the development process uncertainty is reduced and data is generated to fill these gaps. Data and information about the product arise in the context of the development activities. Second, development is multidisciplinary and requires the cooperation of different domains that rely on different process and integration models. These models result in varying perspectives of the same product development process which results in variance regarding knowledge content of the models. Third, development activities in these different domains and their respective departments are parallelized. These concurrent development streams mutually require input from each other. Interface management is necessary to ensure data and information availability through any stage of development. Fourth, development consists of a permanent exchange between synthesis and analysis. This requires appropriate analytical steps of information and continuous corrections of requirements according to results.

To further differentiate product development Paetzold et al. separate a micro and a macro logic (Paetzold et al., 2017). The **micro logic** in product development describes the activities at the concrete project work (Gausemeier et al., 2004). It is based on a generic problem-solving cycle. Ehrlenspiel describes it as a continuous sequence of task clarification, solution generation and solution selection (Ehrlenspiel and Meerkamm, 2013). The concept represents the foundation of iterative product development which is essential for the micro logic in product development. The micro logic opens a generic approach independent of the product that applies to a broad spectrum of individual tasks. The **macro logic** is its counterpart and defines integrating structures in product development that connect and guide activities and methods at the micro logic level. The macro logic of product development relies on process models such as the Waterfall model or the Stage Gate model which are described in detail in the next section. These prescriptive process descriptions organize product development from the first idea to the finished product based on a chronological and logical order of development activities (Lindemann, 2009). They are configured to certain product groups but require further adjustment according to the specific product (Paetzold et al., 2017). These variations explain domain specific development processes that are not necessarily interlacing in multi domain development. The process models divide the overall development process into manageable sections that address different goals of product development. Paetzold et al. identify four domain-independent phases: Planning Phase, Conceptual Phase, Detailing Phase and Integration Phase (Paetzold et al., 2017).

In practice product development is organized in projects that connect micro and macro logic development activities. **Development projects** are “temporary endeavour undertaken to create a unique product, service or result” (PMI, 2008). Lévárdy and Browning view product development projects as systems that transform input factors such as project members, artifacts and information into the product (Lévárdy and Browning, 2009). The project is a temporary system of these interconnected input factors that results in a description of how the product works, looks, gets manufactured, is operated, etc. which Lévárdy et al. term the “product recipe”. They emphasise that each development project is unique because activities necessary to reach the project goal and the goal are dynamic, uncertain, and ambiguous.

The development project and the development process have severe influence on the different aspects of the complete life cycle of the product. Schmidt summarizes project management and systems engineering as the essential perspectives on product development that focus on both micro and macro logic of product development (Schmidt, 2019). **Project management** involves administrative activities which include planning, organizing, coordinating controlling, steering and reporting amongst others. On a micro level project management continuously compares theoretical targets with practical results to derive necessary means to remain on track (Kerzner, 2009). Project management organizes product development activities according to cost and quality requirement. The project manager and project management methods are the implementation of the objective in either an individual role or a micro logic process description. On a macro level project management is often attributed to process models such as the Stage Gate model which might include corresponding roles. **System engineering** views product development as a system consisting of elements such as staff, capabilities, information and artifacts, interdependencies and dynamics (Lévárdy and Browning, 2009). It aims at an integrative view on product development that considers operation, cost, schedule performance, support, test and production. System engineering is based on the conviction that a system is more than the sum of its components. The interplay of dependencies and dynamics between elements of product development is essential in the system engineering view. This macro scale product design is described by the elemental engineering design process by Pahl and Beitz (Pahl and Beitz, 2013) which gives product independent guidance on how to generically design a (mechatronic) product.

2.2.1 Linear and iterative process models

Context and product specific process models divide the product development process into smaller, better predictable phases with specific foci on the macro logic scale. Process models are categorized into linear and iterative (incremental) models. In the following paragraph these categories of process models are explained along with descriptions of the popular representatives.

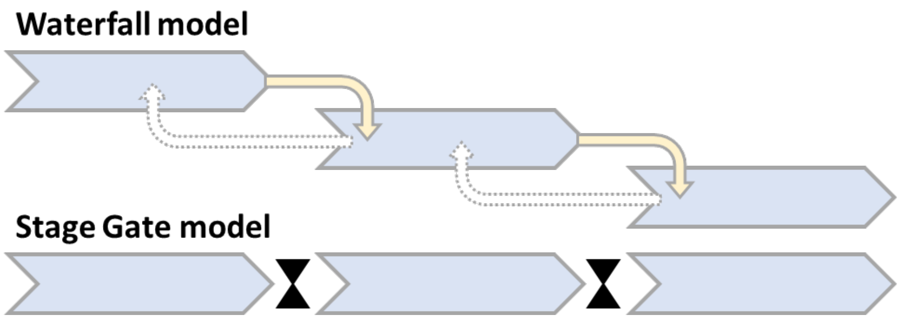


Figure 5: Waterfall and Stage Gate models. Linear, or sequential process models divide the development process into separated phases which are executed consecutively. The waterfall model consists of the phases initiation, analysis, design, construction, testing, deployment and maintenance which flow into each other like a cascade. The original publication (Royce, 1970) *includes iterative design cycles which are represented with dotted arrows. Cooper’s Stage Gate Process adds prespecified verification gates between stages to ensure product maturity* (Cooper, 1983)*.*

Linear or sequential process models divide the product development into separated phases that are executed consecutively (see Figure 5). Extensive planning of the development project throughout the first phases which is often disparagingly labelled as “front loading” characterizes these models (Thomke and Fujimoto, 2000). Verification at gates between these phases ensures that the planned product maturity has been accomplished. Such gates either prolong the earlier phase, stop the whole project or allow the project to continue. Linear process models are one-way process models. Repetitions or going back to earlier phases were originally not intended. The most popular linear process models are the Waterfall model (Royce, 1970) and the Stage-Gate model (Cooper, 1983). The Waterfall model is often used in software development. It is named after progress in product development through its sequential stages that resemble a cascade that flows from one phase to the next in one direction. Its phases are initiation, analysis, design, construction, testing, deployment, and maintenance. Even though the Waterfall model is often referred to for linear process models the original publication of Royce from 1970 includes iterative elements that allowed repetitions of stages (Royce, 1970). The Stage-Gate model adds formal gates between its sequential design stages that ensure that product maturity complies with the initial planning (Cooper, 1990). Gates results either in the next project stage (with conditions), project halt until further decisions or project kill. Gates require powerful steering committees able to thoroughly verify project progress. Concurrent Engineering is another linear process model with characteristics overlapping phases (Haberfellner and de Weck, 2005). Its main advantage is a shorter project duration which results in a high popularity in automotive design, even though the approach risks incomplete or faulty phases.

Iterative process models rely on iterative feedback loops to develop both requirements and product design in parallel. Unlike in linear process models requirements are adjusted throughout iterations and are not fixed after the initial phase. General specifications are transformed during iterations into specific requirements and the product in corresponding subsystems. Within these subsystems detailed product designs are developed and tested that require further system testing and integration. Iterative process models release the complete product. The V-Model and Spiral model (Boehm, 1988) are popular iterative process models. Originally a software development model the V-Model variation described in the VDI Guideline 2206 (VDI 2206, 2004) applies to mechatronic system design. The V-Model connects design and test activities in large iterations (see Figure 2). First, unspecific macro requirements are matched with respective test structures. As the requirements are specified so are corresponding test and integration specifications. Once requirements refinements are sufficiently specific, they are get designed and tested at the lowest part of the V. The refinement of requirements and system characteristics represents the first branch of the V. The resulting product parts are tested and integrated according to the prespecified verification and validation structures climbing up the second branch of the V. It is essential that the findings about product characteristics on different integration levels are used to adjust both earlier requirements and the respective testing and integration system. In automotive development projects the V-model is repeated with varying level of detail und changing overall design goal (e.g. concept, design, industrialization) with each repetition representing one iteration.

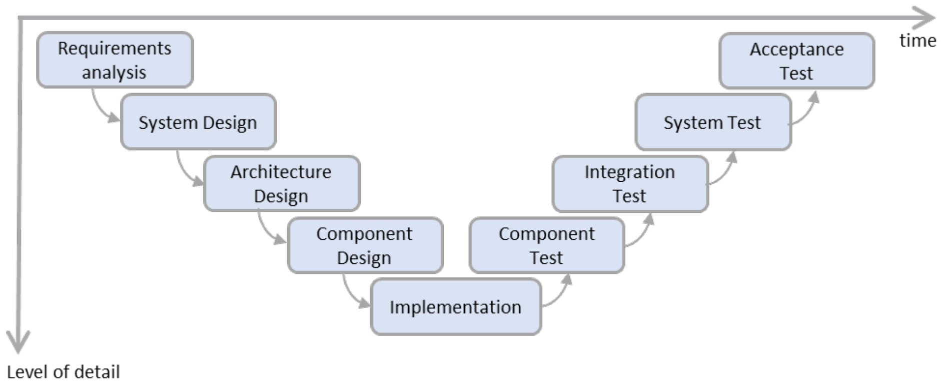


Figure 6: The V-Model specifies interlocking granularity levels of design and respective test activities (based on (Eigner, 2021)*).*

Iterative incremental models are based on the concept of iterative process models. They are based on much shorter iteration lengths including verification and validation at system level ideally each iteration to address unclear requirements or changing customer wishes during each iteration. Their delamination to iterative process models is based on their ability to release product parts with a high customer value every iteration. Typical examples are the agile methods Scrum (Schwaber and

Sutherland, 2020) or Extreme Programming (Beck, 1999) which are described in detail in 2.1.2.2 and 2.1.2.3.

2.2.2 Categorization of agile product development

Agility in product design has been approximated though two perspectives in the thesis at hand. It has been described as an attribute of product development explaining what agility is, what characteristics it results in, what its benefits are and where it came from and as a construct explaining what methods and practices are used to accomplish the desired product development characteristics. The aim of this section is to classify agile product design within the described scaffolding of product development methodology. This allows to compare this new product design approach with existing concepts.

The Fuzziness model of Oestereich and Weiss (Figure 7) summarizes the logic of agile product development straightforward (Oestereich and Weiss, 2008). Initially, the product development project is object to high uncertainty. Imprecise customer requirements and incomplete technology understanding result in a fuzzy solution space. To answer this uncertainty the development team projects its incomplete requirements into a draft of the desired product as an entrepreneurial vision including a description on how to realize it. This initial product vision shapes the scope of the first iteration which must result in an increment. The increment allows to validate and specify the initial product requirements with the customer and to verify the planned product implementation and technology roadmap. The learnings during the iteration help to refine the product vision and result in an adjusted scope for the second iteration. For each following iteration the development team adjusts the scope to improve customer understanding and product maturity. With every iteration the solution space definition improves and the initial uncertainty decreases. This results in a project path that is incrementally reshaped and verified by the outcome of the iterations (Douglass, 2016) and might lead to unexpected solutions. Ostereich and Weiss state "*that the clarity about the product to be produced does not suddenly arise, but comes gradually, and that the goals is not a constant size, but can change over time*" (Oestereich and Weiss, 2008).

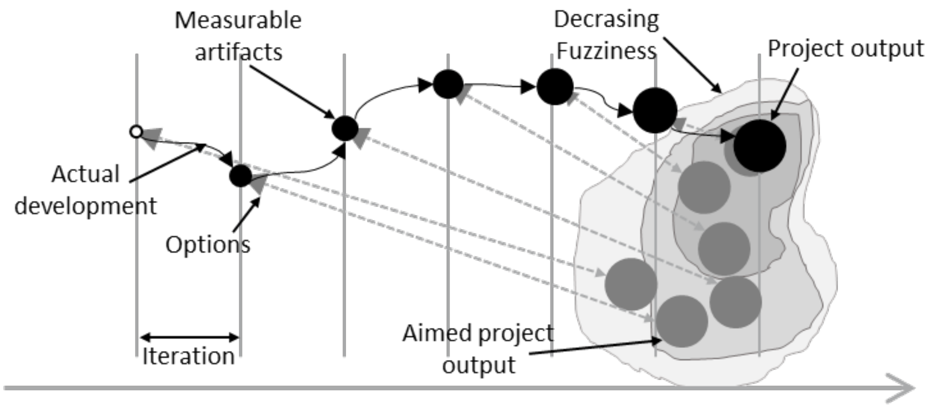


Figure 7: Fundamental logic of agile product development based on the Fuzziness model of (Oestereich and Weiss, 2008)*. Initially the design project is object to uncertainty and can only project the desired outcome vaguely. During the following iterations, which all result in verifiable artifacts, understanding of the product improves in steps. The clarification of the product requirements and the verified design solutions iteratively decrease the design fuzziness.*

Regarding the **micro logic** of product development scaffolding agile product design is also based on the elementary problem-solving cycle (Ehrlenspiel and Meerkamm, 2013) like most other approaches. Any iteration follows the same procedure: Determination of the iteration scope (plan), generation of the increment (do), verification and validation of the increment (check), transfer of learnings to the next iteration (act). Unlike conventional approaches agile design emphasizes the **independency of teams** and hence the micro design cycles. This independence on the micro logic design level is an essential principle. Agile design emphasizes iterative design, short iteration length, consistency of the iteration length and incremental results on a micro logic level. The combination of short iterations and incremental results allows to both validate requirements with the customer and verify implementations in parallel which differs significantly from traditional approaches. Linear approaches require a complete and correct set of requirements early in the design process. Another significant characteristic of agile design is its dominance of the micro logic compared to the macro logic. Iterative learnings are more relevant to the solution space definition than initial planning procedures. Most traditional design methodologies subordinate the micro logic to the macro logic.

Regarding the **macro logic** according to the presented product development scaffolding agile product design best matches the **Spiral model** of Boehm (Haberfellner and de Weck, 2005). Unlike sequential process models agile design allows to change requirements and plans throughout most of the development project depending on the product type characteristics. The focus on product increments to verify product implementation and validate costumer requirements classifies agile design approaches within the subgroup of iterative incremental process models. Besides the described categories agile product development has further differentiating characteristics. Nerur and Balijepally summarize the spectrum of differences between traditional and agile product development and emphasize different goals. Agile development aspires adaption, flexibility, and responsiveness through an emergent, iterative and exploratory design process, while traditional development aims at optimization and employs an deliberate, formal and linear design process (Nerur and Balijepally, 2007).

2.3 Automotive product development

In this subchapter automotive design is described according to the presented product development scaffolding. Additionally, the development context is compared to sweet spot conditions of agile design and central differences are categorized. This allows to better understand the suitability of agile design structures in the context automotive design. The empirical scope of the thesis at hand is the Research and Development department of the partnering company BMW Group and therefore the reference for automotive design in this section. The development department is separated into multiple sub departments or divisions according to product functionalities. The divisions cooperate in large development projects. These projects are divided into process chains according to a generic product architecture. The product architecture separates systems into modules and modules into components based on specific parts of cars. The organizational breakdown of the development department references the product architecture. Divisions are separated into units and groups which are responsible for modules and components.

The **macro logic** of large automotive design projects is structured through a stage gate process model that separates ideation, conceptualization, series design and industrialization phases. Gates are implemented between each of these phases to assess product maturity and business model viability according to a prespecified plan. Each evaluation requires a complete integration of the product which has been developed in modules and components according to the predetermined product architecture. That is why between each stage automotive design relies on at least one complete cycle of the V-Model. Objectives, product maturity and level of detail of the V-Model cycles change with the phase of the stage gate process. Consequently, the automotive design process is a combination of an integrating linear Stage gate and an iterative V-Model process model. The overall V-Model cycle is subdivided into several modules and components that rely on shorter iteration lengths. The iterative V-Model cycles throughout the phases are not incremental since only the final product is released to the costumer. An exception within automotive design is software only products which are directly released as incremental updates into hardware that is already in use. The **micro logic** of automotive design includes a broad spectrum of design activities which reflects the diversity of product types within the automotive product. At the micro logic level these design activities are highly interdependent, even across product types. These dependencies are driven by the high integration level and the physicality of the product. Individual teams depend on input from other teams and decisions within teams influence multiteam systems. Prototypes at a high integration level are applied to manage the complexity of these interdependent design activities. They allow to verify the system behaviour of the product and transparently depict interdependencies between product parts and hence the respective development units.

The highly integrated automotive product development process is implemented in large projects. Several projects are conducted with an offset to avoid simultaneous use of development resources. A vast interlinkage of functional, physical and sequential dependencies causes a high interdependency level of organization units within these product development projects. Significant coordination efforts are necessary to facilitate project progress and avoid asynchrony between sub projects. Vertical hierarchies and specific role descriptions of both technical and project management roles are characteristic for the coordination structure. Coordination therefore accounts for a major part of the product design effort.

Regarding the **nature of the product**, automotive product design includes separated software and hardware development projects and hybrid or mechatronic development projects. Product integration and testing is implemented differently in hardware and software design projects. In software design automated integration and testing has been implemented to a large degree. Most of the hardware design on the other hand is based on sequential testing of physical prototypes. The integration level of these physical prototypes depends on the analysed functional dependencies and overall (phase-based) product maturity. These verification cycles and the necessary infrastructure amount for a large share of the overall design process regarding both time and effort. The more time intensive hardware integration dominates overall design scheduling. The interlinkage between hardware and software in the highly integrated design process results in suboptimal project frames for software specific subproducts. The combination of an integral product architecture and the slow hardware dominated product integration is vulnerable to undetected, unexpected, or peripheral problems.

Automotive design differs considerably from agile sweet spot conditions as described in 2.1.1.5. The two central factors that differentiate automotive design are the physicality of the product and scale of the design process. The physicality of the product results in additional dependencies between components and hence increased independency of design units compared to agile sweet spot conditions. It affects the product verification strategy which relies on highly integrated physical prototypes. It increases the verification efforts, complicates redesigns, and limits verification automatization. The physicality of the product also requires up-front specification of the product architecture and early determination of central design concepts before the start of the actual design process. The scale of the development projects results in large systems of teams. Instead of independent design teams like agile sweet spot conditions, interdependent teams cooperate in multiteam systems. Central drivers of the large scale in automotive design are the product complexity and size and thus the number of product parts that need to be developed in parallel and the spectrum of necessary specializations. To understand the impacts of the characteristic scale and physicality on agility in automotive design both factors are further detailed in the next subchapters.

2.3.1 Agility in mechatronic product design

The aim of this subchapter is to detail and summarize constraints of agile design caused by the physicality of hardware products. The reason agility in mechatronic product design is described is that agile automotive design differs from agile software design regarding the physicality of the product. These differences are essential to understand the implications on agile automotive design. The subchapter is divided into two parts. First, hardware products are differentiated from software products regarding their design process and fundamental product characteristics. These differences to agile sweet spot conditions are connected to drawbacks of agile design of non-software products. Second, literature sources are presented that provide cause effect relations between hardware design characteristics and constraints of agile design.

The term "mechatronics" is a composition of the words "mechanics" and "electronics". This combination reflects that a mechatronic product consists of mechanics, electronics and software and integrate them into one product. A mechatronic product is always based on a hardware but must not necessarily contain a software share. Such products require the cooperation of several disciplines in product design that often result in challenges due to opposing product design approaches (Lückel *et al.*, 2000). Computer scientists and electrical engineers are function-oriented and adjusted to short product lifecycles while mechanical engineers are component-oriented and used to much longer product lifecycles. Their combination in mechatronic product design results in numerous dependencies between the disciplines.

Socha and Walter fundamentally differentiate product design in software and physical products by relating activities to effort needed (see Figure 8) to analyse applicability of agile design in non-software products (Socha and Walter, 2006). They divide the product lifecycle of hardware and software products into the sequential phases Design, Build, Distribute, Intervene and Operate. The Design phase is central in software products while their non-physicality allows to minimize the Build and Distribute phases. Hardware products on the other require additional activities for both the Build and Distribute phases. Also, hardware design relies on sequential design phases that are interdependent.

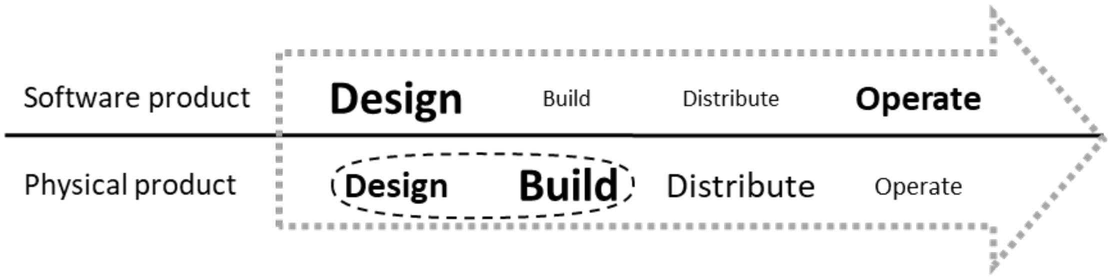


Figure 8: Comparison of development efforts for physical and software products throughout their life cycle (Socha and Walter, 2006)*). Larger and bold font implies more effort is needed during that stage of product development. Software products allow to focus effort on the design phase and require little effort during the build phase. In contrast, physical products need to be materialized in the build stage which represents a large effort. The production phase has a significant influence on the design of a physical product. Therefore, design and production requirements must both be considered in the design stage of the product.*

The phases Design Build and Distribute are not sequential in automotive design as expected for hardware products. In contrast, for example automotive design requires concurrent design efforts since interdependencies between the design activities require parallel activities. Automotive development projects are composed of three parallel, synchronized projects: the design of the product including the design of the verification system, the design of the product manufacturing facilities and the design of the necessary supply chain and product distribution.

Such product design conditions present severe deviations from Boehm's and Kruchten's Sweet Spot conditions (Boehm, 2002; Kruchten, 2013) for agile design. That is why early literature of agility in hardware design has resulted less efficient compared to software design (Schmidt *et al.*, 2019). Ovesen reports difficulty in design team composition. Increasing numbers of necessary domains and design phases require additional designers. Team separations are necessary to avoid oversized teams which in turn result in inter team dependencies (Ovesen, 2012). Ovesen also coined the term constraints of physicality to describe product and process characteristics specific to hardware products that reduce applicability of agile design (Ovesen, 2012). He states that the physicality of the product reduces the applicability of agile design based on his experiences with Scrum. More specifically he defines four constraints that cause this reduced applicability.

First, task breakdown into small and independent work packages is complicated in agile design of hardware products. Scrum requires these packages to fit into one iteration. But physical dependencies between components, necessary functional integration and multiple necessary domains increase the interdependency prevent or complicate the necessary task separation. Second, the separation of deliverables or prototypes is harder in hardware design. Potentially shippable physical products are constrained by long manufacturing times and necessary cooperation between involved domains. The integrated nature of for example mechatronic products and the necessary combination of design and manufacturing in hardware contradicts the idea of functionally separated subproducts which are shippable and provide customer value without the whole product. Third, there is less flexibility in hardware product design compared to software design. Mechatronic products are composed by a higher integration level than software products. Decisions cannot be taken without reducing the level of flexibility significantly. Additionally, physical dependencies increase the overall product interdependency system complexity and often results in incomplete understanding of system implications. Refactoring and repeated rebuilds are therefore much more complicated in mechatronic product design since changes might have unintended consequences for other product parts and often require new physical prototypes. Fourth, time and resource estimation are harder in agile design of hardware products. The higher interdependency level of the physical products decreases transparency of relevant dependencies and complicates necessary estimations. Concept development tasks are characterized by unknown unknowns which cannot be estimated. The first constraint, the difficulty to define small and independent work packages further enhances the estimation constraint.

Atzberger and Paetzold reviewed and confirmed Ovesen's constraints of physicality in 2019. Additionally, they present a set of updated constraints (Atzberger and Paetzold, 2019). Dependencies to external suppliers, verification and certification, complicated tool production prevent fast feedback cycles and negatively affect iteration speed. Legal restrictions require additional documentation. Regarding the number of involved teams, they underline the lack of sufficient coordination structures.

Ronkainen and Abrahamsson published constraints of agile software development in embedded systems (Ronkainen and Abrahamsson, 2003). They emphasis the impact of hardware on agile test-driven design strategies. System performance tests rely on combinations of hardware and software tests, but hardware test capacity and speed do not match software test characteristics which limits whole product testing performance. Furthermore, final software verification relies on functional hardware. Inter team coordination becomes more relevant at the cost of face-to-face communication since larger projects and additional stakeholder require distributed development across teams. Documentation practices of agile methods are often insufficient. Especially, change-prone requirements must be identified and managed. Up-front designs and architecture are necessary for hardware subproducts. Refactoring in turn becomes harder which limits experimenting opportunities of design teams. Transfer of prototypes into production models becomes harder and requires necessary maturity steps of the prototypes.

Greene further confirms the findings of Ronkainen and Abrahamsson in a report of shortcomings of agile methods in embedded firmware development at Intel. Team formation is complicated by the larger number of necessary experts across additional domains. Hardware tests do not match granularity and automation of software tests. Additionally, incomplete test coverage increases design dependency of final system tests. Kaisti et al. confirm the necessity for non-emergent product architecture, up-front design and plans in embedded design and describe the need for techniques to account for relevant specifications (Kaisti *et al.*, 2013). Documentation on a system level is necessary to ensure cooperation between designers and stakeholders. The number of involved design teams grows with increasing product integration which causes changes to affect more teams. To compensate the effort of change integration, embedded products require more rigid architecture and corresponding design practices in later development stages.

Conforto et al. differentiate non-software from software industries regarding agile design (Conforto *et al.*, 2014). Central differences between the product groups are the quantity of interaction between design teams, the number of designers involved, the complexity level of the product, the technological and cost barrier to prototype physically and the length of the development cycle. Their analysis emphasises problems with multidisciplinary teams in non-software industries. Multidisciplinary full-time teams are hindered by the number of specializations and designers. The coordination practices of employed agile methods are insufficient to provide inter team coordination and integrate customers into a multiteam system. Product complexity and prototype availability and integration level further complicate customer integration. Gustavsson's literature review of agile project management in non-software project identifies a lack of process visibility, missing manager buy-in and inadequate knowledge sharing as the top challenges. Insufficient resource allocation, redundant work, inadequate long-term planning, a lack of process visibility and individual tasks are secondary challenges (Gustavsson, 2016).

The presented literature sources have reported similar constraints of agile hardware design. The following summary lists the most referenced ones compared to software design. The number of disciplines and experts increases in hardware design projects. This requires documentation, coordination and communication to improve cooperation. Additional design steps such as manufacturing and distribution are central and connected to product design tasks. Iterative and incremental design is therefore complicated. Between subparts of the products are physical dependencies which increase dependencies between design teams and often require full scale prototypes for verification. Customer involvement is more complicated due to additional design teams and prototypes that do not reflect full product functionality. Hardware verification systems are less automated and less connected compared to software testing. Testing focuses on system tests. Technologies such as continuous integration or automated testing which are essential for agile testing strategies are not available at the same functionality yet.

Still, practical examples of successful agile design in hardware products such as cars or airplanes (Brown, 2013; Denning, 2012; Furuhjelm *et al.*, 2017) and annual industry reports (Komus, 2017; Schmidt *et al.*, 2019) prove that the development of physical products profits from agile design approaches. Nevertheless, agile methods and practices require adjustments to suit the new applications environment (Conforto et al., 2014; Schrof et al., 2018).

2.3.2 Agility in scaled design contexts

The aim of this subchapter is to describe and define scaled agile design and summarize and explain constraints of agile design caused by the scale of the process. This is relevant because automotive design is a large-scale process and therefore must regard the characteristics and limits of agility in scaled contexts. The subchapter is divided into two sections. In the first parts large-scale agile and current industry relevance are described. Definitions of large scale agile are compared and evaluated. The second part summarizes reported constraints of large scale agile.

There is a clear tendency to expand agile product design beyond individual team applications towards large scale applications. Edison et al. analyse in their literature review 191 primary study that focus on scaled agile design (Edison *et al.*, 2021). The sheer number of 191 studies reflects the relevance and interest of both the scientific community and practitioners at large scale agile. Despite this empiric proof of the scaled applications, agile methods were originally thought to be limited to small, co-located design teams (Conboy, 2009). Conboy states that even though the application has expanded to large-scale application the amount of literature regarding corresponding constraints clarifies that these applications might not be simple plug and play (Conboy, 2009). Maples argues that routines, practices and processes that worked well for small teams might be difficult to scale (Maples, 2009).

Before looking into characteristics of scaled agile design it is necessary to agree on a definition of the term "scaled agile". Dingsøyr et al. propose a taxonomy of scaled agile that accounts for the number of interdependent teams in design projects and the sensible coordination structure (Dingsøyr *et al.*, 2014). It is based on three categories. Small-scale agile includes one team and relies on agile coordination practices. Large-scale agile reflects projects that consist of at least two and up to nine teams. Coordination in large scale agile requires additional forums such Scrum of Scrums. Very large scale agile consists of ten or more teams and coordination is divided into several forums. Dikert et al. propose a similar division. They define large-scale to include 50 or more people or at least six teams. The involved persons do not have to be designers and may also include stakeholder. But there must be a need to collaborate between the involved actors (Dikert *et al.*, 2016). Rolland et al. further specifies the number of actors and teams with a network of interdependencies that requires collaboration between actors and teams (Rolland *et al.*, 2016). They also emphasise project size and overall project cost in their definition of large-scale agile.

Scaled or large scale design contexts also oppose agile sweet spot conditions. Increasing numbers of developers, stakeholders and teams connected in large projects affect the applicability of fundamental agile development principles. In the thesis at hand challenges to agile design approaches that relate to project size and complexity are summarized under the term constraints of scale and were noticed first in large software projects. Dependencies on product, process and system level are central causes. The following sources summarize constraints of scale that have been published since 2010. Constraints of scale reflect two categories. Systemic constraints of scaled development and constraints that are caused by the transformation process towards agile design. This study only summarizes systemic constraints.

Edison et al. present the most complete and up to date literature review of scaled agile design to the author's knowledge (Edison *et al.*, 2021). They report **inter team coordination constraints** in scaled agile design application. Synchronization and transparency across dynamic, adaptive teams are difficult. Communication overloads are caused by multiple agile layers and various ceremonies. The adjusted balance between inter and intra team activities increases external distractions for team collaboration. **Organizational structure constraints** are caused by the need to balance generalists and specialists teams, the fluidity of agile roles and flow levelling for limited resources. **Architectural constraints** include difficulties to see the big picture, lack of continuous integration and test automatization and a lack of software security awareness and measure. **Requirements engineering constraints** are driven by the difficulties in coordination rapidly changing requirements planning across teams, prioritisation and formulating small, valuable and measurable stories. **Customer collaboration constraints** are caused by difficulties to maintain a constant pace indefinitely. **Team related constraints** summarize a lack of ownership of user stories, over-commitment for faster delivers, a lack of team autonomy and fear of criticism. **Project management constraints** include conflicts between long-term planning and short-term sprint-based planning of agile, alignment difficulties to existing processes and stakeholder and insufficient meaningful metrics for performance and improvement.

Dikert et al. analyse scaling constraints and success factors in another literature review (Dikert *et al.*, 2016). They summarized systemic constraints into three categories: coordination in multiteam systems, requirements engineering and quality assurance and testing. Inter team coordination suffered from problematic interfacing between teams, distributed teams, individually divergent balance between team autonomy and collaboration with other teams and, insufficient technical consistency between teams and systems. **Requirements engineering** constraints included non-existent high-level requirements management, challenging requirement refinement, difficulties to create and estimate user stories and a gap between long-term and short-term planning**. Quality assurance and testing constraints** are driven by a lack of non-functional tests (e.g. performance, load and memory tests), a lack of test automatization across sub and ambiguous requirements due to insufficient requirements refinement. Uludag et al. confirm Dikert et al.'s findings in a secondary literature review focusing on systemic constraints of scale (Uludag *et al.*, 2018). They confirm Dikert's three systematic constraints categories and identify the additional constraints categories software architecture, team distribution, knowledge management and enterprise architecture. Sekitoleko et al. report technical dependencies between activities, artifacts and teams caused by scaled agile practices. The authors summarize difficulties in task prioritization, product quality, knowledge sharing, planning and product integration (Sekitoleko *et al.*, 2014).

Based on a large case study Dingsøyr et al. argue **that inter team coordination** causes severe problems in scaled agile design because teams which were originally intended autonomous are object to dynamic dependencies between tasks and hence teams (Dingsøyr, Moe, *et al.*, 2018). Šāblis et al. even state that the coordination of such interdependencies is one of the biggest challenges associated with large-scale software development today (Šāblis, Šmite, & Moe, 2020). The agile principle of team autonomy in small scale agile methods negatively impacts coordination and knowledge exchange between teams in multiteam systems. A **balance between team autonomy and inter team coordination** is therefore necessary. Simply scaling existing agile practices like a Scrum of has Scrum has not been proven successful (Paasivaara *et al.*, 2012). Hobbs and Petit confirm the existence of inter team coordination challenges in scaled projects and add two further interrelated constraints of scale: The organization of specialists outside of design teams and the integration of agile systems with other (existing) systems (Hobbs and Petit, 2017). Berger and Eklund see the need for continuous integration of product increments of different teams. This requires appropriate infrastructure and automated system tests. However, these structures cannot be generated by the development teams, but require central provision (Berger and Eklund, 2015).

In summary, the following constraints of scale are most relevant. Scaled agile projects increase project complexity and result in dependencies between tasks and hence teams. Content specific and phase related dependencies develop dynamically and affect project organization considerably. They result in constraints regarding inter team coordination, communication, and knowledge transfer. Central agile principles such as costumer integration, continuous integration and testing as well as emergent architecture are difficult to implement. Distributed teams, inter team dependencies, and the need for specialized teams contradict self-organized and cross-functional teams. To answer these challenges an adjusted balance between team autonomy, inter team coordination and knowledge management is necessary. Xu et al. call for formal centralized coordination strategies based on vertical communication and control for scaled agile development projects (Xu, 2009). Documentation needs to address stakeholder and costumer integration and product architecture needs to support overarching product functionality. Such adjustments must be flexible to project dynamics and avoid inefficient standards in scaled agile application contexts which has been implemented only partly in large scale agile methods (Alqudah and Razali, 2016).

2.4 Coordination theory in product design

"Prediction is very difficult, especially if it's about the future."
Niels Bohr

The aim of this subchapter is to present the state of the art of research on coordination theory relevant to product design with a focus on agile product design. This includes central concepts of coordination and explanations about dependencies from the relevant spectrum of research fields. The reason to include coordination into the State of the Art chapter is the theory's suitability to serve as a theoretical lens to agile design. Coordination theory allows to analyse and categorize agile design structures and to explain its benefits in reference to the application context and project settings. Agile design in its original context small-scale software design is used to generate a comprehensive description of agility in a coordination reference model. Based on this foundation disfunctions or constraints of agility in other contexts are analysed and adjustments or extensions to existing methods are recommended to expand applicability to new design contexts.

Table 5: Definitions of coordination across research fields chronologically ordered.

Definition of coordination	
"The integration or linking together of different parts of an organization to accomplish a collective set of tasks" (p. 322)	(Ven *et al.*, 1976)
"The act of managing interdependencies between activities performed to achieve a goal" (p. 361)	(Malone and Crowston, 1990)
"[…] different people working on a common project agree to a common definition of what they are building, share information, and mesh their activities" (p. 69)	(Kraut and Streeter, 1995)
"The extra work organizations and individuals must complete when individuals are working in concert to accomplish some goal, over and above what they would need to do to accomplish the goal individually"	(Krauss and Fussel, 1990)
"Coordination of understandings refers to the development of shared perceptions and meanings among members, including an appreciation of the ways in which members reliably see and interpret events differently" (p.1)	(McGrath *et al.*, 1999)
"Coordination can be defined as the collective accomplishment of individual goals through a cooperative process" (p. 401)	(Ballard and Seibold, 2003)
"[…] the integration of organizational work under conditions of task interdependence and uncertainty" "A temporally unfolding and contextualized process of input regulation and interaction articulation to realize a collective performance" (p. 1157)	(Faraj and Xiao, 2006)

Table 5 gives an overview of the spectrum of coordination definitions in the last 45 years across research fields. Van de Ven et al. stress the connection of different organization units (e.g. individuals or teams) to accomplish a collective set of tasks, which implies dependencies between such tasks (Ven *et al.*, 1976). Malone and Crowston define coordination in their coordination theory and emphasise the management of interdependencies between activities without addressing people (Malone and Crowston, 1990). Kraut and Streeter expand Van de Ven et al.'s original definition and introduce practices (e.g. agree on a common definition, share information, mesh activities) how this coordination is accomplished (Kraut and Streeter, 1995). Krauss and Fussel on the other hand define coordination broadly as the necessary extra work of cooperating individuals without specifying details (Krauss and Fussel, 1990). McGrath et al. point to how coordination is enabled. They describe the development shared perceptions and meanings among team members as a central enabler for coordination (McGrath *et al.*, 1999). Ballard and Seibold summarize coordination as an collective accomplishment of individual goals through a cooperative process (Ballard and Seibold, 2003). Faraj and Xiao further analyse the concept and the dynamics of coordination and define it as temporally unfolding and changing according to context inputs (Faraj and Xiao, 2006).

The definitions from above draw from different theoretical fields. Organizational theory, coordination theory, psychological theory and cognition theory provide different perspectives on coordination. Even though these definitions focus on different aspects of coordination three common aspects are apparent (Okhuysen and

Bechky, 2009). First, actors need to work together. Second, the work is interdependent. Third, a goal is achieved. Therefore, the thesis at hand is based on Faraj and Xiao's conceptualizations of coordination as

"*…the integration of organizational work under conditions of task interdependence and uncertainty with an emphasis on its dynamic emergence in design projects*" (Faraj and Xiao, 2006).

In this subchapter aspects of coordination in product design are presented relating to several theoretical fields that are important to agile design characteristics. First, the perspective of organization research on coordination is introduced. Second, aspects of coordination in team research are summarized. Third, coordination in multiteam system and inter team coordination is described. Fourth, coordination mechanisms as practical implementation of coordination or coordination activities are summarized and compared across different research fields. Fifth, the result of coordination efforts, the dynamic state of coordination is described. Sixth, the concept of the coordination strategy which integrates coordination determinants, coordination mechanisms and coordination as a state is explained.

2.4.1 Coordination in organization research

The formal study of coordination started with the emergence of large-scale manufacturing in the beginning of the 20th century. This initial coordination research field is dominated by two branches with different approaches. The first group researched the **design of work** and is mostly associated with Frederic W. Taylor and his role in scientific management. Work was observed, analysed and decomposed into its most basic elements to allow for specialization and the reduction of waste. Methods supported standardization and interchangeability of designs, tools, and materials. These efforts intended a most efficient use of workers in production (Taylor, 1916). Later scholars critically reviewed the downside of standardization regarding necessary integration activities and additional communication demand (Scott and Davis, 2015). The second group addressed the **design of organizations**. Henry Fayol a former student of Taylor is best known for it. Design of organizations aspired coordination of work through the adjustment of management systems. Rationalization was driven by principles such as hierarchical systems, centralization, and the subordination of individual interests. The unity of command as the central element in administration needed to be respected. Fayol positioned his approach as top-down in contrast to Taylor's bottom-up work design (Fayol, 1949).

Both approaches defined coordination to be a controlled and efficient state of a work system regarding either relationships between individuals or task and component decomposition. They assumed that product development and respective coordination requirements in different contexts and companies can be formalized into representative models precisely with enough specificity. Work was designed according to these formalized models to enable individuals to fulfil their part as collectives within these systems. Critics have proposed two major shortcomings of these coordination approaches. First, interdependencies between pieces of work are often uncertain or hard to define. This contradicts the assumption that interdependent systems can be described in sufficient detail. Second, processes and structures need to be adapted continuously to changing conditions and cannot be planned as formal elements by organizations. Formalized designs cannot account for all eventualities and therefore require continuous reshaping to emerging coordination challenges. These early approaches to coordination focused mostly on product manufacturing and not product design. They underestimated the influence of uncertainty and generated deterministic and therefore more designable models. In this early organizational Design Research stream unpredictable coordination efforts have been simplified with terms such as "*mutual-adjustment*" (Thompson *et al.*, 2017) or "*ad-hoc coordination*" (Donaldson, 2001). Later theories switch perspective and assume that these coordination efforts represent a significant amount of the overall coordination demand. More complex product development conditions (driven by new technologies and cooperation models) changed the nature of work and the limitations of these "classic" coordination theories became more evident (Okhuysen and Bechky, 2009).

March and Simon therefore proposed a division of **coordination by plan** and **coordination by feedback** in 1958 to account for the human factor in work systems (March and Simon, 1958). Repetitive and predictable tasks are coordinated by scheduling and planning. Uncertainty or dynamic dependencies require repeated exchange and communication. Emergent coordination by feedback is more appropriate in these situations. Based on March and Simon Thompson describes the suitability of coordination methods according to three work **dependency characteristics** (Thompson, 1967). **Pooled dependencies** characterize units that independently complete tasks without explicit interaction. Standardization with little communication and decision effort is a sufficient coordination method. **Sequential dependencies** arise between units that need outputs from one unit as input for another unit. Coordination by planning with mediocre communication effort is suitable for such dependencies. **Reciprocal dependencies** characterize units that rely on simultaneous bidirectional flow of input and output between each other. They represent the strongest form of dependencies and require coordination by feedback. Van de Ven et al. further elaborated March and Simon's division of coordination types into coordination by programming or by feedback answering to different coordination needs. They proposed three modes of coordination with an emphasis on coordination by feedback (Ven *et al.*, 1976). The **impersonal mode** includes most programmable coordination mechanisms such as standardization, plans, rules and hierarchies. Opposed to impersonal coordination is mutual adjustment coordination which relies on direct communication between relevant parties. To better describe mutual adjustment a group mode and an individual mode were differentiated. The **group mode** includes coordination by scheduled and unscheduled meetings in groups. Scheduled meetings are ideally used for routine coordination efforts such as group meetings, while unscheduled meetings provide coordination answering to urgent needs of groups. The **individual mode** describes coordination by feedback between individuals and includes horizontal channels on the same hierarchy level and vertical channels across hierarchies. It is based on informal communication. Individual role occupants serve as the mechanism for making mutual task adjustments. Van de Ven et al. also expand on Thompson's approach of fitting coordination methods to task dependency classes (Ven *et al.*, 1976). They add the categories task **uncertainty** and **size of work unit** to **task dependency** as coordination type determinants. Their empirical findings show that higher task uncertainty increases substitution of impersonal coordination modes using mutual adjustment in form of the group mode coordination and by the individual mode coordination through horizontal channels. Large unit sizes on the other hand result in more impersonal mode of coordination such as plans. Task dependencies increases group mode coordination while individual mode coordination remain invariant and impersonal mode coordination diminishes a little. The collective coordination demand increases with unit size.

Malone and Crowston define coordination as "*management of dependencies among task activities*". They published a **coordination theory** based on actors, interdependent tasks, resources and goals (Malone and Crowston, 1990). Three types of dependencies that result from resources being required by or result from different activities are presented. This typology of dependencies further expands Thompson's categorization of task dependencies (pooled, sequential and reciprocal) with the constraints actors and resources. The **fit dependency** describes a situation in which multiple activities collectively produce components that need to be integrated into a complete product. The **sharing dependency** prevails if multiple activities require the same resource e.g. functional prototypes in physical product testing. The **flow dependency** represents a sequential order of activities. Output from one activity is input for another activity. Usability, accessibility and prerequisite have to be adjusted to coordinate a flow dependency. The theory of Malone and Crowston includes coordination mechanism to manage these dependencies and show their substitutability in different applications (e.g. sequencing, tracking, standardization for flow dependencies or goal selection and decomposition for task-subtask dependencies). Critics of their coordination construct claim that little explanatory theory accompanies the typology of dependencies and the influence of context and time are not represented (Crowston *et al.*, 2006).

2.4.2 Coordination in team and multiteam systems

Ramesh et al. claim that coordination in knowledge-intensive systems requires a new level of coordination adaptivity due to the immense impact of fast innovation on work interdependencies (Ramesh *et al.*, 2002). Therefore, coordination has also been a central topic in the research stream of sociology of work with a focus on intra team and inter team cooperation in information systems. Coordination in team cognition research focuses on coordination behaviour of teams. Aspects such as shared experience, personal knowledge of each other and trust in teams are central coordination enablers in this research stream.

Espinosa, Lerch and Kraut divide explicit and implicit team coordination (Espinosa *et al.*, 2004). Explicit team coordination summarizes activities and coordination mechanisms that are purposely applied to coordinate, while implicit team coordination arises as a consequence of other activities that are used without the direct intention to coordinate. **Explicit coordination** is based on task programming mechanisms (e.g. division of labour, tools, plans and specifications) and communication between parties and individuals. The task programming classification has been described similarly by Van de Ven et al. as impersonal mechanisms (Ven *et al.*, 1976) and by Faraj and Sproull as administrative coordination (Faraj and Sproull, 2000). Coordination through communication has been described earlier under the terms mutual adjustment (Thompson, 1967) as well as personal and group mode coordination (Ven *et al.*, 1976). Formal or informal communication may be between individuals or in groups. In explicit team coordination Espinosa recommends impersonal coordination for routine and predictable tasks and coordination by feedback based on communication for dynamic and unpredictable coordination requirements. Espinosa's conception of explicit coordination overlaps with the early descriptions of coordination and the respective mechanisms in organization theory. **Implicit coordination** relies on shared task knowledge, team cognition and shared mental models (Cannon-Bowers *et al.*, 1993) that develop during close team cooperation. This expertise covers both the task and the team and help to coordinate implicitly. Earlier publications reported "*synchronization of member actions based on unspoken assumptions about what others in the group are likely to do*" (Wittenbaum and Stasser, 1996). Espinosa defines implicit coordination mechanisms as "*available to team members from shared cognition, which enable them to explain and anticipate task states and member actions, thus helping them manage task dependencies*" (Espinosa *et al.*, 2004). The setup of coordination mechanisms and team coordination has a strong influence on how team cognition and hence implicit coordination develops. Throughout cooperation length and intensity team cognition improves and implicit coordination may substitute initially explicit coordination mechanism. Espinosa et al. present a framework that dynamically combines implicit and explicit coordination according to team (e.g. size, experience, and continuity), task and context characteristics (e.g. technology, organization, synchronicity and geographic dispersion). They emphasize that neither implicit nor explicit coordination mechanisms are to be preferred but must fit the specific project requirements.

In continuing work Espinosa, Armour and Boh describe a **taxonomy of coordination types** that includes mechanistic, organic and cognitive coordination (Espinosa *et al.*, 2010). In this taxonomy **mechanistic coordination** refers to plans, processes, automation or rules similar to impersonal coordination from organization theory (Ven *et al.*, 1976). Mechanistic coordination manages dependencies with little communication and is most useful for activities that are routine or well-predictable. **Organic coordination** refers to coordination by feedback or by mutual adjustment (Ven *et al.*, 1976) and mainly involves coordination by communication and interaction. It is most relevant with uncertain and non-routine task. Since it requires more effort Espinosa recommends it if mechanistic coordination is unsuitable, e.g. in unpredictable and dynamic situations. **Cognitive coordination** is an implicit coordination mode which is based on shared cognition (Rico *et al.*, 2008) in teams. It is achieved implicitly and based on tacit team knowledge regarding task and team members. It includes task awareness, presence awareness, transactive memory (knowledge who knows what) (Wegner, 1995) and expertise coordination (Faraj and Sproull, 2000). Shared mental models in teams (Cannon-Bowers *et al.*, 1993) are essential since they supports shared goals and enable common understanding (Kang *et al.*, 2006). Unlike coordination mechanisms in explicit coordination, cognitive coordination cannot be implemented like organic or mechanistic coordination mechanisms, since it requires specific knowledge distribution (e.g. accessibility of cognitive coordination mechanisms is limited by the existing level of shared cognition in teams). It relies on mutual knowledge which is knowledge shared by collaborators they know they mutually share (Krauss and Fussel, 1990)**.** Common grounding is a related concept and requires collaborating parties to have shared meaning in the terms they use to communicate (Cramton, 2001). Li and Maedche showed that with increasing shared cognition cognitive coordination becomes stronger (Li and Maedche, 2012).

Multiteam systems are defined as a setting of multiple teams working jointly and interdependently towards collective goals (Mathieu *et al.*, 2001). In multiteam projects, work of separated teams is often interlinked. Even though team division is often chosen according to product modules interdependencies between teams arise through technical interfaces between these modules (Kazanjian *et al.*, 2000). Such inter team dependencies generate the need for additional inter team coordination to exchange information, share knowledge and solve conflicts (Galbrath, 1973). The value of inter team coordination has been proven regarding performance predictors (Marks *et al.*, 2005), product quality, development time and project commitment (Hoegl *et al.*, 2004). Studies in organizational psychology on multiteam systems (Marks *et al.*, 2001) showed that inter team processes are even more important than intra team processes for the performance of multiteam systems (Marks *et al.*, 2005). Inter team coordination varies from intra team coordination, since coordination requirements and available coordination mechanisms are different. Still, it has been shown that intra team coordination has a large influence on inter team coordination (Firth *et al.*, 2015).

Based on Van de Ven's coordination categories (group and individual mode of personal coordination and impersonal mode of coordination) Dietrich et al. differentiate three patterns in inter team coordination (Dietrich *et al.*, 2013). **Centralized coordination** relies on formal group meetings (e.g. status review meetings) for information and knowledge exchange. Informal group meetings (e.g. colocation of project managers), workshops and integration meeting are complementary channels. Additionally, well-defined roles and responsibilities are applied, and powerful project managers function as connectors between different teams. **Decentralized coordination** on the other hand is largely based on individual contacts between team members. It is not pre-determined in strict roles and responsibilities and therefore able to adapt to changing situations. Functionality reports, testing documents, common databases, resource plans, reporting practices and overall project plants are used as coordination mechanisms. Interaction between teams is frequent and group meetings are complemented by liaison individuals (e.g. project manager). **Balanced coordination** features a balance between centralized and decentralized coordination patterns and relies on group, individual and impersonal coordination mechanisms. Formal reporting practices and the use of documents and databases are relevant in sharing information and coordination work with other teams. Individual mode of coordination is important but is mostly applied by strict roles along existing hierarchies in vertical channels outside development teams. The authors also analysed efficiency of the coordination patterns. They recommend the decentralized coordination scheme in cases of high inter team interdependencies. But decentralized coordination may suffer from problems if task specifications are vague. The selection between centralized and decentralized coordination should be chosen according to task analysability (Dietrich *et al.*, 2013).

Salas et al. complement a model of five mutually interlinked success factors in teamwork (team leadership, mutual performance monitoring, back-up behaviour, adaptability and team orientation) based on coordination mechanisms that apply to single and multi-team projects (Salas *et al.*, 2005). In Salas' model coordination mechanisms are shared mental models, closed-loop communication and mutual trust. **Shared mental models** (Cannon-Bowers and Salas, 2001) enable team members to coordination by anticipating and predicting each other's needs through common understanding of the environment and expectations of performance. Salas et al. divide team-related and task-related mental models. The importance of this coordination mechanism increases in teams that are object to stressful conditions, since available time for direct communication decreases (Cannon-Bowers *et al.*, 1993). **Mutual trust** is a shared perception that individuals in the team will perform their tasks according to team agreements. It also implies that team members will recognize and protects the interests and rights of each other (Simsarian Webber, 2002). This trust is necessary since team members will work on independent tasks and must be able to rely on each other to meet deadlines and contribute as agreed without contra productive or selfish intentions. Trust is relevant in both intra and inter team cooperation. Distrust might by a hindering factor in multi-team systems since different political agendas complicate cooperation. **Closed-loop communication** is the third relevant coordination mechanism and enables efficient information exchange within teams irrespective of medium. Understanding of the meaning of the message is supported by an additional feedback loop between sender and receiver (McIntyre and Salas, 1995). It supports decision making in complex environments, avoids misinterpretations and complements information distribution and selection. Especially in multi team environments this becomes increasingly important since support of shared mental models decreases between different teams.

2.4.3 Coordination mechanisms

Mintzberg describes coordination mechanisms as organizational arrangements that allow individuals to realize collective performance (Mintzberg, 1989). They are the practical implementation of coordination and therefore one of the most basic elements of structure in organizations. According to the chosen coordination approach, coordination mechanisms might be formal or informal as well as emergent or structural elements. The concept of coordination mechanisms is used across most coordination theory streams. Okhuysen and Bechky present the following categories and explanations of coordination mechanisms in their coordination review (Okhuysen and Bechky, 2009). **Roles** represent expectations associated with social positions, and therefore support predictable behaviour (Banks and Hughes, 1959). Defining relationships between roles allows parties to understand and predict who does what (Bechky, 2006). In traditional organization theory roles are used for monitoring and updating in formal hierarchies. But roles also provide a shared understanding of task responsibilities and therefore enable substitution between parties. Inter-group boundary spanning roles expand a common perspective across separated parties. **Plans and rules** are conceptualized as purposive elements of formal organizations (March and Simon, 1958). Plans support a prospective understanding of task completion. Rules complement plans since they establish relationships between parties and allow fast choices in routine situations. Plans and rules define responsibility for tasks and therefore support resource allocation. They can be developed on team or organization level. The development of plans and schedules highlights conflicts and difficulties and increases understanding. Feldman defined **routines** as repeated patterns of behaviour that are bound by rules and customs (Feldman, 2000). They provide a template for task completion, bring parties together and create a common understanding of tasks. In literature, routines have been interpreted as stores of knowledge (Loasby *et al.*, 1983), as stable mechanistic properties of traditional organizations and as complex constructs in which social meaning and social interaction are embedded (Feldman, 2000). Routines support task stability and completion, they facilitate hand-off work, they bring groups together and they create a common perspective. The physical **proximity** of parties significantly influences the amount of interaction and communication between them (Allen, 1977). Visibility and familiarity influences communication and liking. Proximity allows formal and informal monitoring, updating and familiarity. **Familiarity** is the understanding that individuals have of each other and it results from proximity (Okhuysen, 2001). Familiarity leads to stronger relationships that improve coordination. It supports anticipating and responding, the creation of transactive memory systems as a storage of knowledge and trust development.

Objects and representation are a further category of coordination mechanisms if interpreted as boundary objects. The concept of **boundary objects** originally stems from communication research. It is based on the ability of objects to convey technical and social information and mobilize action across social worlds (Star and Griesemer, 1989). Exchange between parties does not require comprehensive communication since knowledge and social dynamics are stored in objects (Winner, 1980; Latour, 1988, 1996). The boundary objects provide interfaces between different social or technical backgrounds and enable exchange without mutual translations (Burris and Henderson, 2001). According to the original concept these objects (the understanding of them) need to be sufficiently plastic (generalizable) to adapt to different social backgrounds but still adequately robust to support a consistent message. According to varying application situations this balance needs to be adjusted dynamically. Boundary objects enable efficient communication between parties that focus on different aspects and prevents miscommunication (e.g. prototypes may connect customers, designers and management who all have very specific interests). This principle even works without physical manifestation (e.g. user stories as a subgroup of boundary objects do not require physical manifestations to efficiently connect customers and designer). **Boundary spanners** are based on a similar concept as boundary objects. They represent roles that efficiently connect different interest groups if direct exchange is impractical or not applicable (Levina and Vaast, 2005). In a nutshell, these boundary concepts allow efficient exchange between parties with different backgrounds and provide efficient information exchange and hence coordination between them.

2.4.4 Coordination determinants

The summarized coordination theory descriptions demonstrate that coordination structures are object to very heterogeneous project needs. To differentiate these coordination requirements individual project characteristics have been identified across the presented research streams that have significant influence on the coordination setup. These **coordination determinants** allow a direct project analysis according to central projects characteristics regarding suitability of different coordination types and mechanisms.

Van de Ven et al. list unit size, task uncertainty and task dependency as significant project characteristics to determine a coordination strategy (Ven *et al.*, 1976). **Unit size** summarizes factors that influence the total number of relevant stakeholders within in a project. It depends on the number of participating teams, the number of designers, the interchangeability of designers, the number of required specializations and substitutability between them and also includes organizational dependencies. **Task uncertainty** integrates the factors task predictability and task changeability. Ramasesh et al. differentiate task uncertainty into known unknowns (recognized uncertainties) and unknown unknowns (unrecognized uncertainties that are unpredictable and of which projects are unaware of) (Ramasesh and Browning, 2014). Especially unknown unknowns require emergent coordination. With increasing project complexity, the additional category unknown knowns becomes relevant since information gets divided and separated into organizational silos. **Task dependency** depicts the degree to which tasks dependent on each other. Thompson defined pooled, sequences and reciprocal task dependency categories (Thompson, 1967). Pooled tasks feature little direct dependencies and can be executed independently. Sequenced tasks are subject to an order regarding task execution. Reciprocal task dependencies include dependencies in both directions and may require repeated exchange between different tasks. Malone and Crowston included a similar task dependency classification into their coordination theory (Malone and Crowston, 1990). They defined fit, flow and sharing dependencies as coordination type differentiator.

Espinosa et al. define task, team and context as most relevant factors that cause dependencies and therefore influence coordination setups in teamwork (Espinosa *et al.*, 2004). The nature of the **task** (e.g. routineness) predefines a large share of the dependencies like Van den Ven's categorization. **Team** variables such as continuity, composition, expertise and size have a strong influence on cognition development (Cannon-Bowers *et al.*, 1993; Cannon-Bowers and Salas, 2001). Regarding the emergence of dependencies Espinosa et al. regard the project **context** as relevant. They differentiate four context sub-factors: technology, organization, synchronicity and dispersion. Especially communication between team members relies on **information technology**. Dependencies, information flow and workflow among collaborators increasingly depend on available information technology. Large scale software development relies on systems that supports continuous integration and simultaneous access for multiple developers to reduce dependencies. The **organization** (e.g. culture, structure, standard procedures) also has a strong influence on the number of dependencies between teams and therefore team interaction. **Synchronicity and geographical distribution** affect both team and task factors and need to be addressed in coordination set-up. Asynchronous and dispersed teams have fewer opportunities to interact and communicate with less rich media. They negatively affect shared cognition and hence rely on more mechanistic coordination mechanisms. Li and Maedche complement different **socio-cultural environments** as an additional factor to generate and influence dependencies (Li and Maedche, 2012). Different value systems and normative practices create socio-cultural boundaries between parties (Holmstrom *et al.*, 2006). Li and Maedche also described that **changing customer requirements** become a strong coordination determinant in dynamic application contexts.

Diane E. Strode differentiates dependencies between actions and presents a dependency taxonomy relevant for agile product design. (Strode, 2016). She defines knowledge, process and resource as three overarching dependency categories that have the potential to influence project progress. **Knowledge dependencies** result if progress relevant information is not at hand. The category includes requirements dependencies (e.g. missing domain knowledge), expertise dependencies (e.g. information only known by individuals), historical dependencies (e.g. knowledge about past decisions) and task allocation dependencies (e.g. knowledge who is doing what and when). **Process dependencies** are caused by the necessary order of tasks and the relevant activities. They are refined by activity dependencies (e.g. one activity requires the completion of another activity) and business process dependencies (e.g. an existing business process causes tasks to be carried out in a predetermined order). **Resource dependencies** occur if an object is required for a progress to occur. Strode defined entity dependencies (e.g. unavailable resource or person) and technical dependencies (e.g. technical dependencies are caused by the presence or absence of software components) as subcategories of resource dependencies.

These coordination determinants characterize coordination requirements and indicate suitable coordination mechanisms. The presented literature clarifies that there is an overlap between scholars regarding task dependencies and project driven factors such as team specific characteristics and context factors.

2.4.5 Coordination outcome

The presented literature shows that coordination is interpreted and approached with different theoretical backgrounds and means which complicates an overarching understanding. Okhuysen et al. report three obstacles regarding the body of literature (Okhuysen and Bechky, 2009). First, interdisciplinary research streams differ strongly regarding the object that is being coordinated. Malone and Crowston's coordination theory focuses on dependencies between tasks that require coordination (Malone and Crowston, 1990). Faraj and Sproull research coordinating knowledge in organizations (Faraj and Sproull, 2000). The integration of these research streams into a shared understanding of coordination is not trivial since their focus of coordination and the corresponding action differs considerably. Second, coordination research is embedded in a broad spectrum of contexts. This complicates comparisons between research streams. Additionally, different terms are used for similar functionalities and mechanisms in different contexts. Confusion regarding contradicting results might be caused by unclear terminology. And third, most of the literature does not provide sufficient explanations why and how coordination mechanisms function. To avoid such misunderstandings between literature streams Okhuysen et al. identified accountability, common understanding and predictability as the three most relevant

integration conditions for coordination (Okhuysen and Bechky, 2009). They represent the means of separated parties to collectively accomplish interdependent tasks. Each condition answers specific demands regarding the integration of specialized work. To establish and maintain these conditions different coordination mechanisms can be implemented or combined.

Accountability establishes who is responsible for particular aspects of the task. It supports the cooperation between interdependent parties since the responsibility for individual parts are clearly allocated between partners. Transparent responsibilities make parties accountable for their contribution and allows them to make other accountable for theirs. Accountability is achieved by various means. Traditionally accountability was built into formal structures such as hierarchical authority by e.g. reporting metrics (Gittell, 2000). But lateral interaction in meetings or public status reports support accountability as well. Both formal and informal or emergent action can lead to accountability. Plans and rules connect tasks and people and transparently show responsibilities. Boundary objects provide scaffolding for the responsible parties. Roles, routines and visibility support monitoring, updating and hand-offs between cooperation partners. Accountability also requires trust between the relevant parties to be able to count on consistent and reliable performance of others which can also be provided by proximity (McEvily *et al.*, 2003).

Common understanding enables a shared perspective on the overall design objective and how different parties fit into it. It assists parties to integrate their effort into a collective conception of the work. It supports a common ground that allows independent partners to integrate activities. In literature three perspectives are considered. First, common understanding of the task regarding necessary action and strategy to perform the task (Cannon-Bowers and Salas, 2001). Second, knowledge of interaction partners in interdependent situation (Reagans *et al.*, 2005). And third, knowledge of broader organization or project goals that characterize the design context (Pinto *et al.*, 1993). Common understanding is provided by formal and emergent coordination mechanisms. It is generated during the development, distribution and executions of plans in both top-down and bottom-up approaches on a system level. Objects such as prototypes allow boundary crossing and provide common understanding on a more task-specific level. Roles facilitate substitution between individuals and groups. Additionally, boundary spanner roles create a common perspective. Proximity of parties creates familiarity and knowledge of expertise distribution. In summary, common understanding provides interdependent parties with a shared conception of interlinked activities.

Predictability supports interdependent partners to anticipate subsequent tasks. It is based on the knowledge or experience of how tasks are divided into smaller subtasks and their particular sequence. Predictability enables parties to count on the successful execution of the work of partners and structure their work accordingly. It allows parties to fit their contribution into the whole enhancing integrating activities. High levels of predictability go hand in hand with trust into cooperation parties. Routines provide predictability. Predictability is provided through familiarity that increases understanding of partners and their tasks. Plans actively define responsibility for tasks and define resource allocation. Boundary objects provide scaffolding for cooperation partner. Routines passively establish essential tasks and enhance task completion and stability. Familiarity increases knowledge of coordination partner and therefore improves anticipating their behaviour and responding accordingly.

Strode et al. described with **coordination effectiveness** a similar concept to differentiate the state of coordination but with a focus on agile design (Strode *et al.*, 2011). They subdivide coordination effectiveness into an implicit and an explicit part. Explicit coordination effectiveness focuses on the persons and objects in a project. The right person or object needs to be in the right place at the right time. Explicit coordination effectiveness therefore draws from Malone and Crowston's coordination theory (Malone and Crowston, 1990). Implicit coordination effectiveness comprises the knowledge held by project parties. It includes knowledge of overall project goals ('know why'), project status ('what is going on and when'), what tasks need to be done ('what to do and when'), what tasks others are doing ('who is doing what') and who knows what. The concept of implicit coordination effectiveness draws from teamwork literature and cognitive coordination (Espinosa *et al.*, 2010). Strode's coordination effectiveness concept is similar to Okhuysen and Bechky's integration conditions. Especially the implicit part directly corresponds to predictability, common understanding and accountability.

The presented integration conditions represent central manifestations of coordination as a state in interdependent systems. They rely on the successful application and combination of coordination mechanisms according to the chosen coordination approach and the specific coordination requirements. If external or internal disturbances throw this system out of balance the applied coordination elements require readjustment. The connection between coordination determinants, coordination mechanisms and integration conditions is provided by the coordination strategy which is further explained in the following paragraph.

2.4.6 Coordination strategy

Coordination in design projects relies on the selection of suitable coordination mechanisms to realize integration conditions according to project specific coordination determinants. This connection is evident in all presented coordination theories. The concept of a coordination strategy combines these inputs into comprehensive approaches. Li and Maedche define the coordination strategy as 'a set of prioritized mechanisms for a given circumstance' (Li and Maedche, 2012). Strode et al. define it as 'a group of coordination mechanisms that manage dependencies in a situation (Strode *et al.*, 2012). Both definitions describe a combination of coordination mechanisms into an overarching strategy to realize coordination. The coordination mechanisms are chosen according to coordination determinants to realize an effective coordination implementation specific to the coordination requirements (see Figure 9). These and given input factors guide the selection of the most relevant coordination mode for the specific situation. For example, small teams can rely on cognitive coordination approaches, while large projects require more mechanistic coordination mechanisms to ensure efficient information distribution and exchange. The selection of coordination mechanisms is therefore predetermined by the selection of the general coordination approach. The coordination strategy realizes desired integration conditions and the resulting state of coordination. Their selection is based on project specific requirements.

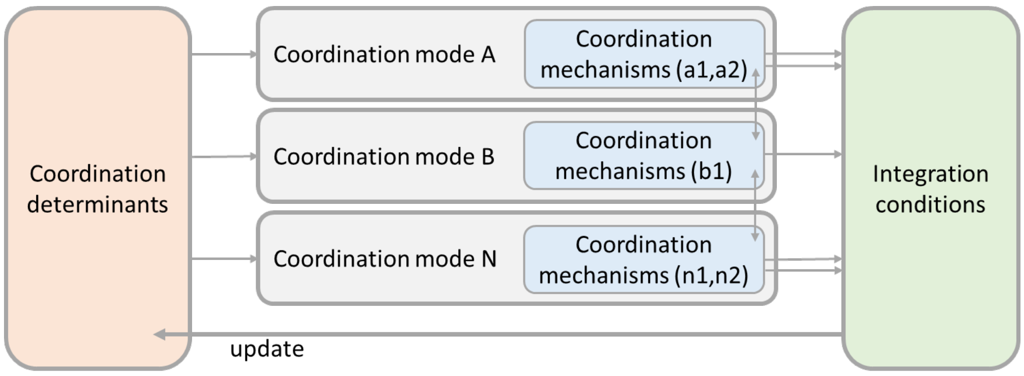

Coordination determinants
Coordination mode A
Coordination mechanisms (a1,a2)
Coordination mode B
Coordination mechanisms (b1)
Coordination mode N
Coordination mechanisms (n1,n2)
Integration conditions
update

Figure 9: The coordination strategy connects coordination determinants with suitable coordination modes and mechanisms to realize specified integration conditions. The integration conditions reflect the state of coordination and the coordination determinants and their implementation the process of coordination. The concept allows to adjust coordination to change. Project dynamics that impact integration conditions lead to changes in coordination determinants and therefore cause an adjusted coordination implementation until the pre-specified integration conditions are re-established.

To be able to handle project dynamics a coordination strategy requires continuous readjustment to remain effective (see Figure 9). After an initial state of coordination has been established it is necessary to maintain it (Li and Maedche, 2012). Repeated re-evaluation of the state and relevance of chosen integration conditions, coordination determinants and mechanisms are necessary. Since coordination mechanisms have a strong mutual influence on each other the coordination strategy must account for the changing efficiency and availability of coordination mechanisms in design projects. An initially mechanistic coordination strategy may be able to apply more implicit coordination with growing team cognition. Otherwise, a growing project team might realize that organic coordination mechanisms lose efficiency with larger number of participating parties.

In summary, the coordination strategy concept connects the process and the state of coordination. While the state coordination is presented in integration conditions (Okhuysen and Bechky, 2009) and coordination effectiveness (Strode *et al.*, 2011) the necessary action to establish and maintain this state are presented in the coordination strategy. The coordination strategy is a consistent coordination setup that defines a combination of coordination mechanisms according to coordination determinants to realize chosen integration conditions. It needs to account for flexible reactions to dynamics in project development and changing suitability of coordination mechanisms. Whenever integration conditions are unfulfilled a re-examination of the coordination elements and hence the coordination strategy is necessary.

3 Research approach

"Write down the problem. Think very hard. Write down the solution.
The Feynman Algorithm"
Murray Gell-Mann

The aim of this research is to comprehend and enable agility in the automotive domain. The research approach describes how this aim was approached methodically. It summarizes the underlying structure of the research project and explains the characteristics of Design Research. It introduces the research methodology Action Research and accounts for methodology inherent and application context specific limitations.

The chapter is subdivided into three interlinked subchapters. Subchapter 3.1 specifies the research based on the research aim. It introduces the research questions and their connection to the research aim. It explains why coordination theory is selected as the theoretical lens. The relevant research fields are described, and corresponding publications are presented. Subchapter 3.1 concludes with the comprehensive research overview which connects the research questions, the research fields and the employed research methods. Subchapter 3.2 introduces Design Research as an independent research field and delaminates it from adjacent and traditional fields. Its research paradigm is described based on its ontological and epistemological position. Additionally, the influence of the research field on the spectrum of suitable research methods is addressed. Concluding, the balance between research relevance and research rigor are addressed by the selection of a comprehensive research model for Design Research projects. Subchapter 3.3 presents the selected research methodology and its practical implementation. Based on the established research principles Action Research is chosen as central research methodology. The origins and the concept of Action Research are briefly summarized. The research project implementation is described by the adapted Action Research conduct, the selection of the design projects and the employed data collection methods. Finally, complementary research methods to improve the Action Research rigor are supplemented.

3.1 Research design

The structure of the research project is defined by the research questions, the employed theoretical lens, the research fields and the research overview. The research questions were formulated to realize the research aim in relation to the established research gap. To address the research questions a suitable design theory was selected as a theoretical lens to analyse the findings. The research fields define bodies of knowledge that are relevant to the research project, and which have been contributed to. The research overview emphasises the systematic approach of the complete research project. It clarifies the connections between research questions, research fields, and research methodology. It allows to reproduce how research fields and research questions are connected and how they were approached methodologically.

The structure of the research project was designed to reflect the practical nature of both the research aim and the collaborative research conduct within the research and development department of the BMW Group. The responsibility of the development department is innovation in automotive product design in the form of new automotive products. The chosen application context offers the opportunity to research agile product design in a range of pure software, to software hardware hybrid and complete hardware products. The research of this heterogeneous application context was conducted through agile pilot projects which introduced agile methods to product design project. They allowed to test and further develop adaptions to existing agile methods to fit the requirements of automotive design. The selection of pilot projects covered a broad range of the automotive design requirements to realize a representative data set. This spectrum improved understanding of the research phenomena and avoided an imprecise problem definition. The independent pilot projects also allowed to incrementally construct a design artifact that answers to a class of problems instead of an individual instance.

3.1.1 Research questions

"*What I love about science is that as you learn, you don't really get answers.*
You just get better questions."
John Green

The research in this study is structured according to three central research questions as presented in subchapter 1.3. The research questions are based on the real-world problem agility in automotive design. They are designed to lead to a scientific understanding of a class of problems and to an enrichment of the respective theory. Their thematic spectrum reflects Mathiassen's guidelines to distinguish research from practical problems (Mathiassen, 2017).

Research question one: ***How to explain agility and its benefits theoretically?***

The first question relates to the empiric and pragmatic nature of agile product design. Agile methods were developed decentralized according to best practices for specific problem fields by practitioners. Even though a broad body of literature from practice and research confirms the benefits of the approach little comprehensive theory-based explanations exist what causes these benefits (Dingsøyr *et al.*, 2012). This theoretical gap remains across application contexts but is especially relevant in the comparably inexperienced hardware domains. The first research question addresses this lack of a theory based theoretical explanation. With its theoretical focus research question one defines the design theory selection and model construction for the research project.

Research question two: ***What constraints reduce agile design applicability how in automotive design?***

The second research question analyses the influence of the application context automotive design on agile product design. Instead of general assessments regarding the suitability of agile development in new application contexts it asks for empirically verified descriptions of limitations that are summarized, compared and classified according to the constraints of scale and physicality categories. Besides empiric evidence theoretical grounding is necessary to explain correlations and overlaps between both categories. The independence of the theoretical concepts constraints of physicality and scale are questioned for automotive application contexts and intersections between both constructs are identified. Consequently, the second research question addresses a comprehensive understanding of constraints to agility in automotive design and asks for a theory-based generalization of the research phenomena agile constraints in relation to and beyond a specific application context. Furthermore, with the second question word "how" research question two demands a design theory verified explanation of the experienced constraints.

Research question three: ***How to enable agility in automotive product design?***

Unlike the first two descriptive and analytical research questions the third research question aligns the research findings towards the discussion of new design artifacts to balance agile constraints in automotive. It builds on the findings of the first and the second research question regarding precise problem description from various perspectives and theoretical explanation of functionalities. The combination of these learnings allows to adjust the existing implementations according to the generated theoretical foundation to improve applicability in automotive design. Such a design construct includes theoretic contributions in the form of design theory extensions according to the findings and specific realizations of this adjustment. Added value is also included for practical applications since the adjustments are chosen according to the demands of practitioners who require straight forward solutions ready to implement.

3.1.2 Theoretical lens coordination theory

To answer the research questions coordination was chosen as a theoretical lens. Coordination theory was selected because most agile methods directly address or indirectly influence coordination structures in design. Relevant coordination activities span from coordination between individuals up to coordination in large, interconnected design systems. The coordination perspective enables to analyse and compare agile design structures on a level field based on the same theoretical foundations. The concept coordination strategy allows to systematise agile methods. This systematization is extended to account for the application context specific conditions for coordination. It allows to differentiate software specific from other application contexts such as automotive. To employ coordination as theoretical lens for agile design in different application contexts a standardized coordination model is necessary that is sufficiently sensitive to differentiate between the examined agile methods and the regarded application context. In the thesis at hand the employed coordination reference model was constructed based on different fields of knowledge in coordination theory to ideally reflect agile design structures. In summary, coordination theory suits the task to analyse agile design in the unfamiliar application context automotive design and recommend adjusted agile coordination structures that reflect the changes in automotive design compared to software design.

Regarding the first research question coordination theory was employed to analyse agile design structures and activities through a coordination lens and reflect them from different fields of coordination theory. This analysis allowed to connect empirically proven benefits with specific coordination characteristics of the respective methods. It also aided the iterative design of the coordination reference model and allowed to combine matching coordination theory aspects from adjacent theoretical fields to ideally mirror agile design structures. To explain agility in product design theoretically a generalized agile coordination strategy was developed and matched with central agile product design characteristics. Coordination theory also aided the research towards the second research question. The empirically collected agile constraints in automotive design were categorized with the help of the coordination reference model. This allowed to reference the set of problems towards their influence on the coordination efficiency of the employed agile methods and the respective agile coordination strategies. With the help of more general project characteristics several coordination determinants were matched to the categorized agile constraints. This allowed to determine agile coordination mechanisms unsuitable for automotive design contexts and explain the experienced constraints. To approach the third research question the connections between agile constraints and coordination determinants were used to recommend alternative coordination structures. The coordination reference model allowed to combine more suitable coordination mechanisms into adjusted coordination strategies for typical project settings in automotive design.

Similar approaches to employ coordination theory to reflect specific aspects of agility in software development or teamwork performance have been employed by Strode et al. (Strode *et al.*, 2012) Espinosa et al. (Espinosa *et al.*, 2007) Hoegl and Gemuenden (Hoegl and Gemuenden, 2001) and Kraut and Streeter (Kraut and Streeter, 1995). The employment of coordination theory to systematise not only agile design structures but also in relation to application context and product characteristics sets the thesis at hand apart from earlier research. To the best knowledge of the author this concept has not been employed before and should be applicable to similar Design Research approaches.

3.1.3 Research fields

The research in this study is structured into five research fields (see Figure 10) that address different bodies of knowledge. Inquiry into these interdependent research fields was not sequential. Throughout the research project research fields were revisited iteratively and theories adjusted according to new findings to maintain overall consistency despite heterogeneous progress in different fields. The following section describes the relevant research fields of the theses at hand and explains how they contributed to answer the research questions.

The first research field was shaped to answer research question one. It connects two focus areas: Agile product design and automotive product design. The focus area **agile product design** was studied to realize a comprehensive understanding of agility as an attribute and as a construct. This included a comparison of agility and agile methods to classic product design methodologies. Agile methods were studied to collect and categorize practices and analyse them as practical implementations of shared values and principles including those of the agile manifesto. Another important aspect agile product design is the origin of agile methods. Both the empiric development as well as the initial application context software development were analysed regarding their influence on agile product design. Additionally, agile coordination strategies were derived of central agile methods which allowed to generalize a representative agile coordination strategy independent of individual methods. Learnings in research field one mostly relied on theoretical inputs and some practical experiences. Regarding **automotive design** general characteristics and current dynamics of the overall design process were summarized, analysed and referenced to conventional and agile product development methodologies. The objective was to describe and categorize the application context according to specific characteristics to draw conclusions regarding the suitability of agile design practices. One aspect was the integral nature of the product and subsequent properties of the design process. Increasing internally and externally driven dynamics of automotive design were analysed as well. Dynamic markets and uncertainty in consumer behaviour, complexity in product design, uncertainty in technology development, digitalization of both product and product design as well as changing legal requirements were among the most relevant influences. Findings of research field one were published at the Vienna Motor Symposium in 2020 (Schrof and Paetzold, 2020).

Research field one indicated that agility in design suits the challenges of a more dynamic automotive product design in theory. Hence transferability from software to automotive design was studied in the second research field. To answer research question two the theoretical concepts **agile constraints of scale and physicality** were studied. Agile constraints of scale summarize limitations of agility in scaled application contexts consisting of multiple interdependent agile teams. Agile constraints of physicality summarize limitations of agile design driven by the physicality of the product in comparison to non-physical software products. A review of the existing literature showed that in several cases agile product design was not directly transferable to the automotive application contexts. Practical evaluations were necessary to understand dependencies and mechanisms that cause this inapplicability. Empirically observed problems in agile automotive design were systematically collected and classified according to the constraints of scale and physicality categories. Both constraints allowed to systematize and reflect design process and product characteristics that cause difficulties to agile design approaches. Specific characteristics of automotive design such as product integration and testing were examined to understand their influences on the limitations of agile design. Findings of research field two were part of several publications (Schmidt *et al.*, 2019; Schrof *et al.*, 2018, 2019; Schrof and Paetzold, 2020).

To answer research question two the theoretical constructs constraints of scale and constraints of physicality were applied to the collected data. Even though these categories origin from different application contexts both are evident in automotive design. They also cause similar mechanisms that limit applicability of agile design. This two-dimensional problem definition complicates a comprehensive understanding due to unclear cross dependencies between both concepts. To avoid a two-dimensional problem space constraint of physicality were analysed through a coordination perspective. This allowed to simplify the problem space and allowed to tread constraints of physicality as (inter team) coordination problems. Descriptions of agile constraints in automotive design collected in research field two, were used to verify this transformation of problem understanding. Theoretical and practical inputs were important in research field two since empiric data confirmed initial theories and reinforced the theoretical foundations of the research.

Research field three focused on **coordination in product design** to provide a theoretical lens necessary for all three research questions. The coordination perspective allowed to compare agile design practices and understand their benefits in design systems. To employ coordination as the theoretical lens of the research a **coordination reference model** was designed. To construct this reference model coordination theories research from different streams of product development and sociology were compared. Coordination mechanisms from organization research, team research and multiteam systems were connected to reflect the characteristics of agile values, principles and the practices of the most relevant agile methods. The connection of the coordination mechanisms to the respective coordination determinants based on the concept of Van de Ven et al. (Ven *et al.*, 1976) allowed to generate adaptive coordination strategies. These coordination strategies improved understanding of agile design and allowed to analyse agile constraints of scale and physicality in theory. Furthermore, agile constraints in automotive design were connected to their impact on the corresponding agile coordination strategies which enabled precise countermeasures. Findings of research field three influenced several publication (Schmidt *et al.*, 2019; Schrof and Paetzold, 2020, 2019).

Research field four concentrated on agile enablers for automotive design based on changes in product architecture and digital design tools. The research in this field increased understanding of agile constraints in automotive and allowed to develop strategies to overcome them. The first focus was the **correlation between product architecture and organization structure** and its influence on agile constraints. Agile product design requires specific team dynamics and structures. Intra team cooperation is emphasized and inter team distractions from outside the teams are avoided. Value creation is localized inside collaborative teams and agile practices are shaped to optimize intra team cooperation. Dependencies from outside the teams are not specifically addressed and therefore reduce agile product design applicability if unavoidable. This minimized inter team exchange and coordination relies on limited dependencies between design teams which is not realistic in automotive design. To address this contradiction product architecture and modularization strategies were researched to understand their influence on organizational dependencies and recommend approaches to reduce them (Schrof and Paetzold, 2019). The research stream relied mostly on theoretical inputs.

The second focus of research field four addressed agile enablers driven by **digitalized design procedures in automotive hardware** components. The sequential and hence time-consuming interplay between various design steps such as component construction, prototype manufacturing, system verification and production in automotive design was scrutinized and alternative, digitalized design tools were employed and evaluated. Component design practices were analysed to assess their impact on agile constraints. To reduce systemic dependencies (e.g. handovers and meetings) and waiting time for hardware prototypes usage of alternative design tools was evaluated. Such digital tools integrate construction and verification cycles in rapid design cycles and hence significantly reduce dependencies to verification, prototype manufacturing and testing units. This approach allows to apply software inspired design methods in hardware applications. It realizes software alike digital testing and integration infrastructure close to the original application context. Practical evaluations in pilot projects were essential and findings were published by Schrof et al. (Schrof *et al.*, 2019). Research field four increased understanding of agile constraints with an emphasis on constraints of physicality and hence applies to research question two. The research also resulted in opportunities to avoid agile constraints in automotive which links the findings to research question three.

Research field five focused on coordination in agile automotive design and a **coordination strategy for agile automotive design** was developed. The research field addresses all research questions. The derivation of coordination strategies in scaled and non-scaled agile methods further increased understanding of agility working mechanisms and hence supported research question one. Using coordination as a theoretical lens to explain the problem space and to understand causes and effects addressed research question two. Most importantly, the generated coordination strategy is the central research result to overcome agile constraints in automotive design and therefore addressed research question three. The coordination perspective allowed to simplify problem space and create specific solutions for the agile constraints in automotive design. This incremental design artefact development generated a comprehensive understanding of coordination requirements in agile automotive design. The flexible structure of this construct allows further adjustments according to project specifics. To ensure functionality practical evaluations in various pilot projects were conducted. Research field six was characterized by a continuous shift between practical assessment, theoretical solution development and practical evaluation throughout the iterative development of the overall coordination strategy.

3.1.4 Research overview

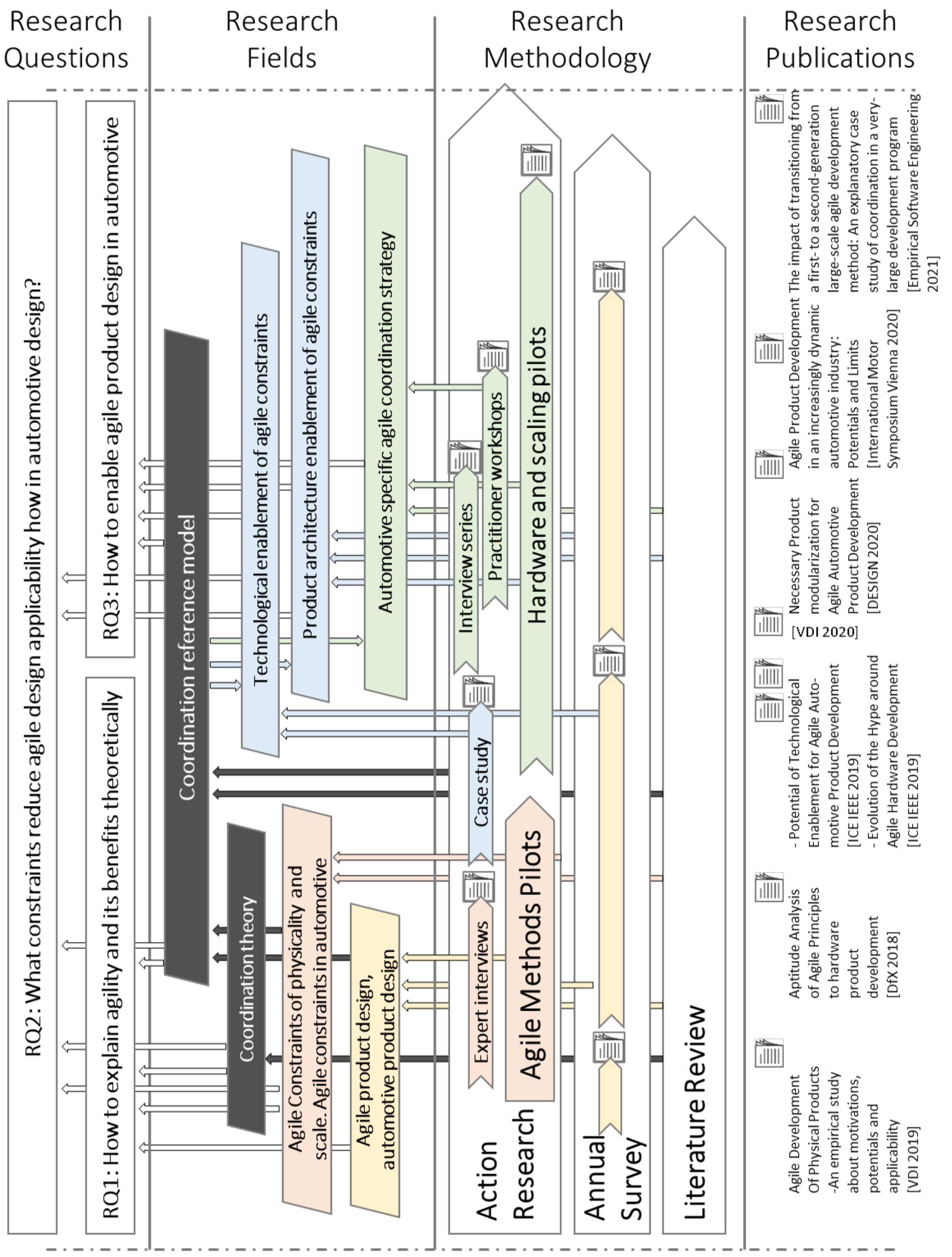

Research Questions
Research Fields
Research Methodology
Research Publications
RQ2: What constraints reduce agile design applicability how in automotive design?
RQ1: How to explain agility and its benefits theoretically
RQ3: How to enable agile product design in automotive
Coordination reference model
Coordination theory
Technological enablement of agile constraints
Product architecture enablement of agile constraints
Automotive specific agile coordination strategy
Agile Constraints of physicality and scale. Agile constraints in automotive
Agile product design, automotive product design
Action Research
Annual Survey
Literature Review
Agile Methods Pilots
Hardware and scaling pilots
Case study
Expert interviews
Interview series
Practitioner workshops
Agile Development Of Physical Products -An empirical study about motivations, potentials and applicability [VDI 2019]
Aptitude Analysis of Agile Principles to hardware product development [DfX 2018]
- Potential of Technological Enablement for Agile Automotive Product Development [ICE IEEE 2019]
- Evolution of the Hype around Agile Hardware Development [ICE IEEE 2019]
[VDI 2020]
Necessary Product modularization for Agile Automotive Product Development [DESIGN 2020]
Agile Product Development in an increasingly dynamic automotive industry: Potentials and Limits [International Motor Symposium Vienna 2020]
The impact of transitioning from a first- to a second-generation large-scale agile development method: An explanatory case study of coordination in a very-large development program [Empirical Software Engineering 2021]

Figure 10: The research overview represents a comprehensive summary of the research project. It connects the research questions with the respective research fields, the research methodology and published scientific papers.

The research was organized according to the research overview in Figure 10. The overview connects the research questions with the visited research fields and clarifies how these research fields were advanced methodologically. The research overview emphasises the systematic approach of the complete research project and underlines the systematology between research questions, research fields and research methodology. The research overview also reflects how research in one research field is based on findings of other research fields and how findings were used to select and approach the next research field. Additionally, the research overview links the published publication of the research to the respective research fields, research questions and employed methods. The following section details the connection between research questions, research fields and methodological approach. The detailed research methodology and employed data collection and analysis methods are described in the following subchapters.

To approach **research question one** a thorough understanding of agility in product design was necessary. This was accomplished through the investigation and connection of the research fields agile product design, agile constraints of physicality and scale and coordination theory. The research field agility in product design was guided methodologically by the literature review which provided a good theoretical understanding. Additionally, Action Research was employed through combined pilot projects and expert interviews in automotive and software application contexts to complement a practical understanding of agile design. The practical experiences also aided the verification of described benefits in real world application contexts. The yearly industry survey influenced understanding of agile design since it allowed to validate practical data beyond automotive design and the central research partner company. Besides agile product design, the research field constraints of physicality and scale provided necessary understanding of agility in product design to answer research question one. Constraints of scale were approached through the literature review and Action Research with a focus on the literature review. Constraints of physicality were approached through the same methods but with a reverse focus on action research since the research field is less mature than constraints of scale. The literature review also dominated the methodological approach to investigate the research field coordination theory. But Action Research activities were central to the adjustment and connection of different fields of knowledge in coordination to reflect relevant structures and the overall systematology of coordination in agile design.

Research question two investigates the applicability of agility in automotive design. To answer it the understanding of agile design from research question one was extended to account for influences of automotive application contexts on the requirements and functionality of agile design. To answer research question two findings of the research fields agility in product design, automotive product design, agile constraints of scale and physicality, agile constraints in automotive and coordination theory were connected. The research field automotive product design relied on the literature review and Action Research methodologically. The literature review focused on conventional product design methodologies and their formalized models. The Action Research focused on the practical implementations and nesting of these models in the automotive design context. Expert interviews, pilot projects and a case study improved led to a complete picture of automotive design within the partnering company. Based on the findings of the research field agile constraints of scale and physicality the research field agile constraints in automotive relied on the literature review and the Action Research methods. The Action Research pilot projects were employed to identify agile constraints in automotive and compare them to constraints of scale and physicality. The literature review was employed to verify these findings with publications from other practitioners in similar industries. The methodological approach to the research fields agility in product design, agile constraints of scale and physicality and coordination theory to address research question two did not differ from the descriptions for research question one above.

To address **research question three** the findings of research questions one and two were connected. The research focus connected the research fields coordination theory (coordination reference model), technological enablement of agile constraints, product architecture enablement of agile constraints and automotive specific agile coordination strategy. The research field technological enablement of agile constraints was based on a case study. Additionally, the literature review provided theoretical inputs to verify alternative or novel technology from comparable publications. Product architecture enablement of agile constraints also relied on the literature review to identify similar approaches in other publications. The yearly industry survey was employed to verify the relevance of the concept. Nevertheless, action research was the central methodology to approach the research field. Several hardware and scaling pilot projects were analysed regarding the mutual influence of product and project architecture. The last research field automotive specific agile coordination strategy relied on inputs from all other research fields. Therefore, a direct connection to research methods is difficult. In general, action research was central to develop and verify the adjusted agile coordination strategy for automotive design. Practitioner workshops were employed to complement the results of the pilot projects.

3.2 Design Research

"Science is common sense in combination with systematics."
Jensen

The presented research strategy is based on a collaborative Design Research project with the industrial partner BMW Group. Such a Design Research project differs from the classic perception of research in natural science in various aspects. The following subchapter describes these differentiations transparently and underlines the scientific validity of Design Research. Design Research is described based on its ontological and epistemological positions to delimit it from other research fields. The described Design Research paradigm also guides the selection of appropriate research methods in the research methodology section. Furthermore, scientific rigor and relevance of Design Research methods are analysed to ensure scientific validity of the research findings in this study.

3.2.1 Research paradigm

The terminology research paradigm refers to a broad framework of perception, understanding, and belief within which theories and practices operate. It is a network of coherent ideas about the nature of the world and the functions of a researcher (Bassey, 1990). The research paradigm clarifies the connection between research strategy and methodology as well as the corresponding philosophical standpoint. Each methodology is based on a philosophic perspective that supports its premises and research logic. Therefore, the alignment of ontology, epistemology and methodology with the research objective is crucial to guarantee a valid, interlocking research paradigm. The given research context has a significant influence on this construct since it predefines certain aspects.

Ontology refers to the philosophical understanding of reality and what sort of things exist. This includes assumptions about the form and the nature of reality. Ontology is concerned with whether reality exists independently of human understanding and social interpretation (e.g. is there a shared social reality or multiple context-specific realities). Snape et al. divide three distinct ontological positions (Snape and Spencer, 2003). Realism claims that there is an external reality independent of what people may think or understand it to be. Idealism maintains that reality can only be understood via the human mind and socially constructed meanings. Materialism claims that there is a real world, but it is only the material or physical world that is real. Other phenomena, for instance, beliefs, values or experiences arise from the material world but do not shape it. In Design Research realist and idealist ontological positions present different interpretations of relevant system elements in their contexts. Iivari et al. present a framework to clarify the contradictory positions in the same context (Iivari *et al.*, 1998). Realism interprets data and information as (relevant) descriptive facts, information systems as technical systems, human beings as deterministic systems, technology as a causal agent and organization and society as stable structures. Contrary to this position Idealism interprets data and information as socially constructed meanings signifying intentions, information systems as a form of social systems, human beings as voluntarist systems with consciousness and free will, technology as malleable structures subject to social and human choice and organization and society as interaction systems or socially constructed systems.

Epistemology concerns itself with the nature, the acquisition, the limits, and the grasp of knowledge. Positivism and Constructivism (Interpretivism) are the two opposing archetypes. The traditional positivistic approach originates from the natural sciences and the constructivist approach was established as a critical response to the positivistic tradition. Positivism relies on replicable empirical evidence and endeavours to be objective. It explains and predicts the social world by searching for regularities and causal relationships. It was originally established by Auguste Comte (1798-1858) and others as a counterbalance to religious dogmas and metaphysical speculations. Logical positivists developed the concept of verification as the basic premise of scientific knowledge. This implies that true knowledge has to be empirically verified through direct observation (Kvale and Brinkmann, 2009). Constructivism interprets knowledge as social constructions and relies on learning through social interaction. Observations are subjective and therefore cannot represent an absolute truth. This criticizes the traditional ideal of research and supports a new understanding of how to perceive science and the act of research (Iivari *et al.*, 1998). Social constructivists search for relations and certainty, knowing that they will never obtain any absolute knowledge or certainty. The term constructivism is helpful because it clarifies the basic principle that reality is socially constructed and that there is no external reality independent of human consciousness (Robson, 2011). This implies that a social world is only understandable from the point of individuals involved in the researched activities. The neutral observer standpoint from the positivist position is therefore impossible. To understand requires occupying the frame of reference of participants in action. Understanding happens from inside not outside of researched systems.

Iivari et al. define **research methodology** as a set of goal-oriented procedures that guide the work and cooperation of the various parties involved in the building of an Design Artifacts (e.g. an application) (Iivari *et al.*, 1998). They report three categories to classify the numerous methods in Design Research: Nomothetic, Idiographic and Constructive methods. **Nomothetic** methods, including formal mathematical analyses, experimental methods (laboratory and field experiments), and nonexperimental methods such as field studies and surveys, are epitomized in the approach and methods employed in the natural sciences, which focus upon the process of testing hypotheses in accordance with the canons of scientific rigor. **Idiographic** methods such as case studies and action research place considerable stress upon getting close to one's subject and exploring its detailed background and life-history" (Burrel and Morgan, 1979). Close to an idealist ontology, **constructive** methods are concerned with the conceptual (models, frameworks, and procedures) and technological engineering of artifacts. As artifacts, they do not describe any existing reality but rather help to create a new one. These method categories also result in different data classes. Nomothetic research methods provide quantitative data, because of their emphasis on systematology. Idiographic methods focus on contextuality and usually collect qualitative data. Greenhalgh et al. added that researcher who employ qualitative research search deeper truths while aiming " *to study things in their natural setting, attempting to make sense of, or interpret, phenomena in terms of the meanings that people bring to them*" (Greenhalgh and Taylor, 1997). Gilbert emphasized that qualitative researchers seek to uncover the world through another's eyes, in a discovery and exploratory process that is deeply experienced (Gilbert, 2000). Ottosson et al. recommend quantitative research to screen areas and qualitative research to get a deeper knowledge of studied phenomena (Ottosson *et al.*, 2006).

3.2.2 Design Research paradigm

Friedman **defines design** in a broad sense. It includes "*solving problems, creating something new, or transforming less desirable situations to preferred situations*" (Friedman, 2003). He stated that most design definitions include three common steps. First, design refers to a process. Second, this process is goal oriented. Third, the goal of design is to solve problems, meet needs, improve situations and create something new or useful. According to this definition the term design refers to the comprehensive product development process and not the aesthetic understanding of design as an art or craft in this study.

Design Research is an umbrella term that comprises **design science and behavioural science** in product development. It is motivated by the desire to improve the practical environment by the introduction of new and innovative artifacts and the processes for building these artifacts (Simon, 1996). Hevner et al. differentiates Design Research from other research positions by its pragmatic nature and its emphasis on practical relevance. He emphasizes that unlike theoretical approaches Design Research is supposed to deliver a clear contribution into the application context (Hevner *et al.*, 2004). This implies a clear delimitation from the positivist position of natural sciences and a shifted balance between relevance and generalizability, rigor and theory. The socio-technical nature of product design requires adapted research approaches. Design Research is object to characteristics of human behaviour in socio-technical systems research. Particularities of human behaviour such as illogical or irregular conduct, driven by unpredictable social influences and unknown earlier experience, cannot be addressed with a realist ontology.

Behavioural science and design science paradigms characterize much of Design Research especially in the Information Systems discipline (Hevner *et al.*, 2004). The behavioural science paradigm seeks to develop and verify theories that explain or predict organizational or individual human behaviour. The design science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative design artifacts. In this study the term design artifact includes both behavioural theories as well as practical artifacts. The two paradigms differentiate in their goal. While behavioural science seeks truth, design science seeks utility.

Design Research covers **conceptual knowledge**, **descriptive knowledge** and **prescriptive knowledge** as three levels of knowledge (Iivari, 2007). The conceptual level comprises concepts, classifications, taxonomies and conceptual frameworks. It relies on conceptual theories for analysing and predicting design. The descriptive level includes observational facts, empirical regularities, and causal laws. It results in descriptive theories for explaining and predicting. The prescriptive level supports how things could be and how to accomplish them. It designs alternative artifacts to achieve certain utilitarian ends. It is based on prescriptive theories for design and action. Prescriptive knowledge emphasizes truth regarding efficiency and effectiveness above absolute truth value. Whilst design science emphasizes prescriptive knowledge, behavioural science is about descriptive and conceptual knowledge. Design theories consist of knowledge of practical character. They are aimed for and related to design activities and as such they are practical theories as described in the pragmatic tradition (Cronen, 2001). The value of practical theories lies in their usefulness for inquiry processes.

3.2.3 Relevance and rigor in Design Research

The research environment in Design Research is not static and predictable but dynamic and continuously shifting. Therefore, research in this domain is in constant peril to generate irrelevant or outdated theories and artifacts. This thread is reinforced by the imprecise problem understanding caused by the nature of product design which is object to probabilistic, unclear and ambivalent influences. Consequently, research assumptions might be incorrect or object to change. Especially, theoretical research is affected by imprecise inputs and dynamics. Outdated or incorrect assumptions may result in research projects irrelevant to both theory and practice. This **relevance challenge** of Design Research requires the research to be adjusted to the application domain regarding research project dynamics and research assumptions sensitivity. Methods that simultaneously build design artifacts together and within an organizational application context while learning from the intervention avoid the relevance challenge (Baskerville and Pries-Heje, 1999).

Most Design Research methods such as Blessing's Design Research Methodology (Blessing and Chakrabarti, 2009) are based on linear stage-gate models in that they separate and sequence building and evaluation of design artifacts. The three basic stages are problem definition, conceptualization and generation of design artifact and evaluation. This sequencing separates problem understanding from shaping design artifacts. It might lead to incomplete or imprecise problem understanding since information may only become relevant during the generation and evaluation of the design artifacts. Fixed stages complicate or restrict retrospective adjusting of initial premises or assumptions. Sequential methods emphasize scientific rigor at the cost of relevance and research consistency. Iterative research methods such as Action Research circumvent this **sequencing challenge** in Design Research. This opposing approach assumes that the design artifact emerges from interaction with the organizational context even if its design is guided by the researchers' initial intent. Such methods are based on repetitive small research cycles from problem understanding to solution evaluation. These iterations allow to evenly expand problem understanding and design artifact functionality.

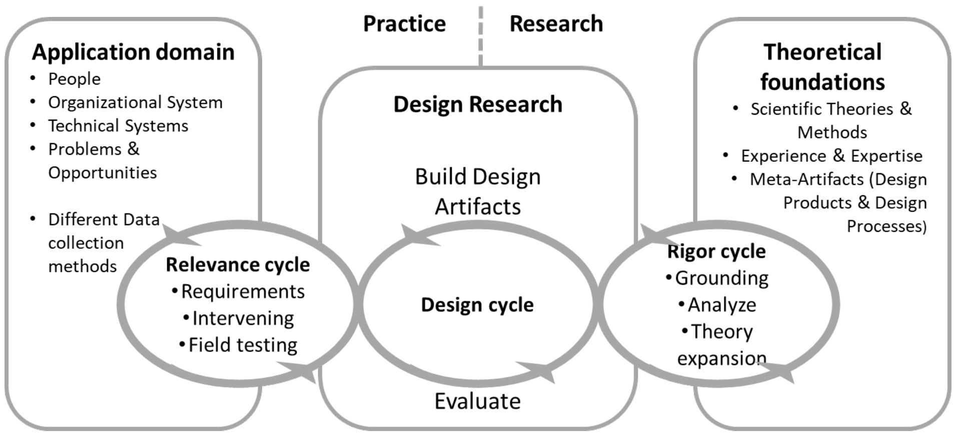


Figure 11 Combination of Design Research cycles according to (Hevner Alan, 2007). The relevance, design and rigor cycle ensure balance between practical and theoretical requirements in Design Research projects.

Hevner integrates the relevance and sequencing challenge into a comprehensive Design Research model that confirms the pragmatic nature of Design Research (Hevner Alan, 2007). He emphasizes that practical utility cannot be the unique aim of research. A synergy between rigor and relevance is necessary. He analyses

Design Research as an embodiment of three closely related cycles of activities (see Figure 11). The **Relevance Cycle** continuously bridges the design science activities and the contextual environment of the research project. It derives requirements from the application context into the research and serves to field test and further develop generated design artifacts. The application context consists of people, organizational systems and technical systems that interact. The cycle also identifies initial research opportunities and acceptance criteria for evaluation. The acceptance criteria are an important input to decide whether additional research cycle iterations are necessary or to stop the research. They prevent unnecessary fine tuning and therefore provide efficiency of the Design Research. The **Rigor Cycle** supports theoretical grounding, theories, scientific methods and experience from the research domain and integrates new knowledge into the existing knowledge base. With state-of-the art application expertise and existing artifacts and process knowledge it comprises two additional, application domain specific knowledge types. It informs the research activities with scientific foundations from the existing, past knowledge base and differentiates the research project from routine designs based on application of well-known processes (Hevner *et al.*, 2004). The **Design Cycle** is positioned between the Relevance Cycle and the Rigor Cycle. It is the central part of the research project and accommodates the activities to construct and evaluate design artifacts. It iterates between building and evaluating the design artifacts adapting to the inputs of the Rigor and the Relevance cycle. It is necessary to maintain a balance between constructing and evaluating design artifacts and ensure that they are based on rigor and relevance. A thorough evaluation is insufficient without a grounded argument for the construction of the artifact.

Goldkuhl and Lind conceptualize design theory as theorized practical knowledge that requires grounding in the existing body of knowledge and practical utility. They agree with Hevner's concept of a rigor cycle to theoretically ground constructed artifacts. But they question whether the presented existing knowledge base within a design theory is sufficient. They claim the need for a broader perspective on grounding and present the concept of multi-grounded Design Research. They present three types of knowledge sources as premise for three types of grounding processes (Goldkuhl and Lind, 2010). **Empirical grounding** comprises grounding through practical evaluation of design artifacts in their application context, which matches Hevner's relevance cycle. **Theoretical grounding** summarizes explanatory grounding of descriptive knowledge based on external, non-design theories, concepts and values. **Internal grounding** evaluates the consistency and cohesion of the design artifact with the existing body of knowledge of design theory. Compared to Hevner's Rigor Cycle the differentiation into theoretical and internal grounding provide a more specific understanding of the theoretical grounding of emergent design artifacts. Goldkuhl specifically mentions both approaches while Hevner does not mention or exclude either. The ability to also employ external theories allows to explain certain aspects and functionalities which are proven empirically but cannot be explained within the existing design theory thoroughly. Therefore, the multi-grounding concept of Godlkuhl and Lind is integrated into the research paradigm of this research project.

3.3 Research methodology

"A process cannot be understood by stopping it.
Understanding must move with the flow of the process, must join it and flow with it."
Frank Herbert

The research in this study is based on an Idealist ontological position since human interactions in socio-technical systems are a central objective. This implies that reality and facts are interpreted as socially constructed which has a strong influence on data collection and analysis methods. The research is based on an epistemology position based on Constructivism. Relevant knowledge includes socio-technical and organizational systems. Knowledge sources are the behaviour of individuals in social systems and the overall system dynamics. Hevner's combination of research cycles was chosen as an overall research logic to realize practical utility and scientific knowledge derivation (Hevner Alan, 2007). Even though the model explains interdependencies in Design Research it does not specify the means to implement the concept in research projects. To realize the concept in a research project a suitable research methodology is necessary. It must reflect the restrictions of the application domain, the research project and the research strategy. In this study Action Research was chosen as central research methodology in a multimethod research approach which is shown in Figure 12. The following section explains how the methodology suits Hevner's research principles in general. Detailed descriptions of the Action Research methodology and comprehensive research methods follow in the subsequent subchapters.

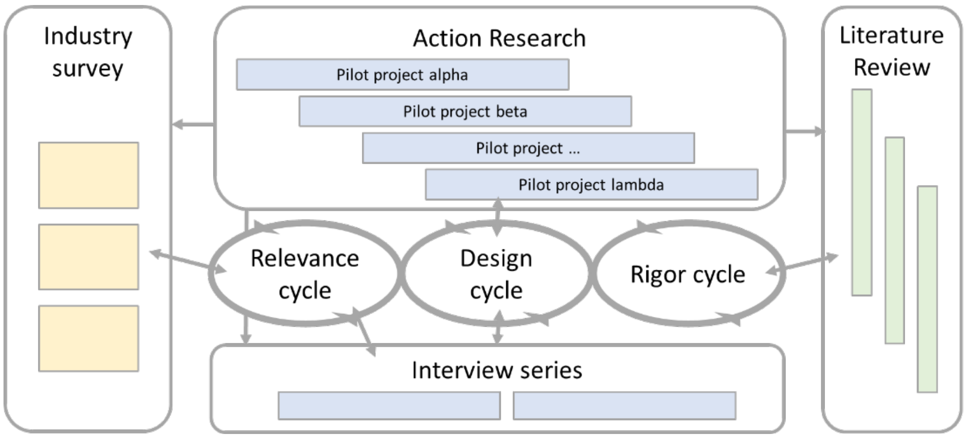


Figure 12: Employed research methodology based on Action Research with grounding in complementary research methods.

Action Research as research methodology was chosen to integrate and realize Hevner's theoretical Design Cycles construct within the given application context. **Action Research** connects the design science activities with contextual environment of the research project through the close collaboration of the researcher with designers within the design project. The close connection promotes the identification of relevant research opportunities and its iterative questioning of relevance ensures research efficiency. Design requirements are transferred from the application context into the research and resulting design artefacts are field tested in the opposite direction. The Action Research data was enriched by an interview series and an annual industry survey independent of the partnering company. Both data sources combined with the Action Research findings represent Hevner's **Relevance Cycle**. But the implemented Action Research methodology also addresses theoretical grounding of practical findings and leads to extensions of the relevant body of knowledge. The iterative action cycles are matched by a continuous literature review to evaluate practical findings with relevant theories or compare them to empiric results from similar environments. This repeated grounding in theory differentiates the Action Research from design activities and reflect Hevner's **Rigor Cycle**. The central objective of Action Research is to construct and evaluate design artifacts to improve design activities within the application context. The researcher participates in design projects and introduces these design artifacts into the real-world design activities. This change aides understanding of the research phenomena, the influence of the design artifact on it and the relevance of the selected design theory. The introduction of change in the form of design artifacts, the iterative evaluation of this change and the corresponding adjustment of the design artifacts in Action Research incorporate Hevner's **Design Cycle**.

The methodology of the research of connects nomothetic, idiographic and constructive research methods. Annual surveys including a broad industrial audience were conducted to verify the overall relevance of the research problem and test hypotheses. These surveys apply to the class of nomothetic methods. The central part of the research strategy was realized within a comprehensive Action Research frame including both case studies and interview series. These idiographic methods were applied to understand the research problem from a practitioner perspective and test specific solutions. Based on the overall problem relevance and the specific adaptions constructive methods were engaged to generate a model that enables the transfer of understanding and implementation to other cases.

3.3.1 Action Research method

Action Research as a research method aims to both **solve current practical problems and expand the scientific knowledge base**. This dual mission includes contributions to theory and assistance in current and anticipated problems of practitioners (Benbasat and Zmud, 1999; Rosemann and Vessey, 2008). The direct practical utility embeds relevance into research projects. Sein et al. conceptualize Action Research as containing inseparable and interwoven activities of constructing emergent design artifacts, intervening within the organization and evaluating the impacts (Sein *et al.*, 2011). In this concept design artifacts are dynamic and emerge from the context of both their initial design and continual reshaping from organizational application.

The fundamental contention of Action Research is that a **complex social process can be studied best by introducing change and observing the effects**. The action researcher purposely creates organizational change and hence discontinues the objective position of the researcher as an external observer. The implications of these actions allow the researcher to better understand the application context, the structural characteristics of the product design process and the effect of the design artifact on it. This in turn increases intertwining between practical problems and theoretical solutions (Babüroglu and Ravn, 1992). The researcher becomes part of the researched object and intervenes to solve immediate and anticipated organizational problems (Baskerville and Pries-Heje, 1999).

Action Research accepts the inability to completely understand a dynamic socio-technical system. Instead, it emphasizes proximity and researcher action to increase understanding of defined parts. Ottosson et al. claim that the observer of reality is at the same time part of reality because of the nature of product design which is driven by human interaction. Passive observation is therefore impossible which confutes a positivistic view of objectivity. Instead Action Research accepts the researcher as being part of the researched object and strives for generalizability by rigor in data analysis, strong theoretical grounding and empirical evaluation (Ottosson *et al.*, 2006). Consequently, Action Research is strongly oriented toward **collaboration between researcher and product designer**. This changes the recognition of the product designer compared to other Design Research methods. The role of product designer changes from researched object to constructive partner of the researcher. This shift is essential to the research logic since both the knowledge of the researcher and the product designer are valued crucial for the generation of relevant design artifacts. Action researchers contribute methodological knowledge and design theories and product designer complement situational, practical knowledge and application context experience. The combination of both knowledge sources complies with Hevner's concept of a Rigor and a Relevance Cycle. In a Design Research project both are necessary and the repeated back and forth between them increases scientific and practical quality of design artifacts. Since Action Research purposely triggers change this affects the product designer as well as the researcher. Both have to readjust to a new situation together and draw learnings from the product design and the Design Research perspective.

The role of the action researcher within the action research project is not predefined and may vary between project management and sporadic observer according to the application context and research settings. Ottosson et al. define four differentiations of the researcher role: Project lead, team member, observer with more than 80% presence and observer with sporadic presence (Ottosson *et al.*, 2006). According to these roles they categorize three levels of Action Research integration into the design project: Action Research, Insider

Action Research IAR and Participatory Action Research PAR. All four role definitions support basic Action Research. Insider Action Research requires at least a high presence of the researcher as an observer. Participatory Action Research PAR requires the researcher to be a productive part of the product design team. Loss of valuable information due to incorrect reconstructions is minimized and first-hand data collection methods are applicable. Only team members or the project leader fulfil the requirements of this integration level. The participatory research position allows to grasp even the smallest details and understand product and project related interdependencies. Action and stimulus are much easier implemented and analysed from within the design team. Nevertheless, this position required additional research capacity and the researcher's interest may be divided between research and project progression.

Action Research is an **iterative research process** that capitalizes on learning by both researchers and subjects within the context of the subjects' social system (Davison *et al.*, 2004). Iterations allow to separate smaller cycles within a research project that have distinct objectives. Ideal iterations include each of Hevner's design cycles and additional knowledge is won. This iterative research nature is based on working hypotheses which are refined over repeated cycles of inquiry. Overall understanding of the research object increases iteratively and assumptions and consequently design artifacts are updated to an emerging knowledge base. Ottosson et al. report the refinement of the research question with an increasing understanding of the research field through subsequent iterations. An initially open research question gradually develops and can with time be broken up into more specific questions. They emphasise how this refinement increases compatibility of research paradigm, research questions, experiences and design artifacts (Ottosson *et al.*, 2006). Checkland et al. support this approach and report initial research themes, instead of fixed research hypotheses. These themes are shaped into specific research questions throughout the research project. They also emphasize the importance the connectedness and fine tuning between application context, research strategy and methodology to generate **generalizable and valid learnings**. A serious and organized Action Research process is essential to present defensible generalizations. (Checkland and Holwell, 1998). Eden et al. supplement that any tools, techniques, or models developed need to be linked to the research design. Exploration of data and theory building has to be explainable to others. Method triangulation is used if possible (Eden and Huxham, 1996).

In a **nutshell** Action Research as a research method addresses practical relevance and scientific rigor in Design Research. It links theory with practice, and thinking with doing (Susman, 1983). Consequently, it avoids shortcomings of other Design Research methods such as practically irrelevant theoretical constructs or disconnected research phases (e.g. problem definition, artifact construction and evaluation phases) in sequential research phases (Baskerville and Myers, 2004). It leads to descriptive knowledge regarding the precise understanding of a design phenomenon in its application context and prescriptive knowledge regarding the generation and evaluation of design artifacts. The iterative method ideally incorporates Hevner's Design Research concept. The repeated shift between research cycles generates scientific understanding and practical utility. Additionally, a mutual grounding of practice and theory is realized as intended by Hevner.

According to Sein et al. an Action Research project can be divided into four overall stages that each rely on characteristic principles (Sein *et al.*, 2011). These stages do not imply a fixed sequence but rather predefined sets of activities. The action researcher is required to alternate between activities and stages according to findings and the dynamics of the research project.

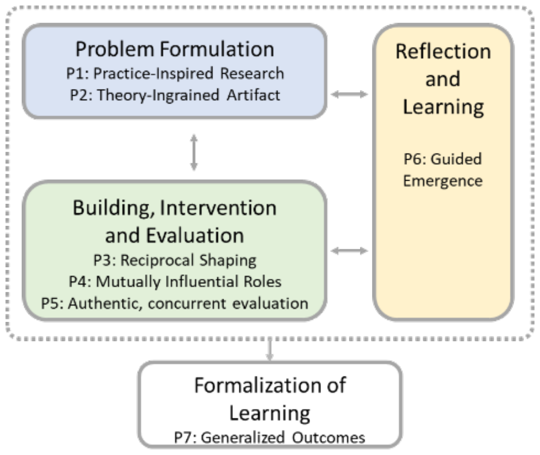

Problem Formulation
P1: Practice-Inspired Research
P2: Theory-Ingrained Artifact
Reflection and Learning
P6: Guided Emergence
Building, Intervention and Evaluation
P3: Reciprocal Shaping
P4: Mutually Influential Roles
P5: Authentic, concurrent evaluation
Formalization of Learning
P7: Generalized Outcomes

Figure 13: Action Research conduct according to (Sein et al., 2011). In practice the stages are not sequential but concurrent in Action Research.

The **Problem Formulation** stage represents the identification of the research phenomena and its articulation as an instance of a class of problems in design theory. The trigger is a problem perceived in practice or anticipated by researchers. Its formulation as a research problem conceptualizes a research opportunity based on existing theories. Within this stage the scope of the research is initially defined, the role of the researcher within the design project is decided, an initial research question is formulated and the long-term commitment between the researcher and the industrial partners is agreed on. Even though a great share of the research strategy and assumptions are defined initially in this this stage they remain flexible to accommodate later change. This stage is based on two principles. **Practice-inspired Research** emphasizes the practical origin of research opportunities instead of theoretical constructs. The researcher has to realize a design artifact that is both able to apply to a specific solution within the research project and answer to a generalized class of problems. **Theory-ingrained Artifact** implies that design artifacts are informed by theory. This principle recognizes the integration of prior theories to structure the problem, to identify solutions opportunities and to guide design of artifacts. This theory input happens during the initial design of the artifact and is repeated as an answer to new learnings during intervention, evaluation, and further shaping.

The **Building, Intervention and Evaluation** stage generates an initial design of the design artifact. During the following iteration it is reshaped by intervention, evaluation and rebuilding cycles. Within this stage organization dominant artifacts are created with design knowledge from organizational intervention. This stage draws on three principles. **Reciprocal Shaping** emphasizes the inseparable mutual influences of design artifact generation and organizational intervention. A change within one part directly causes adaptions within in the other and vice versa. **Mutually influential Roles** signifies different bodies of knowledge from practical and research side within the project. It also allows individuals to play multiple or different roles. **Authentic and Concurrent Evaluation** is a central objective of Action Research that signifies that evaluation is not a separate stage but a continuous part of the construction of design artifacts resulting in anticipated and not anticipated adaptions. Evaluation should be addressed whenever possible and authenticity is emphasized above controlled setting.

The **Reflection and Learning** stage focuses on transferring design knowledge from a particular instance to a broader class of problems. Essentially this is done continuously during the other stages. This stage clearly differentiates Action Research from project management or simple problem solving. It enables a comprehensive understanding of the research construct from problem framing and design artifact emergence to the theoretical additions to the body of design knowledge. It also represents an important feedback mechanism to the research project and therefore increases understanding of the connection between research problem and design artifact. Finally, this stage allows to assess the need for additional design cycles and therefore provides efficiency to the research project. **Guided emergence**, the interplay between the seemingly conflicting perspectives design and emergence, is the central principle of this stage. Design artifacts reflect both the initial, theory-informed design of the researcher and the reshaping during practical evaluation cycles by organizational use. Walls et al. include changes to design, meta-design and meta-requirements alongside action and evaluation cycles that can have significant impact on the initial design artifact (Walls *et al.*, 1992). During the evolvement of the design artifact new design principles can be conducted. This principle indicates the need for sensibility of the action researcher to signals that indicate the need for refinement.

The **Formalization of Learning** stage aims to formalize new knowledge. Van Aken refers to the development of general solution concepts for classes of field problems from (individual) research project situated knowledge. This includes outlining the accomplishments of design artifacts and organizational outcomes from the application context to formalize knowledge (Aken, 2004). These results need characterization as design principles and with further analysis and reflection as adaptions or comprehensions of design theories applied to the design artifact. Tasks include sharing of assessments with practitioners, articulation in reference to selected theories and publication of formalized results. This stage draws on **Generalized Outcomes** as central principle. While the design artifact represents an ensemble to address a practical problem both need to be generalized, which is difficult due to the situational nature of Action Research. Sein et al. suggest three level to move from specific to generic and abstract knowledge (Sein *et al.*, 2011). First, generalization of the problem instance to a class of problems. Second, generalization of the solution instance to a class of solutions independent of the organization-specific solution. Third, derivation of design principles from the research project. This requires reconceptualization of the deduced case-specific knowledge to the defined class of solution.

3.3.1.1 Selection of pilot projects

Research activities were based on product design projects (agile pilot projects) that offered the chance to change the status quo product design approach according to agile principles, practices or complete agile methods. The researcher implemented an initial change in working practices to these pilot projects and repeatedly adapted this change according to system reactions. This action across pilot projects aided the understanding of the research problem and the generation of a respective design theory.

The pilot projects were chosen according to specific selection criteria. First, the projects had to be object to agile constraints of scale or physicality. Thus, large software product design projects, including several sub-projects or smaller but mechatronic product design projects were chosen. Second, the degree to how much change was accepted in the projects was relevant. Projects that only allowed minimal action were excluded. This criterion strongly reflected the motivation of the responsible management to try alternative design strategies. Third, the applicability of (novel) design tools that enabled digitalization of previously manual tasks or testing was another selection criterion. Fourth, the level of dependencies to other organization units influenced pilot project selection. Project independence was focused in the first year of the overall research project. In the second- and third-year project selection focus shifted to inter team dependencies (dependencies between teams). Hence, larger, connected projects were chosen. Fifth, the duration of the projects was another criterion. Only projects that lasted at least several weeks were selected. This criterion was important since in shorter projects initial team motivation, access to teams and earlier knowledge of agile working models might have caused indeterminable and inconsistent influence onto the results.

Change was introduced to the selected pilot projects in three intensity levels. Independent of the chosen level the applied change was adjusted thoroughly to the specific requirements of each individual pilot projects. The first level included agile practices and principles that were introduced to existing teams. This included Kanban boards, Retrospective meetings, iterative design cycles or complete agile methods such as Scrum amongst others. The second level of change affected besides agile practices and principles digital and non-digital working infrastructure. Team colocation and innovative work environments were applied, and novel software tools were introduced to enable continuous product integration and testing to non-SW application contexts. The third level of implemented change included modifications of the existing organization structure. Hierarchy structures and project management standards were revoked and agile roles such as Scrum Master or Product Owner were introduced permanently. Multi project allocations of designer were discontinued.

Table 6: Set of agile pilot projects in automotive design. The table details the chosen Action Research mode, the applicable agile constraint category, the initial project management motivation to change, the employed agile tools, the organizational dependency level, and the project duration.

Pilot Project	Action mode	Action level	Agile constrains	Motivation to change	Digital tools	Dependency level	Project duration
alpha	AR	two	CoP	high	CAD/E tools	high	two months
beta	IAR	two	CoP, CoS	medium	Simulation tools	high	three months
gamma	PAR	two	CoP	medium	CAD/E tools	medium	four weeks
delta	IAR	one	CoS	low	CAD/E tools	high	two months
epsilon	AR	two	CoP	low	-	medium	three months
zeta	IAR	three	CoP, CoS	high	CAD/E, simulation	medium	four months
eta	PAR	three	CoP, CoS	medium	CAD/E tools	high	six months
theta	PAR	three	CoP	medium	Generative CAD	medium	three months
iota	IAR	one	CoP	low	CAD, simulation	high	two weeks
kappa	AR	three	CoS	high	Agile tool chain	high	indefinite
lambda	AR	three	CoS	medium	Agile tool chain	high	indefinite

In total eleven projects were researched in detail. Details of the individual design projects are given in 5.1. Findings in this study are based primarily on these pilot projects. Additionally, learnings from more than 20 similar pilot projects were collected but are not described and analysed in detail in this study. In Table 6 the selected pilot projects and information of the research mode, the action level, the encountered agile constraints, the underlying motivation to change, introduced digital tools, the dependency level and the project duration are summarized.

Action research mode and hence researcher participation varied among the pilot projects. Participatory Action Research was chosen in pilot projects gamma, eta and theta. The researcher was as an active and productive team member in these pilot projects. To support the team and implement action the researcher incorporated the role of the Scrum Master or of an agile coach. Additionally, the researcher was part of inter team coordination and specific product design tasks. Insider Action Research IAR was chosen in pilot projects beta, delta, iota, zeta. The researcher executed the Scrum Master or agile coach role in these projects and did not take part in specific product design tasks. Action Research without explicit role support of the researcher was chosen in pilot projects alpha, epsilon, kappa and lambda. The researcher did not occupy a specific team role in these projects but consulted the preparation of these projects and the correct implementation of the agile practices. Preparatory works also included team coaching. During the pilot projects the researcher readjusted the agile working model implementation and focused coaching according to the specific situation in the projects. This support lasted from the beginning to the end of the pilot projects.

3.3.1.2 *Data collection*

The data collection focus was structured according to the presented research questions, which resulted in three focus fields. First, benefits and problems of implemented and adopted agile change was recorded. Second, application context characteristics were categorized and connected to experienced problems throughout the pilot projects. Third, successful adjustments to initial change were collected with a focus on constraints of scale and physicality. The documentation and evaluation of implemented change was based on a multi-method data collection approach. The applied method selection was chosen to avoid researcher bias and a methodological influence on data sets. Method triangulation (Blessing and Chakrabarti, 2009) allowed to approach the research phenomena with problem specific methods from different perspectives to increase data relevance and specificity. The set of data collection methods was also adjusted to the project specific availability and accessibility of data sources. The most relevant data collection methods within the pilot projects are sketched in Figure 14.

In every pilot project the initially intended and implemented action was documented. This pilot project preparation documentation included project motivation, objective and intension of introduced change. Additional change and trigger throughout the pilot projects were added to the documentation continuously. Implications of action were recorded with various data collection methods. The central data collection methods were personal notes and direct evaluations of the action researcher. These summaries were continuously expanded throughout the project. Focus was on the working model and not on project specific design progress. Semi-structured interviews (Bogner *et al.*, 2014) were another important data collection method applied in most pilot projects. Project members and management were asked during and after the projects to share their perception of the implemented action and the overall progress of the pilot projects. The interviews were conducted by different researcher to avoid researcher bias. Moreover, pilot project members were asked to summarize their experiences unstructured to address unexpected feedback and complement findings.

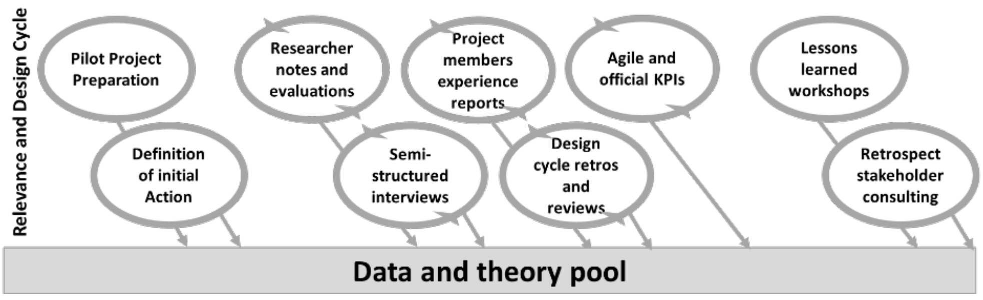

Relevance and Design Cycle
Pilot Project Preparation
Definition of initial Action
Researcher notes and evaluations
Semi-structured interviews
Project members experience reports
Design cycle retros and reviews
Agile and official KPIs
Lessons learned workshops
Retrospect stakeholder consulting
Data and theory pool

Figure 14: Data sources of the Action Research pilot projects. Data from before, throughout and after the pilot projects was collected to provide a representative view on each pilot project. Data collection sources across pilot projects were unaltered to ensure data consistency and comparability.

Besides these direct evaluations methods material from the pilot projects was collected and analysed. Especially retrospective meetings were documented since they directly address the working model and continuously filtrate and document experienced drawbacks and improvements. They were used to get direct feedback for previous change and input for further action within the iterative design cycle frame. Planning and Review meetings and agile practices such as burn-down charts were used to evaluate product design progress and efficiency in retrospect. Project progress in form of performance indicators were used to derive feasibility and efficacy of agile practices. The iterative planning review design cycle was an ideal measurement environment to evaluate the impact of implemented change at a high sensitivity. The ratio between planned and finished talks as well as overall story points allowed to understand impact of action in detail. Most meetings included boards for visualization and information aggregation such as Scrum boards which were documented as well. Besides agile meetings also lessons learned workshops were conducted to evaluate the complete pilot project in retrospect from both internal and external perspectives. These workshops were especially helpful to collect learnings outside of the personal range and visibility of the researcher including tendencies in upper management. Besides these direct information collection methods pilot projects were approached after the official project termination in retrospect. During discussions with the former team members and stakeholders it was investigated whether the implemented impulses were transferred into standard working practices or even further expanded.

Overall data analysis was driven by the following objectives. First, definition and further understanding of the research phenomena agile design and its implications in the defined application context automotive design. Second, elaboration and further sharpening of the research questions. Third, verification and generalization of the research phenomena across pilot projects and outside of the application context. Fourth, construction and evaluation of a suitable design model according to the specific research phenomena in pilot projects. Fifth, design of a comprehensive design model covering pilot projects and supplementary data sources. Derivation of a generalizable design theory answering to the categorized class of design problems.

Data collection and analysis were performed by a several researchers with changing responsibilities to avoid researcher bias from individual researcher. Additionally, standard key performance indicators from the partnering company and stakeholder evaluations of pilot projects were added to the data to further minimize scientific interest-based bias in data evaluation and further increase the relevance of data analysis. Tools such as MAXQDA were used in open, axial and selective coding procedures in pilot projects and especially for interview series.

3.3.2 Structured and narrative literature reviews

The literature review is a central element of Hevner's Rigor Cycle and it addressed several research objectives throughout the project. First, it provided initial understanding of agility and its dependency of the application context. A combination of scientific publications and industry reports was used as an initial theoretical base of the research project. More specifically, literature provided an overview of the maturity and implementation of agility in its original application context and therefore confirmed relevance and urgency of the research project. Practical reports were used to estimate and characterize transferability to other application contexts. Additionally, accompanying challenges in mechatronic product development were summarized. Second, the literature review supported the categorization of emergent design artifacts into the existing body of knowledge. Field reports and theoretical explanation were used to compare, analyse, and explain findings from pilot projects. Third, systematic exploration of theory constructs from design science and complementary fields allowed justified adjustments to the existing theoretical grounding perspectives on the research phenomena. A sketch of the literature review conduct and the researched objectives is given in Figure 15.

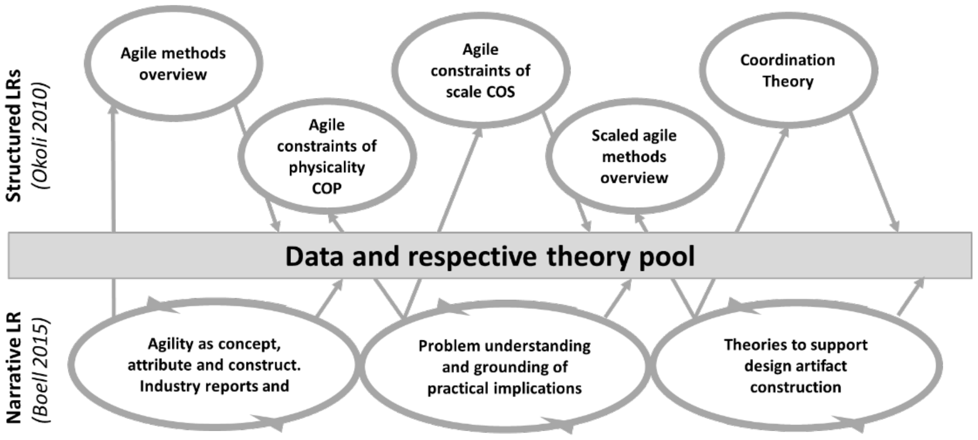

Structured LRs
(Okoli 2010)
Agile methods overview
Agile constraints of physicality COP
Agile constraints of scale COS
Scaled agile methods overview
Coordination Theory
Data and respective theory pool
Narrative LR
(Boell 2015)
Agility as concept, attribute and construct. Industry reports and
Problem understanding and grounding of practical implications
Theories to support design artifact construction

Figure 15: The literature review included of a continuous narrative literature review and several structured literature reviews. The narrative review allowed to maintain an overview of relevant research fields and the structured review was used to thoroughly screen crucial research fields.

Methodologically the literature review was based on two interlaced approaches. Throughout the complete research project a traditional or narrative review was conducted (Boell and Cecez-Kecmanovic, 2015). This iterative review approach allows the researcher to adjust the focus of the search according to earlier findings research results and expand the review to initially not considered fields. This broad and persistent literature review was complemented by specific structured literature reviews SLR (Okoli and Schabram, 2010). These compact reviews were applied several times throughout the complete research project and generated more comprehensive pictures of the selected review objectives. The application of these two approaches into a comprehensive review strategy improved both completeness and thorough understanding of focus fields as well as sensitivity to unobvious but relevant theories and findings and cross-validation of practical findings and theoretical constructs. Combining both review methods had further benefits, since findings of the narrative review were used as reference points for more rigorous structured literature reviews.

This literature review strategy was combined to reflect the quality criteria of literature reviews summarized in Boell and Cecez-Kecmanovic (Boell and Cecez-Kecmanovic, 2015). Comprehensiveness of literature, breadths and depths of understanding signifies that the quality of literature depends on comprehensiveness of and insight into the body of literature researched as well as breadth and depth of its understanding of the researcher (Boote and Beile, 2005). To generate a solid research argument by thoroughly piercing through a body of knowledge and hence localize gaps or flaws is the basic contribution of a literature review to a research project. Feak and Swales declare this aspect of a review as Argument development (Feak and Swales, 2009). Ongoing engagement describes a most vital aspect of narrative literature reviews. From this perspective reviews are an iterative process that continuously further develop understanding and the researcher’s ability to assess relevance and quality of publications. It is not a discrete task but an ongoing process that accompanies a research project up to the final transcript (Combs *et al.*, 2010). Ongoing engagement also improves criticality in reviews which is necessary to question relevance, quality and rigor of contributions and hence filter inputs into research projects (Alvesson and Sandberg, 2011). These criteria enabled unevenly by the two chosen review methods. The structured reviews focused on depth of understanding and criticality in the literature review. The narrative review supported comprehensiveness, argument development and ongoing engagement.

3.3.3 Complementary data collection

Besides data collection in the Action Research pilot projects additional data sources outside of the central application context were used to verify plausibility of the Action Research approach and results. These data sets were collected with the goal to research and confirm the relevance of the research phenomena on a broader scale. Both the generalization of the theoretical constructs behind design phenomena to a class of design problems and the generalization of the design artifact into design theory rely on data to confirm a broader applicability beyond the focus research environment. Apart from this the application of additional research methods minimizes the risk of a disproportionate methodological influence on the research results.

Specifically, the following methods and contexts were used. Outside of the pilot projects but within the specific application context (company BMW) interview series were conducted. Especially in product integration and testing processual dependencies, manual activities, cooperation and coordination systems were analysed. Technological maturity to implement continuous integration tool chains and processual suitability of existing organization structures to implement agile collaboration models were questioned in these interview series. Unlike the more flexible often interrupted and resumed interview discussions in pilot projects these interviews were based on strict semi-structured interview guidelines. The interview sample was selected strictly according to experience in mechatronic product integration and complexity of relevant process partners. Further expert interviews were conducted outside of the application context regarding the transferability of agile practices to mechatronic product design.

An annual industry survey was conducted online to addresses a broad audience of experts in the field of agile mechatronic product design independent of industries (Atzberger, Nicklas, *et al.*, 2020; Schmidt *et al.*, 2019). This survey aimed at several goals unreachable with the earlier presented research methods. First, the survey allowed to reflect the yearly dynamics in a broad set of practical application fields regarding problem interpretation and diffusion of approaches to overcome them. Second, the survey generated data outside of the specific application context that allowed to confirm the relevance of the class of problems. Third, the survey enabled an industry independent evaluation of design artifacts iteratively constructed throughout pilot projects. This approval of experts from other industries is an important precondition to derive general design theories from specific design artifacts. Fourth, this data allowed to collect additional practical implications of the research problem from other industries. Especially if implication change with the application context additional sources help to avoid missing undetected dependencies and therefore increase understanding of the underlying causes. Fifth, the quantitative data analyses could only be used in this part of the research. This alternative data breakdown was used as a repeated evaluation of the research findings and drawbacks of the chosen data collection and analyses methods were compensated.

4 Coordination perspective of agile product design

"The best way to show that a stick is crooked is not to argue about it or to spend time denouncing it, but to lay a straight stick alongside it"
D.L. Moody

In this chapter coordination is used as a theoretical lens to analyse and understand the most relevant agile methods in accordance to avoid unreflective agile black box applications (see chapter SOA scientific call for theoretical explanation) and answer research question one. Therefore, a coordination reference model is synthesized to analyse and compare coordination strategies from different agile frameworks. These coordination strategies are evaluated regarding their suitability for different application contexts according to their coordination efficiency. These evaluations are also used to verify and explain empirically proven agile benefits. Beyond individual practice evaluation the coordination reference model compares interlinked sets of practices of agile methods in relation to different application contexts. This allows to understand agile method suitability regarding project and domain specific characteristics. As a result, the model allows to adjust agile coordination strategies to application contexts.

Similar research on coordination in agile methods has started in the early 2000s and resulted in a heterogeneous research stream including a broad set of approaches. Cao and Ramesh recognized that classical coordination theory might explain the efficiency of agile software development. They argue that agile methods focus on personal and group mode coordination (Lan and Ramesh, 2007). Barlow et al. proposed a methodology selection framework based on standardization, planning and mutual adjustment (Barlow et al., 2011). Strode et al. showed how agile software development generates effective coordination by coordination mechanisms that enable project team synchronization, support team proximity and availability, role substitutability as well as boundary spanning mechanisms (Strode, 2014). Further empiric research was published focusing on individual cases. Coordination in a XP project has been achieved by using unit testing, card games for planning and concurrent versioning systems (Mackenzie and Monk, 2004). Wallboards displaying stories, tasks, work allocations and progress were significant coordination mechanisms in six agile teams (Sharp et al., 2009). Pikkarainen et al. concluded that in two co-located agile projects coordination by documentation was substituted by communication in communities of practice. (Pikkarainen et al., 2008). Pries-Heje et al. explained the efficiency of Scrum by its practices regarding communication, coordination, social integration and control based on a case of two distributed teams (Pries-Heje and Pries-Heje, 2011). Dingsøyr et al. reported how in large scale agile projects coordination modes change throughout the project (Berger and Eklund, 2015) (Dingsøyr et al., 2017). Scherer et al. researched coordination efficiency in multiteam software development projects (Scheerer et al., 2014). Still, coordination theory has not been employed to research and structure large scale hardware design in automotive design.

This chapter is divided into three parts. First, the development of the coordination reference model based on established coordination theory inputs is presented. Second, the derivation of agile coordination strategies of several agile methods including suitable coordination determinants is described. Third, the connection of beneficial characteristics of agile design (depending on application contexts) to the analysed coordination strategies is explained to answer research question one.

4.1 Coordination reference model

The analysis and comparison of coordination strategies of agile working models requires a coordination reference model. This model needs to screen agile practices regarding their impact on coordination efficiency. This includes intra and inter team coordination. Besides individual practices it must evaluate coordination systems that are composed of several interlinked coordination mechanisms. Furthermore, these interdependent coordination systems must be related to coordination determinants which represent application context characteristics.

The applied coordination reference model was specifically designed to reflect agile product design. It is based on the coordination model of Van de Ven et al. (Ven *et al.*, 1976) which allows to relate the suitability of coordination modes to specific coordination contexts which are represented by coordination determinants. The model connects impersonal, group and individual mode coordination to task uncertainty, task interdependence and size of work unit. Additionally, the concepts of boundary spanning (Star and Griesemer, 1989), and implicit cognitive coordination (Espinosa *et al.*, 2004) were introduced to reflect the strong team focus of agile product design. The coordination reference model allows to deduce coordination strategies (Li and Maedche, 2012) from different agile methods. It differentiates between intra team and inter team coordination to extend its applicability to scaled application contexts. The complete model as depicted in Figure 16 is described in the following paragraphs.

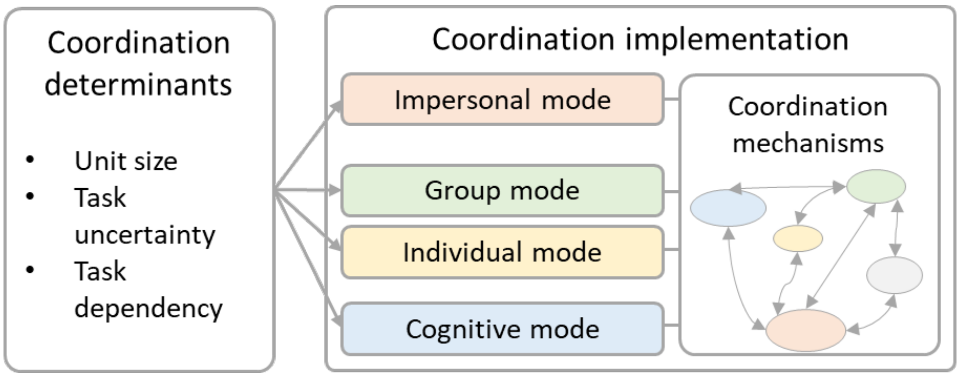


Figure 16: The coordination reference model. It allows to analyse coordination strategies of agile methods. It is based on the original model of Van de Ven et al. (Ven *et al.*, 1976)*. It connects the coordination determinants unit size, task uncertainty and task dependency to the coordination modes impersonal mode, group mode and individual mode coordination. To better reflect agile design the model was extended to include cognitive mode coordination* (Espinosa *et al.*, 2004) *and boundary spanning* (Star and Griesemer, 1989)*.*

Like in the original model (Ven *et al.*, 1976), the differentiation of impersonal and mutual adjustment coordination relies on the three separated **coordination modes**: impersonal mode coordination, group mode coordination and individual mode coordination. These coordination modes consist of sets of coordination mechanisms based on similar coordination principles. **Impersonal mode coordination** includes coordination mechanisms such as “pre-established plans, schedules, forecasts, formalized tools, policies and procedures, and standardized information and communication systems” (Ven *et al.*, 1976). These mechanisms present codified blueprints of action that are impersonally specified. Roles are formally prescribed in impersonal standardized blueprints or action programs which avoids role articulation (Thompson, 1967) to the given set of tasks. The application of the impersonal coordination mechanisms requires minimal mutual adjustment such as verbal exchange between the task performers after their implementation. Coordination by feedback is based on mutual adjustment upon new information (Thompson, 1967). De Van et al. divide it into group mode and individual mode. Regarding **group mode coordination** the model follows the division of Hage et al. into scheduled and unscheduled meetings as coordination mechanisms (Hage *et al.*, 1971). Scheduled meetings serve for routine and plannable routine meetings like stuff meetings. Unscheduled meetings represent unplanned communication like informal, impromptu exchanges between more than two individuals in work-related fields. This also covers spontaneous collaboration on design activities in groups. Individual role occupants mutually adjust their tasks based on either vertical or horizontal communication channels in **individual mode coordination**. Typical coordination mechanisms for vertical communication would be line manager and unit supervisor. Horizontal exchange is based on direct communication between individuals on a one-to-one basis in non-hierarchical relations.

Impersonal mode, group mode and individual mode coordination are explicit coordination modes. Their coordination mechanisms require action to obtain a state of coordination. Unlike explicit coordination implicit coordination is based on shared information and cognition and requires no coordination-specific action. Research on teamwork has shown that implicit, **cognitive coordination** mechanisms have a significant influence on overall coordination efficiency (Espinosa *et al.*, 2004). Agile approaches emphasise close collaboration in small, long-term teams which are ideal prerequisites for the application of cognitive coordination. Therefore, this study includes cognitive mode coordination as a fourth coordination mode to the coordination reference model (see Figure 16) to better reflect agile coordination strategies. Espinosa et al. presented a similar adjustment of the Van de Ven model (Espinosa *et al.*, 2010). Cognitive mode coordination is based on knowledge collaborating actors or parties have of the system, each other and of each other's tasks. It allows them to anticipate each other's action without explicit coordination effort. Task awareness, presence awareness, transactive memory (Wegner, 1995) and expertise coordination (Faraj and Sproull, 2000) are factors that influence cognitive coordination. Shared mental models in teams (Cannon-Bowers *et al.*, 1993) are essential since they support shared goals and enable common understanding (Kang *et al.*, 2006). Mutual knowledge (mutual knowledge of collaborators) and common grounding (similar meaning in terms) are further enabling factors to cognitive coordination. In contrast to impersonal and personal coordination modes cognitive coordination requires shared mental models, mutual knowledge and trust between actors and hence cannot be implemented by generic coordination mechanisms. Since it is based on personal exchange, individual knowledge, trust which require long-term cooperation its application is mostly limited to intra team coordination.

The coordination reference model also includes the concept of **boundary spanning** as coordination mechanism. They represent relevant agile practices such as prototyping or agile artefacts such as the backlog in the model. The individual implementations of boundary objects, boundary roles and boundary activities fit the impersonal, group and personal coordination modes and combine their characteristics. As coordination mechanisms they connect the coordination modes of the coordination reference model. The boundary concept is used to facilitate coordination between representatives of different design objectives with different backgrounds and fields of interest. Star et al. define boundary objects as artefacts or concepts with enough structure to support activities in separated social worlds, and with enough elasticity to cut across multiple social worlds (Star and Griesemer, 1989). To do so these objects satisfy the institutional requirements of each world. They are weakly structured in common use and strongly structured in local use (Star and Griesemer, 1989). This allows them to tie together actors in multiple social worlds, while being capable of assuming different meanings in each world (Briers and Wai, 2001). Bergman et al. define boundary objects in design as artefacts that enable, propel, and connect design routines, align stakeholder interests, and identify and reduce gaps in design knowledge (Bergman *et al.*, 2007). They represent, transform, mobilize, and legitimize design knowledge by facilitating shared understanding and promoting agreements about designs. Bergman et al. define four essential features of design boundary objects (Bergman *et al.*, 2007). First, they promote shared representations. Second, they transform design knowledge. Third, they mobilize for design action. Fourth, they legitimize design knowledge. Boundary spanner and boundary spanning activities are further extensions of the concept. For example a coordinator between different knowledge and interest group might serve as a boundary role (Levina and Vaast, 2005). Certain activities between these stakeholders have a similar affect and serve as boundary spanning activity. Especially in design environments that rely on a broad spectrum of specializations and need to connect many stakeholders, boundary objects can increase coordination efficiency significantly.

Van de Ven et al.'s model includes the **coordination determinants** task uncertainty, task interdependence and size of work unit that influence coordination mode selection. The combination of these factors allows to reconstruct different application contexts based on comparable input parameters. **Task uncertainty** mirrors the difficulty and variability of the activities necessary to complete a task. This includes project characteristics such as hardware and software specific design characteristics. It is reflected by the complexity level of the task, the necessary thinking time and task predictability. **Task interdependence** describes how task responsible parties are dependent on the output of other organization units to complete their task. This includes interdependencies between activities, knowledge and roles (Pennings, 1975). High task interdependency reduces the share of one-person jobs and increases the necessary degree of collaboration to complete tasks. Task interdependence also differentiated between intra team and inter team dependencies. **Size of work unit** is defined as the total number of people within an interdependent project.

Van de Ven et al. showed how these coordination determinants influence the presented coordination modes. Increasing task uncertainty results in a substitution of impersonal mode coordination with mutual adjustment coordination. Within the group mode coordination unscheduled meetings show a stronger increase than scheduled meetings. Regarding individual mode coordination the use of horizontal channels increases while vertical channels remain constant. Increasing task interdependency results in an increased use of group mode coordination while impersonal mode coordination and individual mode coordination remain invariant. Within the group mode coordination scheduled meetings increase most. Increasing size of work unit leads to more impersonal mode coordination. This affects mostly rules and plans. Dietrich et al. researched the same coordination determinants in multiteam systems and came to very similar connections (Dietrich *et al.*, 2013). Regarding the cognitive coordination mode unit size and inter team task dependencies reduce its applicability most since personal relations and shared mental models are harder to establish in larger groups and across teams.

The coordination reference model determines mutual influence between coordination modes and coordination determinants to account for the criticism of Jarzabkowski and Okhuysen (Jarzabkowski *et al.*, 2012; Okhuysen and Bechky, 2009) that the original Van de Ven et al. model (Ven *et al.*, 1976) only reflects a static view of coordination. Connections between coordination mechanisms allow to depict emergent coordination dynamics that reflect changing coordination determinants. These connections between coordination mechanisms and the analyses of mutually influential coordination modes prevent a static understanding of coordination in the coordination reference model.

The mutual influences between coordination modes are analysed as well. The connection of coordination modes to coordination determinants, based on Van de Ven et al. original observations, allows to match coordination strategies of the researched agile working models to application contexts. The coordination reference model also accounts for the cost and sensitivity of its coordination modes. Cost of coordination is approximated by the combined time necessary to execute coordination activities for all stakeholders. Impersonal and cognitive coordination modes require little time, while group mode and individual mode coordination require much more time. Besides cost the model also considers coordination mode sensitivity. Group and individual mode coordination are able to cope with higher levels of complexity than impersonal mode coordination. The coordination reference model connects this task sensitivity with the presented coordination determinants. Requirements of coordination modes are considered as well. E.g. cognitive mode coordination requires long-term teams that can rely on individuals that know and trust each other and that have experience in the required tasks and know project goals and settings.

In summary, the **coordination reference model** is based the original model of Van de Ven et al. (Ven *et al.*, 1976). The concepts of cognitive mode coordination and boundary objects were added to better reflect coordination in agile frameworks.

4.2 Agile coordination strategies

The aim of this subchapter is to deduce the coordination strategies of the agile methods Scrum and eXtreme Programming. The analyses are based on the presented coordination reference model. The agile methods were selected for several reasons. They are the most popular agile methods in practical application (VersionOne, 2020). Both methods rely on large numbers of industrial experience reports and scientific evaluations. Later methods are often based on them. They represent two opposing positions within agile methods regarding domain limitations. Extreme Programming has been established for software design and therefore includes many domain specific practices, while Scrum is not limited to the domain. They include most of the employed agile practices in the agile pilot projects.

The analyses of the coordination strategies for both methods. consist of three steps. First, the employed coordination modes and mechanisms are defined. Second, the balance of the coordination system is analysed. Connections and trigger between coordination modes and mechanisms are mapped to explain the system behaviour. Third, the agile coordination strategy is summarized. Characteristics and behaviour of the coordination strategy are matched with empirically proven benefits of agile methods.

4.2.1 Analysis Scrum coordination strategy

The coordination strategy of **Scrum** (see subchapter 2.1.2.2) is analysed in the following section. Scrum emphasises intra team coordination mechanisms. It combines three categories of coordination mechanisms which are meetings, roles and artefacts. It includes seven different **meetings**: The Planning (part I and II), the Daily, the Review, the Retro, the Refinement and the Sprint which structure iterative development cycles. Three **roles** with specific tasks and responsibilities are defined: The Product Owner, the Scrum Master and the Development Team. Additionally, three **artefacts** the Backlog, the Sprint Backlog and the Increment are defined. The Scrum Guide (Schwaber and Sutherland, 2020) advises small, long-term, cross-function, co-located and collaborating teams. The framework specifically recommends minimal inter team (team external) dependencies and emphasizes intra team collaboration. It advises short iterative design cycles with a fixed length which are called **Sprints**.

These basic elements and their connection within the framework are analysed in the following section to depict the Scrum coordination strategy. The coordination reference model is used as a theoretical lens to analyse the Scrum elements regarding their influence on its coordination efficiency. The analysis is structured into three steps. First, the resulting coordination modes are investigated. Second, for every coordination mode the connected mechanisms are summarized. Third, the connections between coordination modes and respective coordination mechanisms are outlined.

4.2.1.1 Applied coordination modes

The **Scrum meetings** enable **group mode coordination** in the Scrum framework. Besides the Sprint and the Refinement all meetings can be categorized as scheduled group mode coordination. They are structured according to design objectives and therefore connect roles that also answer to different design objectives. The Sprint on the other hand offers the opportunity for unscheduled meetings. This includes informal exchange and or spontaneous design activities within the team. Unscheduled periods are purposely included in the Sprint to provide unscheduled exchange. Besides this group mode coordination, the meeting also support **cognitive mode coordination** within the team. The Planning meetings provides a shared vision and common understanding throughout the following Sprint, while the Review meeting generated mutual trust within the team and towards the framework. The Retro meeting provides information of team dynamics and therefore also supports cognitive coordination.

The **Scrum roles** support **impersonal mode coordination** as standardized blueprints of individual responsibilities (Thompson, 1967) in product design. They also support **individual mode coordination**. Since no disciplinary hierarchy is associated with the role responsibility horizontal communication channels are dominant. The Scrum roles are directly linked to the **scheduled group mode coordination** mechanisms. The role definitions clarify responsibility within the team and towards all meetings and therefore facilitate group mode coordination. The Scrum Master and the Product Owner are **boundary spanners** and connect the Development Team to customers and stakeholders. Lastly, the Scrum roles also support **cognitive mode coordination** since responsibility and functions are clearly assigned to individuals which allows a shared understanding of the overall design activities.

The **Scrum artefacts** provide **impersonal mode coordination**. The Backlog and the Sprint Backlog allow coordination between stakeholders without personal communication outside of the meetings. They fulfil Bergman's four categories of boundary objects in design. The Backlog promotes a shared representation of the product, the Increment includes transformed design knowledge from different parties, the Backlog and the Increment mobilize for action and the Increment legitimizes design knowledge. Additionally, they document findings and results from meetings and the Sprint. The integration of the artefacts into the Scrum design cycle ensures they document and communicate the status quo. The artefacts are designed to function as **boundary object** coordination mechanisms between the Scrum roles, the customer, and stakeholders. This function facilitates the meeting efficiency since information can be transferred between different roles even though different backgrounds and objectives are given. The User Stories within the Backlog are an example to connect Product Owner and Development Team. Additionally, the increment or prototypes are boundary objects in Scrum.

The required **team characteristics** from the Scrum Guide shape coordination in a Scrum framework as well. **Group mode coordination** is enabled by the small size of the teams, the collaborative approach to design tasks, the co-location and the cross-functional composition of the team. Each aspect ensures that group mode coordination is the most usable and useful coordination mode. The same aspects also support individual mode coordination but to a lesser degree. In the long term these aspects generate very close teams that can rely on **cognitive mode coordination**. Another factor to influence coordination mode selection is the handling of **external or inter team dependencies**. The role construct attaches external dependencies to the Product Owner and to a smaller degree to the Scrum Master. The Development Team can therefore focus on design tasks and hence rely on **group mode or individual mode coordination** within the team. This use of mutual adjustment coordination relies on a small number of external dependencies that can be channelled by individual role occupants.

In summary **impersonal mode coordination** is implemented in the Scrum framework through the Backlog, the Sprint Backlog and the Scrum role definitions. Additional elements such as User Stories also provide impersonal mode coordination. All Scrum artefacts are implemented as boundary objects to be easily adjustable and maintain a high level of flexibility and actuality. Besides these objects the Scrum framework its guidelines and standards function as impersonal coordination mechanisms as well. **Group mode coordination** is the central coordination mode in Scrum. The meetings, the roles and the team characteristics function as group mode coordination mechanisms. Even though scheduled meetings structure a large share of the design cycle unscheduled meetings are driven by the Sprint frame and the close team collaboration. Group mode coordination mechanisms connect all roles and artefacts and therefore present the central coordination mechanisms. The different roles and their non-disciplinary relations enable **individual mode coordination** mechanisms through horizontal channels. Close collaboration also requires a shift between group and individual mode coordination depending on the task. **Cognitive mode coordination** is enabled through the close personal exchange during the scheduled meetings that provide a shared vision, a common understanding and generate trust among the teams. The role division clarifies responsibilities and simplifies group dynamics. Required team characteristics such as long-term teams cultivate cognitive coordination mechanisms. Additionally, the simple overall framework provides common grounding in terminology which also support cognitive mode coordination. Collaboration, co-location and cross-functionality all require teams to cooperate very close and hence grow together. Lastly, the focus on intra team cooperation and the separation from external dependencies avoid inter team dependencies that also increase unit size and complicate cognitive coordination mechanisms.

4.2.1.2 Connection and balance between coordination modes

After analysing the applied coordination modes in Scrum the following section discusses how these coordination modes are connected in the framework. Figure 17 gives an overview of how the coordination modes influence each other. The construct helps to understand the overall coordination strategy besides the individual coordination implementations.

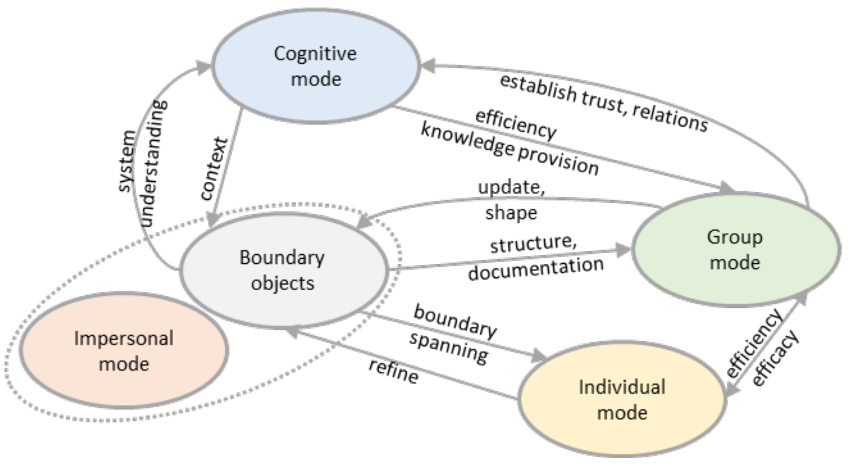


Figure 17: Scrum coordination strategy. Group, cognitive and impersonal (including boundary spanning) mode coordination are central. The coordination system's flexibility, efficiency and efficacy profit from the close interlinkage of the employed coordination modes.

Impersonal mode coordination mechanisms like the Backlog, the Sprint Backlog and the Increment are directly linked to group mode coordination mechanisms. The Sprint Backlog structures the Sprint, the Daily and the Review meetings, while the Backlog structures the Planning meeting. They are input channels for tasks and document and distribute findings from the verbal meetings replacing additional protocols. Their structure as boundary objects facilitates cooperation between different roles and stakeholders throughout scheduled and unscheduled meetings. This also affects individual mode coordination in the Backlog Refinement. These objects also enable cognitive coordination since they provide transparency and efficiently channel information in projects and hence facilitate system understanding. The Scrum framework itself and its roles are also impersonal coordination mechanisms which supports cognitive mode coordination in the design teams since they structure complex product design systems into well-described roles, meetings and artefacts whose connections and linkages are transparently stated in the framework rules.

Group mode coordination mechanisms on the other hand also have a strong influence on the boundary object artefacts. Most prominently the Planning meeting is the central input channel for the Sprint Backlog, while the Review meeting verifies and validates the Increment of the Sprint. This shows that there is a strong bidirectional linkage between the impersonal mode coordination boundary objects and group mode coordination scheduled Scrum meetings. The meetings also provide the basis for further individual mode coordination that might be more efficient than a group meeting. E.g. the Daily meeting only detects coordination demand to be solved either in other meetings or in individual mode coordination. Also, the Sprint encourages direct exchange between individuals in collaborative tasks. Indirectly the meetings also provide a base for cognitive mode coordination through repeated interaction between team members, shared mental models of the system, transparency of responsibilities and trust in task fulfilment.

Individual mode coordination might trigger unscheduled group mode meetings or influence boundary objects. E.g. the adaption of the Backlog during a Refinement of the Product Owner with customers. **Cognitive mode coordination** supports the other coordination mechanisms since a large share of the necessary information is already distributed and does not need to be communicated explicitly. E.g. a personal exchange or a meeting can focus on the point, or a boundary object can be simplified and still be understood. In very dynamic and uncertain environments this indirect extension of the given information might also lead to misinformation if the established mutual knowledge is outdated. Therefore, the shared knowledge needs reference frames and actualizations from the other coordination modes.

In summary, the Scrum framework relies on interlinked coordination modes that are designed to support and complement each other. Its compact design cycle applies and combines different coordination modes to guide the design progress. The connection between these coordination modes is not fixed but able to adapt to specific or emergent project requirements.

4.2.1.3 Deduced Scrum coordination strategy

The presented analysis shows that the Scrum coordination strategy is **dominated by mutual adjustment coordination** especially in the shape of group mode coordination. Individual mode coordination is also relevant, but to a lesser degree. **Impersonal mode coordination** in the form of boundary objects and boundary spanner roles is interlinked with group mode coordination. Additionally, the framework and its elements are designed to generate cognitive mode coordination mechanisms. According to Van de Ven's relation between coordination modes and coordination determinants this combination of coordination modes is suitable for small unit sizes, high task uncertainty and high intra team and low inter team task dependencies. The focus on cognitive mode coordination further emphasises this application focus.

The presented combination of coordination modes is designed to provide project **flexibility and speed** which are necessary in dynamic application contexts characterized by high task uncertainty. It combines coordination efficiency and efficacy since coordination mechanisms are continuously adapted to the demand. Individual coordination modes and mechanisms are intensified or reduced according to demand. Efficient impersonal mode coordination is used in predictable situations and more flexible mutual adjustment coordination modes are applied in uncertain situation. This adjustment mechanism answers to short term changes and long-term tendencies in projects. Task dependencies are described only from an intra team perspective. Inter team dependencies are not included explicitly.

The strong **connection between the employed coordination modes** supports flexibility. The Scrum framework provides a well-coordinated system, that according to Espinosa (Espinosa et al., 2004), relies on an effective mix of mechanisms for the coordination needs of the task. Different modes are matched according to their strengths and weaknesses to generate a mutually supporting coordination system. Group mode coordination and impersonal mode coordination are interlinked very close. Impersonal mode coordination artefacts are updated throughout scheduled group mode meetings and scheduled meetings are structured according to boundary objects. Scheduled group meetings and impersonal coordination mode facilitate mutual knowledge, trust and shared vision which support implicit coordination. This system supports fast knowledge exchange, generation and documentation.

These results match the findings of Pries-Heje et al. (Pries-Heje and Pries-Heje, 2011). They explain the success of Scrum with social capital, which is the ability to build trust, motivate and build relations between individuals. All concepts that reflect the strong cognitive and group mode coordination in the presented coordination analysis. It also provides a common language and a shared target to aim for. This social capital concept is close to the cognitive coordination concept. They also emphasise the importance of boundary objects and impersonal mechanisms to ensure quality and track progress in Scrum. They argument that the meeting structure enables efficient communication. Lastly, they draw a similar conclusion saying that the framework suits small teams in dynamic application contexts.

4.2.2 Analysis eXtreme Programming XP coordination strategy

The coordination strategy of the agile method **eXtreme Programming** XP (Beck, 2004) is analysed in the following section. XP was chosen for four reasons. First, it is almost as popular as Scrum (VersionOne, 2020). Second, its coordination relies unlike Scrum on highly automated coordination mechanisms that require specific IT infrastructure. Third, some practices exclusively address software design. Fourth, the framework emphasises product design practices and not project management like Scrum. XP consists of values, principles and practices. The underlying **values** are communication, simplicity, feedback, courage and respect. The basic **principles** are feedback, assuming simplicity and embracing change. More relevant for the coordination strategy are the XP **practices**: On-Site Customer, Planning Game, Stand-Up meetings, Metaphor, Short Releases, Retrospective, Testing, Simple Design, Refactoring, Pair Programming, Collective Ownership, Continuous Integration, Coding Standards and Sustainable Pace. Like Scrum XP recommends short iterations based on coding, testing, listening and designing development cycles. These basic elements and their connection within the framework are analysed to understand the resulting coordination strategy.

The coordination reference model is used as a theoretical lens to analyse the XP practices regarding their impact on coordination. The analysis is structured into three steps. First, the resulting coordination modes are investigated. Second, for every coordination mode the connected elements are summarized. Third, the connections between coordination modes and respective framework elements are outlined.

4.2.2.1 Applied coordination modes

The coordination modes in XP are differentiated according to the presented design practices. **On-Site Customer** representation triggers individual mode coordination or group mode coordination between designer and customer both in personal and group mode exchange. The **Planning Game** functions as a boundary spanning activity allowing different social or expertise communities to interact directly without complicate translation activities. Its implementation as a group activity turns it into a (scheduled) group mode coordination mechanism. The **Stand-Up** meetings are classic mechanisms of the group mode coordination. Similar to the Planning Game the **Metaphor** connects different individuals from different task specific focus areas. Its construction is a boundary spanning mechanism while the artefact remains a boundary object. The **Retrospective** meeting provides frequent reconsideration and adjustment of the process and is a scheduled group mode coordination mechanism. The **Simple Design** practice represents the idea to choose the simplest possible design to fulfil requirements. It allows designer to comprehend outputs of other designer and is therefore categorized as a cognitive mode coordination mechanism. **Refactoring** includes the continuous code improvement and simplification which implicates that initial code needn't be perfect. It supports impersonal and cognitive mode coordination mechanisms. The resulting design procedure requires and enables continuous product adjustments and therefore requires repeated synchronisation points. **Pair programming** is the most relevant individual mode coordination mechanism and relies on close personal exchange between two designers. **Short Releases** and the corresponding short design cycles as well as the incremental design support impersonal mode coordination. The **Continuous Integration** practice is one of the central impersonal mode coordination mechanisms. It enables a continuous actualization of the current increment and hence allows feedback and evaluation cycles. **Coding standards** as a form of team agreed rules are an impersonal mode coordination mechanism. The last practice, (automated) **Testing** is another very relevant impersonal mode coordination mechanism. Developing tests first and then starting the classic design process reverses to original sequence of design cycles and enables a very efficient design process. This includes mostly verification but also validation tests. Mistakes are spotted fast and further necessary coordination mechanisms to generate solutions can be triggered. Like the Simple Design practice once implemented the Testing practice functions like the implicit cognitive mode coordination mechanisms, since the system automatically detects mistakes and generates coordination trigger.

In summary, **impersonal mode coordination** is central to the XP framework. The testing and continuous integration mechanisms are essential to this coordination mode. The Continuous Integration practice enables transparent product representation throughout the design process without integration delays. The Continuous Testing practice on the other hand simplifies the original Demming design cycle (Moen and Norman, 2009). Instead of an independent Check phase the Testing practice is automated and connected to the Do phase which increases design efficiency significantly since further design on unnoticed errors is avoided. This efficiency gain is also due to the amount of saved personal interaction that would usually be necessary during the Check phase to find and fix errors. The practice benefits intra team but also inter team and team-customer coordination. These benefits of the Testing practice rely on the Continuous Integration practice to avoid tests of incomplete products and actuality gaps (Schrof and Paetzold, 2020). Further supporting coordination mechanisms are the Coding Standard practice and the Simple Design practice that allow to efficiently implement tests and aid error fixing. The Short Releases practice also supports the impersonal mode coordination since it is a design rule that enables short design cycles and provides direct feedback from the customer in short intervals. These impersonal mode coordination mechanisms avoid non-value design works regarding both costumer requirements and design implementation.

Even though, XP relies on strong impersonal mode coordination, **individual mode coordination** is reflected through several XP practices as well. The On-Site customer allows direct personal exchange between designer and customer. The metaphor as a boundary spanning (non-physical) object requires direct exchange and further connects them. The most important individual mode coordination mechanism in XP is the Pair Programming practice. It provides a very close personal relation between two individuals.

Group mode coordination is also part of the XP coordination construct even though only two scheduled meetings are described in the framework. The Retrospective and Stand-Up meeting coordination mechanisms provide non-automatized mutual adjustment coordination in teams. The Retrospective focuses on the overall, long-term design process while the Stand-Up meeting has a short-term focus. The Planning Game coordination mechanism connects designers with customer requirements and customer expectations. The central XP values and the Collective Ownership practice emphasis team collaboration and therefore group mode coordination.

Boundary spanning coordination mechanisms are the Metaphor and the Planning Game practices. They connect groups with different design objectives and enable an efficient and uncomplicated information exchange. **Cognitive mode coordination** is a result of the scheduled and unscheduled personal meetings coordination mechanisms in the form of a common understanding and trust among the team. The On-Site Customer practice supports a shared product vision and the Planning Game practice allows to realistically transform this vision into increments. The Simple Design practice generates traceability and product understanding for teams. Personal knowledge and trust among the team is also created by the Pair Programming coordination mechanism.

The Sustainable Pace coordination mechanism emphasises compact, iterative design cycles that ensure entanglement of coordination modes. Additionally, the straightforward XP framework provides common grounding in terminology. It also limits project boundaries which increases system understanding and hence cognitive coordination. A large share is of dependencies is automatically handled by the testing and integration system which allows individuals to focus on components without understanding the complete system.

4.2.2.2 Connection and balance between coordination modes

Impersonal mode coordination mechanisms such as the Testing and Continuous Integration practices have a direct influence on the implementation of group and individual mode coordination mechanisms as shown in Figure 18. The continuous integration and testing systems automatically detect design mistakes and problems that require designers’ attention. This automated analysis allows to employ group and individual mode coordination only if necessary. Efficient automated, impersonal mechanisms are used for repetitive, standard tasks and personal mechanisms are triggered to handle complex problems or unpredictable tasks only if necessary. This approach also provides faster design since these impersonal coordination mode mechanisms are quicker than personal coordination modes. Meeting efficiency increases if only relevant topics are addressed. This connection between impersonal mode coordination and individual mode coordination applies to group mode and individual mode coordination mechanisms. The automated integration and testing infrastructure is linked to cognitive coordination mechanisms as well. The implemented system provides a similar service since information distribution results in coordination without explicit effort. This implicit coordination is not based on cognitive models but on digital data processing in a connected product design IT infrastructure. Therefore, automated testing and integration practices replace some aspects of cognitive mode coordination while providing a similar service. Lastly, the XP framework itself is an impersonal mode coordination that structures and connects impersonal and mutual adjustment coordination modes.

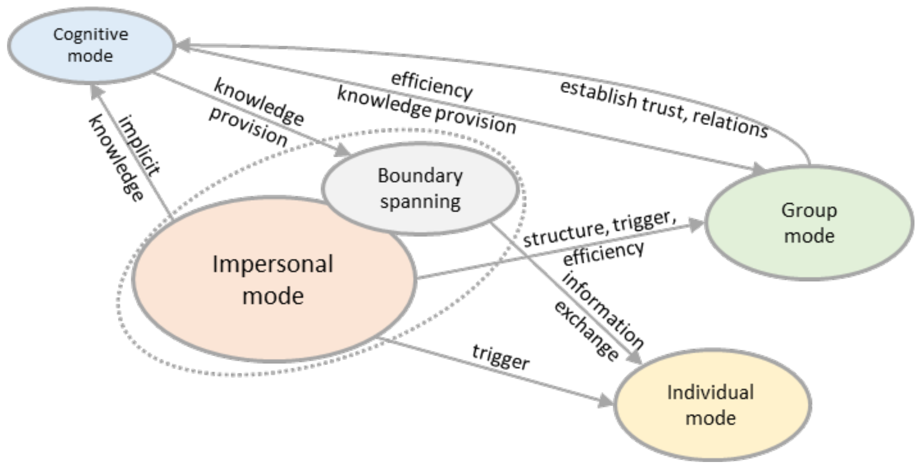


Figure 18: XP coordination strategy. Automated impersonal mode coordination mechanisms represent the foundation of the XP coordination system. Group, individual and cognitive mode coordination are also present. The coordination modes are well connected into a coordination system and the connections between them improve coordination efficiency and efficacy.

Personal and group mode coordination are linked to other coordination modes. Pair Programming generates cognitive coordination in teams. The Retrospective reviews the complete product design process including impersonal and mutual adjustment coordination mechanisms. This group mode coordination mechanism adjusts other coordination modes according to feedback from the team. The Daily Stand-Up summarizes problems and results at team level on a daily rhythm. It allows direct feedback and reaction and is a straight connection to other personal and impersonal coordination modes. Both personal and group mode coordination mechanisms also support cognitive coordination mode since they require personal exchange on product design relevant tasks.

Boundary Spanning practices such as the Planning game connect individuals and facilitate information exchange. The Metaphor practice functions similarly. Both improve personal exchange on group and individual mode coordination. Additionally, they expand system understanding and hence cognitive mode coordination in the long-term. **Cognitive mode coordination** on the other hand improves group mode coordination and individual mode coordination as well as boundary spanning since the amount of necessary information exchange is reduced and the remaining information exchange is handled more precisely.

In summary, impersonal and individual mode coordination are connected very close. Impersonal mechanisms trigger and structure personal mechanisms. Personal mechanisms replace impersonal mechanisms if complex problems need to be addressed. This connection between coordination mechanisms is designed to be flexible to adapt to changing project characteristics and project dynamics. Unlike Scrum XP only applies to software products and requires elaborate IT system support in the form of testing algorithms and corresponding IT infrastructure.

4.2.2.3 Deduced XP coordination strategy

The presented analysis shows that the XP coordination strategy emphasizes **impersonal mode coordination**, but **personal and group mode coordination** mechanisms are relevant as well. Boundary spanning activities and objects are a third pillar in the XP coordination strategy. **Cognitive mode coordination** is enabled to a smaller degree through the framework's elements. The impersonal mode coordination mechanisms provide a similar service that does not rely on implicit personal but system knowledge in IT systems. In a nutshell, XP establishes an interlinked coordination system. According to Van de Ven's relation between coordination modes and coordination determinants this combination of coordination modes is suitable for large team sizes, medium task uncertainty, high intra team and medium inter team dependencies. Larger units benefit from the implicit impersonal mode coordination mechanisms which provide a similar function like cognitive mode coordination does in smaller teams. Compared to the Scrum coordination strategy impersonal mode coordination mechanisms are more relevant while cognitive and group mode coordination are less relevant but still employed. The absence of roles and instead the focus on practices is the result of this shift. Still, examples like the Pair Programming practice underline individual mode coordination relevance. Boundary spanning focuses on activities and less on roles and objects. Quality and progress control are implemented in impersonal mode mechanisms, while Scrum relied here on group mode mechanisms.

The XP coordination strategy results in **coordination efficiency, speed and flexibility** which are necessary in new product design environments. Unlike Scrum the XP coordination strategy requires well-adjusted Continuous Integration and Testing IT systems. The corresponding costs must be balanced with long-term reductions in coordination efforts. Impersonal coordination mechanisms provide high efficiency levels for standardized and predictable tasks. More sensitive and costly coordination mechanisms are applied only if necessary. Automated impersonal mechanisms increase coordination speed for most coordination activities. Still, the XP strategy can switch between impersonal and individual mode coordination, which results in coordination flexibility. Adjustments of coordination mechanisms require little time once the necessary IT infrastructure is established. Compared to Scrum the XP coordination flexibility is quicker regarding small changes but slower on a large scale if adjustments to the IT infrastructure are necessary. Additionally, XP is applicable to larger units. The impersonal mode coordination mechanisms provide inter team and intra team coordination while personal, mutual adjustment mechanisms are more suitable in intra team coordination.

The selection of XP design practices shows a strong **connection between coordination modes**. The Impersonal coordination mechanisms such as Coding Standards, Refactoring, Continuous integration, and Testing are adjusted mutually and personal coordination modes are connected to them as well. The balance between impersonal and personal mechanisms provides efficiency for standard design works and flexibility for unexpected tasks. Automatization of impersonal coordination mechanisms also increases the coordination coverage of relevant activities which would be inefficient with other coordination modes. This allows to improve sensitivity for personal coordination modes that require more coordination effort. The connection between coordination modes is triggered mostly from impersonal mechanisms. Still, mutual adjustment practices like the Planning Game or the Daily Stand-up present the opposite direction. Boundary spanning activities like the Metaphor practice structure group mode coordination meetings and facilitate individual mode coordination mechanisms such as Pair Programming. Since the integration and testing infrastructure supports cognitive coordination less detailed information is necessary to generate system comprehension. Implicit coordination is therefore based on both digital system representation and personal knowledge of individuals.

Unlike Scrum XP requires **complex IT infrastructure** to provide the presented impersonal coordination mechanisms. Especially, the Continuous Integration and Testing practices rely on dedicated infrastructure and software toolchains which are a significant cost factor. Besides, such infrastructure is very product specific. While the technology is a standard in software design, the same is not true for most hardware design systems. Immature technology in hardware design might outbalance the expected benefits. Additionally, designers require competencies regarding test-driven design and integration infrastructure deployment.

The deduced XP coordination strategy matches earlier findings. XU et al. analysed coordination in large agile systems through impersonal practices such as Coding Standards and personal practices like On-Site Customers in XP in 2009 (Xu, 2009). Strode et al. looked into the coordination of co-located agile team through Shared Code Ownership in 2012 (Strode *et al.*, 2012).

4.2.3 Findings in response to research question one

In a review about the last decade of agile development in 2012 Dingsøyr et al. stated that the theory behind agile development is multifarious and a holistic explanation why agile works does not exist (Dingsøyr *et al.*, 2012). The presented coordination analysis of agile methods shows that the coordination perspective improves understanding of why agile design works. Coordination theory cannot provide the holistic understanding Dingsøyr et al. asked for, but it reflects central traits of agility in design. The four core concepts of agility (Baham and Hirschheim, 2021) help to elaborate the relation between agility in design and coordination theory. Close collaboration in teams is based on constant coordination within design teams. Continuous costumer involvement is based on repeated inter team coordination between designers and costumers. Inspect and adapt cycles rely on reflection within design teams to inspect and adapt. Iterative and incremental design are based on repeated analysis and replanning and hence require intensive intra and inter team coordination. All core concepts directly involve coordination between and within groups. The focus on personal exchange and communication in agile design practices underlines the relevance of coordination further. Coordination theory therefore enables to analyse agility in design and hence to answer research question one: How to explain agility and its benefits theoretically. Coordination theory only represents one central element to a holistic explanation of agile design. The theoretical explanations based on coordination theory may serve as building blocks for further applicable design theories.

The coordination strategy concept was chosen to apply coordination theory for the analysis of agile design. It connects coordination determinants to suitable coordination modes and respective coordination mechanisms within a coordination system. It also allows to relate coordination system behaviour with attributed benefits of agile design. To derive comparable coordination strategies from different agile methods a coordination reference model was necessary. The employed model was constructed to connect relevant concepts of the broad field coordination theory that best reflect agile design structures such as coordination between and within teams. The coordination reference model allows to categorize individual agile elements. Furthermore, it can analyse the coordination connection between agile elements. Even more, it allows to define the system behaviour of coordination systems, summarizing the mutual influence of several coordination elements. To evaluate the suitability of agile coordination strategies it is also able to account for the influence of design project characteristics. Lastly, it allows to explain how disturbances to agile coordination strategies lead to agile constraints.

The analyses of agile coordination strategies revealed repetitive patterns across agile methods. Agile methods rely on mutual adjustment mode, impersonal mode and cognitive mode coordination. Scrum emphasises group mode coordination mechanisms that provide efficacy, synchronization, knowledge exchange and learning. Its practices also recommend individual mode coordination mechanisms. At the same time impersonal mode coordination provides efficiency and productivity and often relies on boundary object coordination mechanisms. Typical impersonal mode coordination mechanisms such as detailed long-term plans or strict roll descriptions are avoided. What sets agile methods distinctively apart from conventional design methodologies is their focus on close team collaboration. This results in cognitive mode coordination mechanisms. These mechanisms enable very efficient coordination within teams that require little explicit coordination activities. They improve design transparency, design speed and mutual learning within teams. Besides cognitive mode coordination a similar pattern of implicit mode coordination is achieved by impersonal mechanisms relying on IT systems such as continuous toolchains that provide continues integration and testing. Unlike cognitive coordination mechanisms they are not restricted to small teams but require costly infrastructure. Both mechanisms provide very efficient coordination since they require very little explicit coordination activities. Agile methods further improve coordination within teams by reducing dependencies to other teams. This focus on intrateam coordination allows to further improve coordination efficiency within teams.

Besides the selection of coordination mechanisms their connectivity within agile methods further explains the reported benefits of agile design. This connection allows the coordination system to self-adjust to changing design requirements. Throughout unclear project phases the coordination system focuses coordination mechanisms on learning and efficacy. While unclarity decreases and projects goals become clear the coordination system changes to more efficiency-oriented coordination mechanisms. This self-adjusting coordination system relies on coordination mechanisms connected by straight forward design practices. The system automatically adjusts the coordination objective and designers can focus completely on design activities. A common characteristic of agile methods are short design cycles that frame the coordination activities. These continuous design cycles are also central for the self-adjustment of the coordination system since each new design cycle allows to reconfigure the coordination mechanisms. The combination of coordination mechanisms from different coordination modes is more powerful than the sum of the individual elements. The composition of coordination mechanisms is designed not for one specific situation, but for a spectrum of requirements. Still, it remains lightweight since the sensitive adjustment happens automatically triggered by agile practices as a reaction to changing projects dynamics. This self-adjusting system results in both very effective and efficient coordination in design projects.

In summary, the coordination theory perspective on agile design enables comprehension beyond the straightforward design practices of agile methods. It shows that agile methods rely on emergent coordination strategies that combine specific team collaboration focused coordination modes and reconfigure the respective coordination mechanisms according to project dynamics. This self-adjustment of the coordination strategy is triggered by agile practices in accordance with design project requirements. This self-adjusting coordination system provides efficiency and effectivity. The presented coordination strategies of agile methods explain the reported benefits. Nevertheless, they have some limitations regarding design project applicability. They rely on compact, autonomous teams with little external dependencies. Intra team coordination is emphasised and inter team coordination is avoided. Especially boundary spanning and cognitive mode coordination mechanisms cannot simply be transferred to larger multiteam systems.

Other research streams have employed similar concepts to explain agile design. Socha et al. interpreted agile projects as complex adaptive systems that consist of interdependent components that learn and adapt collectively to internal and external stimuli in a self-organized manner (Socha et al., 2013). Strode et al. also analysed coordination efficiency in agile design. They focused on mutual adjustment practices relying on group and individual mode coordination. In their view the focus on mutual adjustment coordination engenders high coordination cost (Strode et al., 2011). Especially in large-scale agile coordination theory has been used to analyse constraints of scale in agile design and propose suitable agile practices (Dingsøyr, Bjørnson, et al., 2018). Pikkarainen et al. explain the benefits of Scrum in software development with a focus on communication also relying on boundary objects (Pikkarainen et al., 2008). Evans et al. analyse the benefits of Scrum relying on the concept of social capital (Randy Evans and Carson, 2005). Pries-Heje et al. also rely on coordination to explain the benefits of Scrum with a focus on the articulation theory as a prerequisite for successful collaboration based on a case study (Pries-Heje and Pries-Heje, 2011).

4.3 Inter team coordination in second generation large-scale agile methods

The presented analyses of agile coordination strategies were focused on the small-scale agile methods Scrum and XP. Their intended applications are small or medium-sized teams with minimal external dependencies. Both frameworks recommend autonomous teams and therefore avoid inter team cooperation. Little to no explanation is given how inter team coordination in larger projects should be structured.

In this subchapter, coordination strategies of scaled agile methods are analysed. The emphasis is put on inter team coordination mechanisms. Specifically, the second-generation (Dingsøyr et al., 2021) large-scale agile software design frameworks Large Scale Scrum LeSS and scaled agile meetings and Scaled Agile Framework SAFe (Leffingwell and Kruchten, 2007; SAFe, 2021) were selected because both are based on Scrum at the team level and recommend test driven development similar to XP. This facilitates comparison between the small scale and large-scale agile coordination strategies. In the 2020 industry report State Of Agile (VersionOne, 2020) both frameworks are ranked highest in industrial application. Regarding their coordination strategies LeSS and SAFe open a spectrum with varying foci between mutual adjustment and impersonal mode inter team coordination mechanisms. The inter team coordination strategy analyses differ from the analyses of the agile methods Scrum and XP. Only inter team coordination mechanisms and corresponding parts of the frameworks are analysed. Complete descriptions of the frameworks and their intra team coordination mechanisms are not included in this section.

The inter team coordination analyses are divided into several steps. First, a compact introduction to the coordination strategy of the respective framework is presented. This includes a summary of the inter team coordination practices and principles. Second, the coordination mechanisms are subdivided according to the coordination reference model into coordination modes and their mutual influences are analysed. Third, the inter team coordination strategies are summarized and compared to the underlying Scrum coordination strategy. The balance between inter team and intra team coordination as well as the balance between coordination structure and emergent coordination are part of this section.

4.3.1 LeSS - inter team coordination modes and mechanisms

The scaled agile framework Large Scale Scrum LeSS (Larman *et al.*, 2017) is based on the agile method Scrum. The analysis of its inter team coordination mechanisms is focused on the LeSS implementation that is suitable for up to eight teams. Nevertheless, most findings apply to the LeSS Huge implementation. The framework was designed to preserve the lightweight coordination strategy of Scrum and add only necessary coordination structure to avoid incoordination on a project level. According do Dietrich et al.'s classification (Dietrich *et al.*, 2013) decentralized, self-organized coordination is emphasized instead of centralized coordination. This implies that inter team coordination remains the responsibility of the development teams. Like in the Scrum analysis the inter team coordination analysis is based on the roles, meetings and artefacts in LeSS.

The LeSS **roles** are very similar to the Scrum roles and include only few adaptions. A hierarchy is introduced to the Product with the Area Product Owner. The Manager role is added, and leading teams might be responsible for inter team coordination between other teams in large projects if necessary. Scouts and Travellers are team members that stay with other teams for a defined period. Regarding inter team coordination mechanisms all additional role descriptions are standardized blueprints of individual responsibilities (Thompson, 1967) and therefore impersonal mode coordination mechanisms. The Area Product Owner and the Manager imply vertical individual mode coordination mechanisms, while the Scout and the Traveller support horizontal individual mode coordination mechanisms. The PO hierarchy increases the boundary spanning ability of the PO role and improves inter team coordination. The leading team is based on group mode and cognitive mode within the team. Due to the team's inter team coordination responsibility these intra team mechanisms are transformed to inter team coordination mechanisms to the other teams. The leading team's team members become interfaces between the other teams. The LeSS **meetings** remain mostly the Scrum meetings for regular design teams. Still, some team meetings are changed to or complemented by additional multi-team meetings. These multi-team meetings are the Product Backlog Refinement, the Planning II, the Overall Sprint Review, the Overall Retrospective, and the Design Workshop. These multi-team meetings enable group mode inter team coordination. The Overall Review is organized like a bazar which means that most team members can talk to other teams while only few remain to present the team increment. This exchange between individuals is also an individual mode coordination mechanism. Lastly, all these personal meetings bring designers across teams repeatedly together and hence generate cognitive mode coordination within the project across teams. LeSS adapts the Scrum **artefact** Product Backlog to a multi-team Backlog. Its function as an impersonal boundary object coordination mechanism is extended to also include inter team coordination.

Besides these Scrum structures LeSS includes the concept **Community of Practice COP** as a mutual adjustment inter team coordination mechanism. Groups of interest organize themselves across feature teams in these communities to address shared responsibilities such as architecture or to organize exchange between role holders. These communities can have a mentor to lead the meetings and be responsible for the organization of the COP. These communities of practice are group mode inter team coordination mechanisms. Continuous communities also create cognitive mode inter team coordination. Like XP LeSS emphasises **test driven development** including test automation, acceptance testing, continuous integration, and continuous delivery. The continuous integration and test automation of new increments from all teams highlights problems between product parts and the responsible teams immediately. Further personal inter team coordination is triggered by this impersonal mode inter team coordination mechanism. As analysed in XP these practices enable very efficient impersonal mode inter team coordination up to a certain complexity but require elaborate IT infrastructure.

Table 7: Selected inter team coordination mechanisms in Large Scale Scrum LeSS.

LeSS principles, practices	**Coordination mode**
Teams responsible for inter team coordination	Group mode, individual mode
Decentralized over centralized inter team coordination	Group mode, individual mode
Leading coordination teams	Group, cognitive mode
Multi-team meetings	Group mode
Multi-team artefacts	Boundary spanning
Additional roles, little hierarchy	Group, personal, cognitive mode
Community of practice and community mentor	Group mode, cognitive mode
Coordination friendly environment	Impersonal mode
Test driven development	Impersonal mode
Coordination through integration	Impersonal mode

Table 7 summarizes the inter team coordination mechanisms in LeSS. The inter team coordination strategy emphasises group mode coordination and impersonal mode coordination. The lightweight approach relies on self-organized mutual exchange mechanisms between the design teams and underlines decentralized coordination. Still, the LeSS meetings are a centralized connector between roles and artefacts and design teams and present therefore a centralized balanced to the self-organized approach. The balanced inter team coordination strategy highlights personal exchanges between teams and hence individual mode coordination. Leading teams provide further mutual adjustment inter team coordination. Their team structure and exchange with other teams relies partly on cognitive mode coordination. Besides these leading teams and continuous personal exchange in multi-team meetings cognitive mode coordination does no remain a central inter team coordination mechanism in LeSS in comparison to Scrum. Test driven development coordination mechanisms present an additional centralized inter team coordination mechanisms in LeSS. Besides these design principles few additional impersonal structures and hierarchies are implemented. In summary, LeSS applies an inter team coordination strategy based on mutual adjustment and impersonal mode coordination. The focus on group mode and individual mode coordination is close to the original Scrum coordination strategy. Cognitive mode coordination is not as important in inter team coordination. The test-driven development approach increases the relevance of impersonal mode compared to the Scrum coordination strategy.

4.3.2 Essential SAFe - inter team coordination modes and mechanisms

Like LeSS the scaled framework *Scaled Agile Framework SAFe* is also based on the agile method Scrum. Additionally, it integrates aspects of Design Thinking, Kanban and Scrumban as agile methods at the team level. The following analysis of inter team coordination mechanisms is focused on Essential SAFe but the findings also apply to its other implementations Large Solution SAFe, Portfolio SAFe and Full SAFe. Unlike LeSS SAFe adds more structure to provide centralized project and inter team coordination. Inter team coordination responsibility is therefore shifted from the design teams towards the framework structures. Like in the Scrum and LeSS analysis the inter team coordination analysis is based on the SAFe roles, meetings and artefacts.

Compared to Scrum SAFE applies the same **roles** but introduces clear hierarchies between them. The Product Owner role is divided into a Business Owner, Product Management and the Product Owner. The Scrum Master is divided into a Release Train Engineer and the Scrum Master, both roles are personally responsible for inter team coordination. Development Teams are complemented by the independent System Architect role, who is responsible for overall technical product architecture across teams. Additionally, specialized service teams such as the System Team are introduced. Unlike regular design teams they are responsible for the development of infrastructure, product integration, end-to-end testing, or system demos. Regarding inter team coordination these additional roles are standardized blueprints for responsibilities and hence impersonal mode inter team coordination mechanisms. SAFe introduces hierarchies to all Scrum roles which creates vertical individual mode inter team coordination structures at the cost of unformalized horizontal exchange. The role hierarchies increase the boundary spanning ability of the Product Owner and Scrum Master but negatively affect more flexible group, personal and cognitive mode coordination. The concept of specialized service teams is based on group mode and cognitive mode inter team coordination mechanism similar to the leading team in LeSS. The role structure shows a clear shift from team-organized inter team coordination to personalized inter team coordination structures based on role responsibilities. SAFe also introduces the **multi-team meetings** PI Planning, System Demo, Scrum of Scrums, Product Owner Sync and the Inspect & Adapt meeting to provide inter team coordination. All of these meetings are group mode inter team coordination mechanisms. Furthermore, the regular personal exchange within these meetings establishes cognitive mode coordination across teams. These multi-team meetings in SAFe are specified in detail and rely on more specifications than in LeSS. The Innovation and Planning Iteration is another SAFe adaption that repurposes a regular development iteration within each PI to innovation and inter team coordination. Since it is not structured by additional meetings it is both an unscheduled group mode and individual mode inter team coordination mechanism. Similar to its approach on Scrum roles SAFe introduces a hierarchy to the Scrum artefact Product Backlog as well. The **artefact** Program Backlog is a multi-team Product Backlog and hence an impersonal, boundary object inter team coordination mechanism. Besides the Product Backlog SAFe establishes further artefacts such as the Vision, the Roadmap, Milestones, PI Objectives, the Architectural Runway, the Program Board that affect inter team coordination. The vision, the roadmap and the Milestones are impersonal mode inter team coordination mechanisms. The Program Board and the PI Objectives are both generated throughout the Inspect and Adapt meeting and represent inter team coordination boundary objects. During the meeting they structure group mode inter team coordination mechanisms and throughout the PI they function as impersonal mode inter team coordination mechanisms. The Architectural Runway is another impersonal mode inter team coordination mechanism.

SAFe also relies on the **Community of Practice COP** concept to facilitate inter team mutual adjustment coordination like LeSS. Depending of the COP relevance this can include a mentor role. This COP approach is another group mode inter team coordination mechanism. Continuous COPs also generate cognitive mode coordination across teams since personal connections are established between teams. Like LeSS SAFe is based on **test driven development** which functions as an impersonal mode inter team coordination mechanism. SAFe also recommends **colocation** of all teams which enables unscheduled group mode inter team coordination.

Table 8: Selected inter team coordination mechanisms in Scaled Agile Framework SAFe.

SAFe principles, practices	**Coordination mode**
Inter team coordination responsibility in roles and structures	Impersonal, individual mode
Centralized inter team coordination	Impersonal, group mode
Test driven development	Impersonal mode
Coordination through integration	Impersonal mode
Structure artefacts	Impersonal mode
Multi-team artefacts	Boundary spanning
Service teams provide inter team coordination	Group mode
Multi-team meetings	Group mode
Additional roles, little hierarchy	group, personal, cognitive mode
Communities of practice and mentors	Group, personal, cognitive mode
Colocation of all teams	Group mode

Table 8 summarizes the inter team coordination mechanisms in SAFe. Its inter team coordination strategy emphasises group mode and formalized impersonal mode inter team coordination. Unlike LeSS SAFe adds several new roles, role hierarchies and multi-team artefacts to the underlying Scrum framework. This causes a severe shift towards impersonal mode inter team coordination mechanisms at the cost of more flexible mutual adjustment coordination and establishes centralized inter team coordination. The multi-team SAFe meetings remain the central connector to inter team coordination mechanisms based on roles and artefacts. These rigid structures suppress additional personal inter team coordination between individual team members. This results in clear coordination interfaces and role responsibilities but slows direct exchange between teams. Like in LeSS test-driven Development presents the second important impersonal mode inter team coordination mechanism. Service teams provide group mode inter team coordination and increase flexibility of the inter team coordination system. The SAFe coordination strategy replaces cognitive mode coordination with impersonal and group mode coordination compared to Scrum. Artefacts that function as boundary objects such as the Vision also provide implicit cognitive coordination. The SAFe inter team coordination strategy is based to a large degree on centralized impersonal coordination and group mode coordination mechanisms. Additional roles and hierarchies within the roles establish vertical individual mode coordination. Little cognitive inter team coordination is provided in SAFe.

5 Results

"It is a capital mistake to theorize before one has data."
Arthur Conan Doyle

The aim of this chapter is to present the collected data, describe the context of the data collection and summarize findings. Eleven agile pilot projects which were accompanied over a time span of four years represent the main data source. The data is structured according to matching bottom-up and top-down data analyses. The bottom-up analysis identifies and ranks practical problems of agile methods in automotive application contexts while the bottom-up data analysis compares them to the established constraint categories scale and physicality. The presented findings are the foundation of the following discussion chapter of the thesis at hand.

The results chapter is subdivided into four sections. In the first section, the agile pilot projects are described to provide empirical context. The descriptions reflect the same characteristics across all pilot projects and include the objective of the project, its length, the number of affiliated designers and teams, the level of interdependencies between tasks, and the chosen agile method as well as the implemented changes to it throughout the project. The second section presents a bottom-up data analysis to identify and rank the experienced problems of the agile working models across the pilot project. The third section supplements a top-down data analysis to evaluate the relevance of the constraints of scale and constraints of physicality categories in automotive design. The fourth section redefines problem understanding and outlines agile automotive design as an agile scaling problem. Coordination theory is recommended as the appropriate design theory for further discussion of the collected data.

5.1 Agile pilot projects in automotive design

An overview of the agile pilot projects and the encountered problems regarding agile product design are presented in this section of the study. The project descriptions provide context and hence improve understanding of the experienced challenges. Project descriptions are generalized to protect the intellectual property of the partnering company BMW Group. Therefore, project goals and connections to ongoing design activities are generalized. A consistent description sequence is used to improve comparability between pilot projects. First, a generalized specification of the project goals and an estimation of the relevant design phase are described. The presented projects are either situated in the early, conceptual product design phase (about 0-1,5 years after project start) or the serialization phase (about 1,5- 4 years after project start) of automotive product design. None of the pilot projects were situated in later phases. Second, the project size is described. This includes the team size, the number of affiliated teams and the number of relevant stakeholders. Third, dependencies to project external stakeholders from other organization units are categorized qualitatively into low medium and high, depending on the overall share of time spent to deal with them. Fourth, the initially chosen agile working model, implemented changes to it and the duration of the project are reflected. Change was implemented in two phases. The first phase included the initial introduction of agile methods or practices and the second phase focused on the introduction of inter team coordination mechanisms. Finally, pilot descriptions include whether the focus of the pilot project was on hardware or software design and what implications these characteristics had on the agile working model and the testing strategy. Table 9 shows a summary of the characteristics across all selected pilot projects. The following section introduces the design projects individually.

Table 9: Summary of the researched agile pilot projects. The design phase locates the design project within the product development phases conceptualization and serialization. The project size connects the team size, the number of teams and the number of relevant stakeholders. The dependency differentiates between low, medium, and high dependency levels. The agile method refers to the initially introduced agile framework and the implemented change describes adaptions to it during the project. The product type differentiates between hardware and software products.

	Design phase	**Project size, #Stakeholders**	**Dependencies**	**Agile method**	**Implemented change**	**Product type**
Alpha	Conceptual	One team, ~20 SH	Medium	Scrum	Product Owner Team POT	HW
Beta	Conceptual	Two teams, ~20 SH	Medium	Scrum	Product Owner Couple	SW, (HW)
Gamma	Conceptual	One Team, ~10 SH	Low	Scrum	-	HW
Delta	Serialization	One team, ~5SH	Low	Scrum	-	HW
Epsilon	Serialization	One coordination team, several design teams, > 30SH	High	Kanban	Standard iteration length across teams	HW, SW
Zeta	Serialization	Two teams, ~20 SH	Medium	Scrum, Kanban	Team reorganizations, method adaptions	SW (HW)
Eta	Conceptual	> 5 teams, > 30 SH	High	Selected agile principles	Elements of several non-scaled agile methods	HW, SW
Theta	Conceptual	One Team, ~10 SH	Low	Scrum	Team backlog, PO only SH integration	HW, SW
Iota	Late serialization	One team, ~20 SH	Medium	Scrum-based	Product Owner Team POT	HW
Kappa	Conceptual, serialization	> 5 teams, > 30 SH	High	LeSS-based	several	SW
Lambda	Conceptual, serialization	> 5 teams, > 30 SH	High	SaFE-based	BizDevOp	SW

The objective of pilot project **Alpha** was the design of a revolutionary hybrid drive train within the restrictions of an already existing vehicle architecture. Since this included conceptual work it was part of the early phase of the product design process. Regarding project size the project included a core team of seven designers, several adjunct teams and various stakeholders. The cross-functional core team represented a broad specialization range in drive train and vehicle architecture. The existing vehicle architecture and the participating number of organization units resulted in a medium level of relevant dependencies. Project duration was three months, and the chosen agile working model was based on to the Scrum method. It included a Scrum master and the Scrum meetings. A Product Owner team was introduced to integrate important stakeholders. The project focused on hardware design and relied on established software tools such as CAD and well-established functional simulation tools. The focus on hardware design affected the agile working model directly since many dependencies to other organizations units such as verification and production had to be considered and integrated. Regarding the small size of the project little inter team cooperation problems affected the core team. Still, stakeholder coordination was challenging. Most testing was done virtually which suited the short iterative development cycles.

The design and implementation of new use cases for alternative user interface hardware was the goal of the **Beta** pilot project. The conceptual work was situated in the first third of the product design process. The project included two teams with one team focusing on use cases and visual designs and one team on implementing the software prototypes on existing hardware. Since this project affected future user interface architectures to a large extent more than 15 stakeholders from several organization units had to be integrated. The project teams required expertise in design, user experience, and software design. Due to the relevance of the project for future user interface architectures a great share of dependencies to other units had to be coordinated. The project duration was three months, and the applied agile working model was based on Scrum with some additional practices (e.g. the W*ork In Progress WIP* limit). Throughout the project a second Product Owner was introduced to improve stakeholder management. Additionally, the meeting structure and team composition was adapted. The resulting prototypes were realized as software embedded in regular user interface hardware. Therefore, both software and hardware characteristics influenced the design process. The teams relied on established visual design and simulation tools to be able to get feedback from user experience experts. The inter team cooperation was handled well even though some designers were company external service provider. Problems caused by the scale of the project mostly occurred due to the challenging stakeholder integration which was addressed by shared review meetings and prototypes that were handed to stakeholders for validation. Testing was problematic especially regarding the employed hardware prototypes. The chosen iteration length of one week was insufficient to design and realize the use cases in prototypes. The teams addressed this problem by implementing two connected design phases that lasted two iterations and accepting incomplete design cycles during iterations.

Like project Alpha the goal of pilot project **Gamma** was the design of an alternative drive train configuration in an existing vehicle architecture. The project was situated in the early phase of the design process. It consisted of one large team that supported a large spectrum of competencies in drive train configuration including production, crash, and durability experts. Compared to project Alpha this larger expertise range allowed the integration of external stakeholders and dependencies into the team. Still the complexity of the task required the repeated integration of additional experts and stakeholders. The project length was four weeks and the chosen agile working model based on Scrum. Little adjustments were made to the working model throughout the project. The focus of the project was hardware design and hence standard hardware design tools such as CAD and simulation programs were applied. Similar to project Alpha physical dependencies to neighbouring modules and components caused dependencies to other organization units. The network of personal contacts of the larger team allowed to coordinate these more efficiently. Testing in this early phase of design was done virtually and caused therefore little problems. Also, stakeholder integration was more successful since personal contacts to the most relevant stakeholders provided direct and efficient communication channels.

The design of new armoured vehicle concepts and the adjustment of the existing armoured vehicle strategy was the goal of pilot project **Delta**. The project was situated in the serialization phase of the product design process and lasted four months. Several teams were involved but only one team applied an agile working model close to the Scrum method with little changes throughout the project. In this team, external dependencies were limited to few essential stakeholders. Most of the activities were based on conceptual work and hence neither software nor hardware product characteristics were relevant to the working model. For the same reason no software tools besides regular office tools were applied. Hardware restrictions limited the solution space, but no project size related problems were noted. Only reviews with the Product Owner were part of the pilot project which allowed to apply short compact design cycles.

Pilot project **Epsilon** was part of a larger project to integrate a fuel cell energy source into an existing vehicle architecture and apply it to a small fleet of test vehicles. The reconfiguration of the existing product was situated after the initial product design process. The project size included a coordination team and several design teams for technical challenges and production integration. The size of the project required a broad spectrum of specializations ranging from project management to mechatronic product design including visual exterior vehicle design, crash simulation and manufacturing. The size of the project resulted in various project internal and external dependencies that severely affected the chosen Kanban-based agile working model. The agile pilot project duration was two months, but the overall design process continued for two years. A project-wide iterative cycle was introduced to synchronize the teams. Team compositions were reconfigured to enable cross-functional and co-located teams since dependency management between teams was a problem. Additionally, Retrospectives were introduced to increase continuous improvement and adaptivity of the initially chosen agile working model. Both hardware and software design tasks were necessary and standard tools were applied. Testing affected the agile working model severely since complete vehicle tests required detailed preparations and prototype constructions that did not fit the short iterative design cycles. Regarding prototype testing the coordination team had to integrate many stakeholders which were not directly affiliated with the project. This challenging situation resulted in repeating problems in project coordination.

The design of a user-centric, software-based function to enhance efficient driving was the goal of pilot project **Zeta**. The project was situated late in the overall product design process. One large team was responsible for the design of the software trainer. The team was repeatedly divided to focus on specific tasks. Activities were divided into conceptual functionality, user interface and visual design on the one side and technical implementation as working software on the other side. The project was object to medium external dependencies. Cooperation with the organization unit responsible for the central processing unit were as important as the integration into the existing design language. Also, external software service provider had to be integrated. The project lasted four months and results were provided much faster than in comparable non-agile projects. The agile working model was a combination of practices from Scrum and Kanban with a focus on Scrum. The working model was continuously adapted to changing requirements. The embedded nature of the product resulted in hardware and software specific tasks. But throughout the project product design was mostly focused on software. Established design tools were applied. The agile working model did not suffer from the hardware focused activities and the integration of the stakeholders into agile meetings worked well. Also, collaboration between the divided teams worked very well. Testing required hardware infrastructure and specific user interface components, but the team was able to manage the required activities even though some hardware specific tasks lasted several iterations.

Pilot project **ETA** included the complete development of a new vehicle generation of a niche car. The project started in the early phase of the product design process. It included various teams and experts that were structured around a central coordinative team including the head of the project. A very broad spectrum of specialities was required since design activities included all necessary steps to design a new car. The same reason led to a very large number of stakeholders and hierarchy that needed to be integrated and caused a complex network of dependencies to other organization units. Such a project usually lasts several years but the research frame was restricted to 18 months. Even though the overall product design process remained a stage-gate model essential agile principles and practices were applied in varying intensity at different levels. The core team was restructured to allow for a cross-functional team. Few central meetings were scheduled, and the project head position was implemented as a product owner. Supporting organization units including hardware prototype handling were dedicated exclusively to the project which allowed unscheduled meetings and direct exchange. A shared iterative rhythm was chosen for the whole project. Since the project did not have direct grip to some technical design teams from other organization units a Kanban approach was chosen to increase transparency and integrate their results. The hardware dominant nature of the product resulted in severe implications regarding the working model. Necessary full-scale hardware prototypes required long term planning. Prototype testing caused dependencies to many stakeholders and therefore resulted in challenges to the central coordination team throughout the whole project. The number and distribution of teams also impaired inter team cooperation. Additionally, the project was object to upper management attention which led to continuous distractions caused by detailed reporting and top-down driven changes.

Software inspired engineering methods were tested for component development in the **Theta** pilot project. Simplified put, component construction was done autonomously according to manually adjusted restrictions by an advanced, automated simulation tool in repeated design loops. Restriction also included crash and durability requirements while focus areas were component strength and weight. The available construction space and design restrictions were sufficient to enable the simulation tool to generate CAD geometries, which rely on additive manufacturing for production. This generative approach reduced necessary testing effort since these criteria were already part of the software driven design. The project was a subproject of the Zeta pilot and focused on one structural component of the vehicle project. An additional goal of the Theta project was to establish the new design technology. One team of experts regarding the new design process collaborated using a Scrum approach. They were supported by several external engineering service provider. Additionally, a product owner team consisting of high-level management supported the project. The novelty of the technology required parallel verification from existing testing methods which caused dependencies to further organizations units such as prototype manufacturing. The complete project including prototype manufacturing lasted five months. Even though hardware products were part of the design process the approach allowed to reduce constraints of physicality significantly due to virtual mirroring of physical dependencies and due the integrated product testing. Therefore, short agile design cycles were applied successfully. Still in this project parallel testing was coordinated by the team to verify the new design approach based on the simulation technology. The number of stakeholder and cooperation partner required a large coordination effort, but the agile working model enabled the team to successfully manage the resulting dependencies.

Pilot project **Iota** was set up to deal with an emergent issue in electric drive train design that required immediate action. Testing of hardware prototypes showed use cases that may have resulted in negative user experience. Therefore, the existing technical hardware had to be readjusted accordingly. The emerging problems were detected late in the product design process which resulted in pressure from management. One large team consisting of 15 designers was dedicated to the project. The work required a set of specialization ranging from mass scale production integration to mechatronic component development. Additionally, expertise in battery chemistry and crash verification were necessary. According to tasks the team was repeatedly divided into sub teams. No specific agile method was chosen but practices of Design thinking, a Kanban board and Scrum roles were combined. The immediate pilot and hence the research phase only lasted two weeks but further design and verification continued for another two months. Even though the number of stakeholders was relatively low compared to the other pilot projects management expected fast solutions. Hence direct and continuous reporting structures were established. The nature of the design task focused on hardware adjustments which also affected the agile working model. Manufacturing configurations, material design and long-term contracts with external supplier were some of the challenges that resulted from the focus on hardware. Testing was done mostly virtually to avoid slow prototype tooling and construction which suited the chosen agile working model well. To increase testing accuracy different simulations were compared and combined to increase verification accuracy.

The project **Kappa** was situated within the IT department. It presents a large-scale agile transformation of the organization structure from team level up to top management. The scaling framework SaFE was introduced first and comprehensively adjusted to the requirements of the IT department. To better connect the IT department to the partnering organization units and increase cooperation between the partners the SaFE framework was broadened to a BizDevOp approach. The resulting framework is called *Agile Working Model AWM*. The transformation towards this framework lasted several years and presents a continuing effort. It affected a couple of thousand employees and many external service provider. Several agile transformation teams were shaped to support the agile transformation on different levels, adapt the working model and ensure its correct application. Since the IT department is a close cooperation partner of nearly all other organization units many external dependencies led to the BizDevOp reorganization. The integration of stakeholders and partners from both the business and the operation side allowed to bundle dependencies between these partners in respective projects. The overall size of the IT department and the number of affiliated software designer has an important influence on the agile working model. Inter team cooperation was facilitated by multiple agile practices, roles and artefacts. Hierarchy in agile roles and the introduction of project levels ranging from team to company initiative supported transparency beyond individual projects. Still, new agile practices and model adjustments are further developed answering to needs of the design and product owner teams. A complete agile tool chain was implemented to facilitate continuous testing, integration and deployment. The tool chain also improved project management and dependency tracking.

The project **Lambda** provided the software infrastructure and services necessary to support vehicle navigation and autonomous driving functions. The design activity during the research was situated between the first and second half of the product design process. The project consisted of thirteen software design teams, a team of agile coaches and a Product Owner team also including security experts. The necessary expertise was mainly situated in embedded software design. The close cooperation with the autonomous driving organization units resulted in external dependencies to these partners. Still some additional external dependencies existed towards the IT department. Additionally, the project also relied on external service provider which further increased dependencies. The research duration was limited to six weeks in this continuing project. The chosen scaled agile working model LeSS was adjusted at some points to increase inter team cooperation. But unlike the Kappa project adaptions in the Lambda project to the underlying scaled agile working model were few. LeSS was chosen since the most important cooperation partner had implemented this scaled agile working model earlier and framework-based cooperation problems were to be avoided. The project had little problems with hardware design since most of the activity focused on software design. Still, some embedded testing required the installation of hardware-based testing units. Like the Kappa project a complete agile tool chain was used which enables continuous testing and integration. The tool chain also had a direct interface to partnering design teams from other organizations units which enabled cooperation in project management and dependency transparency. More than half of the teams were not co-located on a shared working space (international distribution of teams) which affected inter team cooperation and required adaptions to the agile working model.

5.2 Bottom-up data breakdown

The first step of the data analysis identified reoccurring problems across the agile pilot projects. The collected data was analysed according to a bottom-up coding procedure independent of predetermining design theories. It summarizes the problems that complicate agile working models and represents a first level coding of broader problem categories. These problem generalizations enable comparisons across projects, serve as the base for theory-based analyses and frame the derivation of solution spaces. Table 10 summarizes the problem categories that were encountered throughout the pilot projects. The table does not represent a complete list of all encountered problems but summarizes problems that occurred across multiple pilot projects and that relate to the applied agile frameworks. The experienced problems were cross coded with problem categories from relevant literature and with findings from concurrent interview series. Detailed descriptions of the experienced problems are presented in the following paragraphs.

Table 10: Encountered problems in each pilot project affecting the employed agile working models.

Pilot Projects	**Encountered problems**		
Alpha (V8 PHEV)	● Inter team cooperation ● Team composition ● # Stakeholder and experts	● Information distribution ● Documentation ● Inflexible architecture ● Inflexible requirements	● System integration ● # Specialists ● Integrative design ● Task division
Beta (ABK)	● Inter team cooperation ● Planning ● Resource allocation ● Redundant work ● Distributed teams ● Team composition ● Insufficient communication ● # Stakeholder and experts	● Information exchange ● Documentation ● (communication channels) ● Inflexible architecture	● Prototype (physical) ● System integration ● # Specialists ● Integrative design ● Task division
Gamma (LU PHEV)	● Planning ● Redundant work ● Team composition ● # Stakeholder and experts	● Information exchange ● Documentation ● Inflexible architecture ● Inflexible requirements	● System integration ● #specialists ● Integrative design ● Task division
Delta (Fuel cell)	● Inter team cooperation ● Planning ● Resource allocation ● Redundant work	● Information distribution ● Information exchange ● Documentation ● Inflexible architecture	● Testing ● Prototype (physical) ● Tooling ● System integration

	• Task prioritization • Distributed teams • Team composition • Insufficient communication • # Stakeholder and experts	• Inflexible requirements	• Task division • # Specialists • Integrative design
Epsilon (Armoured vehicle)	• Inter team cooperation • Planning • Redundant work • Task prioritization • Team composition • Insufficient communication • # Stakeholder and experts	• Information distribution • Documentation • Inflexible architecture • Inflexible requirements	• Testing • Tooling • System integration • Task division • Necessary number of specialists • Integrative design
Zeta (Efficiency trainer)	• Inter team cooperation • Planning • Resource allocation • Task prioritization • Distributed teams • Team composition • Insufficient communication • # Stakeholder and experts	• Information distribution • Information exchange	• Tooling • Prototype (physical) • System integration • Necessary number of specialists • Integrative design
Eta (i8)	• Inter team cooperation • Planning • Resource allocation • Redundant work • Task prioritization • Distributed teams • Team composition • Insufficient communication • # Stakeholder and experts	• Information distribution • Documentation • Inflexible architecture • Inflexible requirements	• Testing • Prototype (physical) • Tooling • System integration • Task division • Necessary number of specialists • Integrative design • Task division
Theta (GeDe)	• Inter team cooperation • Resource allocation • Redundant work • Task prioritization • Team composition • # Stakeholder and experts	• Information exchange • Inflexible architecture • Inflexible requirements	• Testing • System integration • Integrative design
Iota (Storage exhaust)	• Redundant work • Task prioritization • Team composition • # Stakeholder and experts	• Documentation • Inflexible architecture • Inflexible requirements	• Prototype (physical) • Tooling • System integration • Necessary number of specialists • Integrative design • Task division
Kappa (FG)	• Inter team cooperation • Planning • Resource allocation • Redundant work • Task prioritization • Distributed teams • # Stakeholder and experts	• Information exchange • Documentation • Inflexible architecture • Inflexible requirement	• System integration • Task division
Lambda (Foresight)	• Inter team cooperation • Planning • Task prioritization • Distributed teams • Team composition • # Stakeholder and experts	• Documentation • Inflexible requirements	• System integration • System integration • Task division

Inter team cooperation problems occurred if task dependencies required several teams to cooperate. Most of the projects applied a Scrum-based agile working model which does not specify inter team coordination mechanisms. Therefore, no advice on inter team cooperation was given. Two pilot projects applied scaled agile approaches that included some inter team coordination practices but still suffered from inter team cooperation problems in automotive design. **Planning** in short iterations was another problem. Dependencies between tasks complicated the generation of an unmistakable task priority. Large tasks included activities that overlapped iterations which further complicated iterative planning. Additionally, external stakeholders requested long term planning to adjust their activities, which in some cases overruled the iterative agile planning practices.

Resource allocation problems were caused by task dependencies between teams. Even though the pilot projects were able to increase transparency regarding their own tasks, little transparency was experienced for activities from other teams of the same project or from other parts of the organization. This lack of information caused nonoptimal resource allocation and **redundant work**. The amount of team external dependencies complicated **task prioritization** and **task division**. Some pilot projects were unable to divide tasks into sufficiently small parts because of task dependencies. Large items prevented continuous iterative development cycles since they required longer time spans than the chosen iteration rhythm. In a conventional reaction more capacity and hence additional teams were used to balance such large items. These capacity expansions resulted in **distributed teams** which were unable to remain in close contact. But distributed teams also resulted from high total numbers of teams in design projects.

In most pilot projects a broad range of expertise was necessary which complicated **team composition** because cross-functionality and upper limit of team size requirements collided. As a result, teams were divided which resulted in dependencies between teams and limited cross-functionality of individual teams. Inadequate or inexistent inter team communication and cooperation practices led to **insufficient communication** (channels) between teams working on dependent tasks. The large **number of stakeholders** and experts further complicated this tendency. These information and requirement exchanges were further complicated by organizational separation of the given organization structures. Cooperation with company external suppliers and certification service providers suffered from existing bureaucracy and legal requirements and therefore severely complicated agile working models. In some cases, this added up to redundant work and increased task prioritization and division problems.

The level and quality of **documentation** was repeatedly criticized since automotive product design relies on the cooperation of many parties for several years. Agile practices that reduced bureaucracy efforts resulted in unreliable documentation levels and made inter team cooperation even more difficult in several pilot projects. Also, legal requirements demanded a very specific level of product design documentation to guarantee user safety compliance. Regarding inter team cooperation agile **communication channels** were criticised as being focused completely on intra team cooperation and being hardly inapplicable to multiteam systems.

Some cases reported problems with the inflexible **product architecture** and **requirements** structure. The existing requirements management was unable to handle the speed and amount of change driven by agile working models. On the other hand, projects reported that they required more structure in requirements definition and a more stable product architecture to handle testing, tooling and dependencies to other components. Agile approaches that minimize architecture predetermination faced challenges in such environments.

Testing and **integrative design** were a large problem in most pilot projects. Software inspired agile testing guidelines were not practicable in hardware design projects. Product focused design activities must be matched with integrative design activities that provide and verify system properties. Automotive verification is still based to a large degree on physical prototypes from component to system level. These prototypes require extensive manufacturing machinery and represent a large share of the overall development effort. The necessary preparation of the **tooling** (manufacturing of prototypes) often contradicted fast iterative testing cycles. Tooling also represented challenges regarding the necessary design of the mass production system which represents a large share of the overall design effort in automotive. Physical dependencies between components further complicated **task division** and testing. Necessary system integration of hardware components to modules and systems further complicated this problem since the slowest component determined complete system tests. Such system tests required **specialists** from various organization units which further increased inter team cooperation problems and hence problem complexity. Additionally, a large share of the development and testing effort in automotive design focuses on design efforts to manage physical dependencies between components to avoid undesired properties of the complete product without being directly affiliated to costumer functions. Implementation of system integration and verification as well as the generation of new design approaches are also viewed as non-function development effort in design projects.

5.2.1 Distribution and relevance of experienced problems

Table 10 shows the distribution of experienced problems across agile pilot projects. To better understand the overall impact of individual problems on automotive design the following section presents a comparison of problem relevance. A ranking of the experienced agile problems was implemented based on the number of affected pilot projects.

The **number of external stakeholders and experts** and **system integration** are the two most relevant problems. They have been reported in all analysed pilot projects and therefore seem to have the strongest influence on agile product design in automotive. The required spectrum of specializations, dependencies between components organization units, external service provider in design activities, the need for full scale prototypes in testing and verification as well as little automated product verification are characteristics of automotive design that cause these problems.

Team composition was problematic in ten out of the eleven agile pilot projects. Especially the generation of cross-functional teams was difficult. Usually, many specialists were required which led to oversized teams, team division or part-time teams. All three options were problematic to the chosen agile working models. The tendency of very narrow specialization fields in automotive design together with the broader required spectrum of expertise in hardware-focused automotive design caused and intensified this problem category.

Inter team cooperation, documentation and integrative design work caused complications in nine agile pilot projects. **Inter team cooperation** problems appeared in both the scaled agile pilot projects Iota and Kappa including multiple teams but also in the smaller projects based on few teams. Even pilot projects consisting of only one team such as Theta suffered from inter team cooperation problems with project external teams. Automotive design activities are tightly interlinked due to the physical dependencies between components and modules and the existing verification approach. The applied inter team coordination practices were not able to solve this and additional coordination demand and required further adaptions.

Insufficient **documentation** intensified this problem since documentation is an important inter team coordination mechanism in automotive design. Therefore, agile practices that reduced documentation effort led to additional inter team cooperation challenges. The relevance of challenges caused **integrative design** work in automotive clarifies that design activities are necessary for the creation of design infrastructure or to focus on problems caused by physical dependencies between components. The employed agile frameworks emphasized focus the product increments and neglected the necessary verification system of necessary product properties.

The high number of specialists, planning, task prioritization and redundant work challenges were reported in eight pilot projects. The large **number of specialists** caused complications in pilot projects that focused on hardware or embedded software. Challenges with agile **planning** practices occurred in larger pilot projects including several teams. Challenges with **documentation** and **task prioritization** were reported from multiteam pilot projects, but also from smaller projects with many external stakeholders such as Alpha. **Redundant work** challenges were distributed equally amongst pilot projects.

The remaining problem categories were reported in six or less pilot projects. Their distribution in pilot projects is interesting since for example **resource allocation** problems were reported in small and large projects independent of hardware or software focused design. This is an indication that even though pilot projects faced similar challenges some approaches were more successful than others in coping with these challenges. For example, pilot projects Beta and Theta faced physical prototyping as an agile challenge. But they had developed completely different strategies to deal with physicality in prototyping.

In summary, most of the experienced problems in the pilot projects are related to inter team or team external **coordination problems**. Increasing numbers of external experts, stakeholders and teams resulted in complications independent of the chosen agile practices and independent of hardware or software design. They caused challenges in inter team cooperation, in planning, in task prioritization, in resource allocation and in task division. All pilot projects with multiple design teams that cooperate on dependent tasks suffered from inter team cooperation challenges. The data analysis also shows that hardware related design activities resulted in more problems to agile working models than software only products. Physical dependencies between components resulted in dependencies between design activities and hence tasks. These dependencies caused larger and more complicated projects and implied project external dependencies. Both factors resulted in inter team cooperation challenges. Additionally, the physicality of the product and the corresponding design activities required a broader spectrum of specializations. While the relative contribution of such specialists to the complete design process diminished the number of required individuals increased. This tendency led to severe coordination complications in system integration.

5.3 Top-down data breakdown: Constraints of physicality and scale

The presented bottom-up data analysis describes the experienced problems, gives context to the pilot project application domain, and ranks the problems according to occurrence across pilot projects. To understand and better differentiate the cause-effect relations between automotive design and the experienced problems a supplementary top-down data analysis was conducted. This analysis regroups the experienced problems (see Table 10) according to the influence of the physicality of the product and the scale of the automotive design process. Both factors have been identified as central constraints to agile working models in the literature research (see sections 2.4 and 2.5). The connection between experienced agile problems and constraints of physicality and or scale is based on cause-effect relations which are presented in the following paragraphs.

Figure 19 summarizes experienced problems from the bottom-up analysis that are related to the scale of the design project. Scaled projects in automotive consist of several sub projects that focus on different aspects of the design process. Each sub project may include several teams. The sub projects are not independent but object to dependencies to other sub projects, due to product or process specific connections. On a system perspective this causes networks of inter team dependencies. Throughout the pilot projects reoccurring characteristics of scaled projects were identified. The following section investigates the cause-effect relations between these characteristics and the experienced problems from the bottom-up analysis.

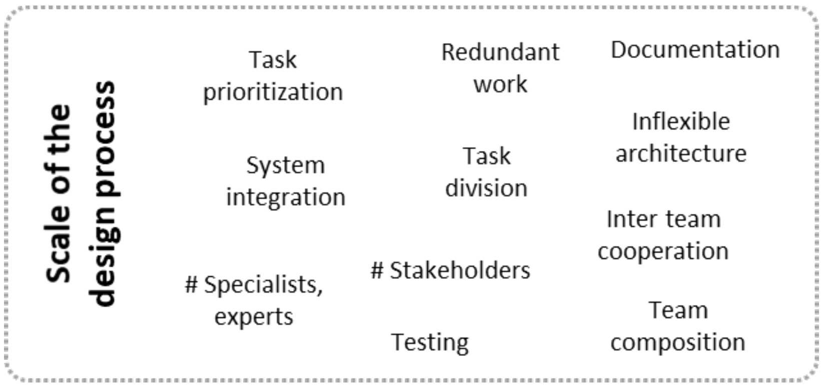


Figure 19: The scale of the project in automotive design relates to most of the experienced problems.

Multiple teams working on the same project required inter team cooperation between the teams. This caused inter team coordination problems since little inter team coordination structures were given in the applied agile working models. Similarly, the agile implementation of design documentation was not sufficient to facilitate inter team cooperation which was a problem in most pilot project. **Team distribution** within teams and distribution between teams to different locations further enhanced these problems because applied agile coordination mechanisms were based on personal exchange in meetings or on sight which was unpractical for large projects. **Dependencies between teams** resulted in task division, task prioritization and system integration problems. Multiple teams also resulted in knowledge and information exchange and hence affected planning efficiency within the scaled project which caused redundant work. Besides the number of parallel teams, **team size** was another scale specific characteristic that caused some of the experienced problems. Large teams complicated intra team cooperation, task division and planning according to the chosen agile practices. Larger teams also led to testing problems due to insufficient exchange within the teams. Additionally, **large numbers of stakeholders and experts** for each team were encountered in most pilot projects. The stakeholders and experts had to be integrated into product design on a team level. This was problematic since they were not obliged to the agreed working practices and demanded special treatment which often consumed a large part of overall team capacity. The encountered **project hierarchy** complicated planning, task prioritization, documentation, communication, and system integration problems because they wanted the teams to also integrate traditional project management and reporting structures. Lastly, the scale of the design systems also included the **heterogeneity of the project** and **project external dependencies** which complicated system integration, planning, documentation, and inter team cooperation.

This summary shows that most of the collected problems to agile working models were influenced by the scale of automotive design projects. It underlines the significance of scaling factor to the experienced problems in agile automotive design. This connection has been drawn in further publications from large-scale agile software design and is summarized in this study as **constraints of scale**.

Figure 20 summarizes experienced agile problems of the bottom-up analysis that are related to the **physicality of the product**. It is the second overarching category that has a strong influence on the experienced problems in the pilot projects. It summarizes attributes that differentiate hardware products from non-hardware products such as software. In automotive these attributes include physical dependencies between components that are not existent in non-physical products. In the pilot projects they affected product verification, prototype manufacturing, product manufacturing but also project size and dependencies between sub projects.

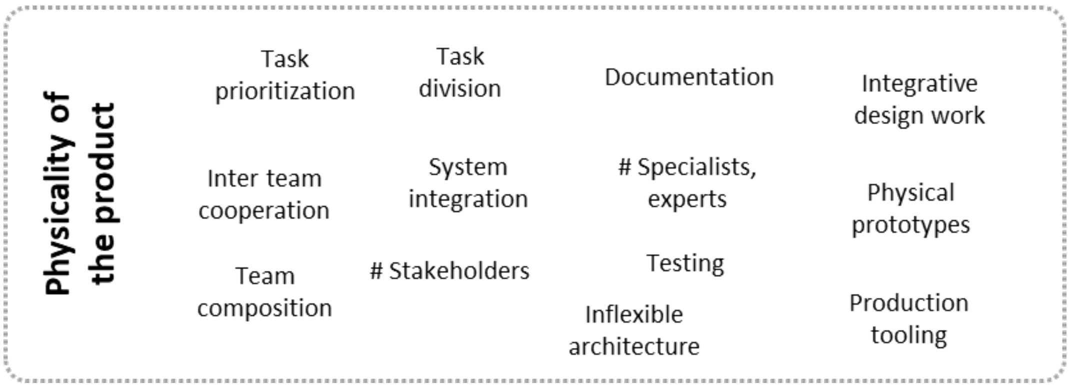


Figure 20: The physicality of the product in automotive design affects most of the experienced problems.

Physical dependencies between sub-products directly affected several of the experienced problems. The physical dependencies between components resulted in inter team dependencies which required inter team cooperation. These dependencies between teams negatively affected task division and prioritization. The management of the physical dependencies required a large spectrum of specialists and stakeholders in each team which complicated team composition. Additionally, the higher complexity level of dependencies decreased product architecture flexibility.

Testing was also affected by the network of physical dependencies since most testing was done on full scale integrative prototypes that required elaborate integrative design efforts themselves and hence contradicted short iterative design cycles. **Physical product verification** and **prototype manufacturing** caused the problems categories system testing, system integration, documentation, integrative design activities number of specialists and tooling for prototypes. Indirectly the necessary specializations to manufacture prototypes also affected inter team cooperation, team composition, the number of external stakeholders, planning and integrative design work.

The design of the **product mass manufacturing** presents a large share of the effort in hardware design which is not the case for software products. This influenced tooling for production, planning, the number of specialists and stakeholder and integrative design work, documentation, inflexible architecture, system integration, inter team cooperation and team composition. In summary, the majority of the collected problem categories from the pilot projects in automotive design are directly affected by the physicality of the product. Therefore, physicality presents a significant constraint to agility in automotive product design.

To conclude the top-down analysis, a match with the bottom-up analysis has been established. The collected problems from the agile pilot projects are directly related to the physicality of the product and the scale of the design process. Therefore, both constraints are central to agile automotive design. Furthermore, the bottom-up data analysis revealed that a large share of the experienced problems is related to insufficient and unspecified inter team coordination. The top-down data analysis complements that these inter team coordination problems are connected to constraints of scale and physicality. Moreover, most problems are influenced by both constraints which allows to draw two conclusions. First, the collected data and the chosen data analysis do not point to an additional equally significant constraint category besides physicality and scale. Second, the overlapping influences of physicality and scale show that both categories are interlinked and have similar constraining effects on agile product design.

5.4 Problem space integration

The presented top-down data analysis clarifies that both agile constraints of physicality and scale are evident and relevant in automotive design. The match with the bottom-up analysis shows significant overlapping between both fields (see Figure 21). Hence, most reported problems are influenced by constraints of physicality and scale. This questions the delimitation of the two separated constraints categories. Alternatively, a combined constraint category would integrate problem understanding. To evaluate this concept the following section compares cause-effect relations between constraints of physicality and the experienced problems across the analysed agile pilot projects.

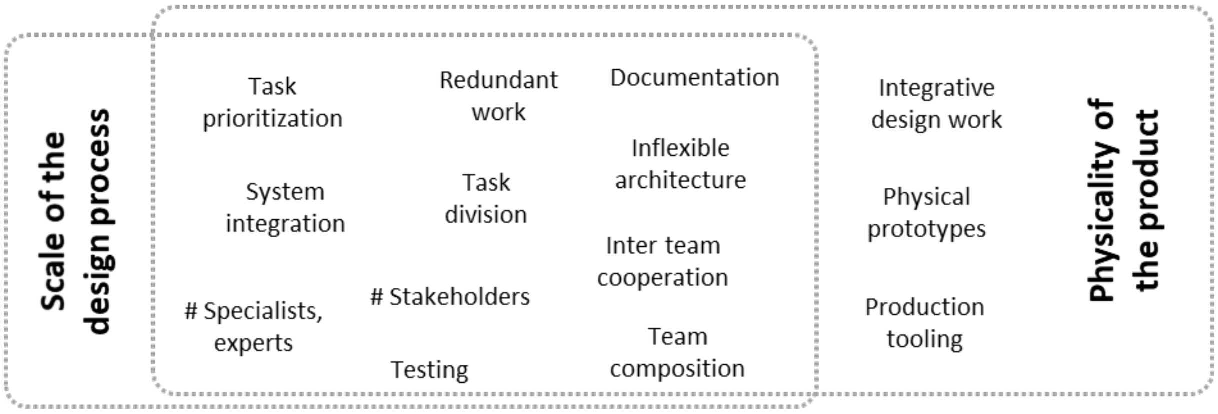


Figure 21: Overlap between problems caused by constraints of scale and physicality across pilot projects.

Inter team cooperation complications are caused by both categories. The physicality of the product requires a larger spectrum of expertise throughout the design process which requires additional specializations and experts. This causes team division into interdependent teams to remain under team size limits and hence inter team cooperation problems. Different teams are responsible for these steps which results in inter team dependencies between those teams. Project scaling on the other hand does not per se affect the required spectrum of expertise but the total number of designers and hence teams to drive the complete project. These teams are part of larger projects which creates dependencies between them and therefore causes inter team cooperation problems. Both constraints result in challenges to agile working models that require inter team coordination mechanisms which are typical for scaling problems.

Other constraints of physicality problem categories result in coordination challenges as well. **Physical dependencies** between product parts result in dependencies between tasks and therefore dependencies between the responsible organization units which are teams in agile design projects. Such inter team dependencies complicate planning and task prioritization. Necessary knowledge is separated in different teams. Problems in these areas indicate insufficient inter team exchange and communication channels that cause inadequate knowledge exchange. Increasing dependencies between tasks also complicate documentation and system integration and hence require additional inter team cooperation mechanisms.

The design of physical products requires additional **integrative design activities** in product verification (e.g. vibrations, acoustic, security, legal admission amongst others) that only apply to hardware (automotive) products. Since these design activities are closely linked to other design activities, they either require additional expertise and capacity in existing design teams or additional supporting teams that increase inter team dependencies. Either way they result in team composition or inter team coordination challenges.

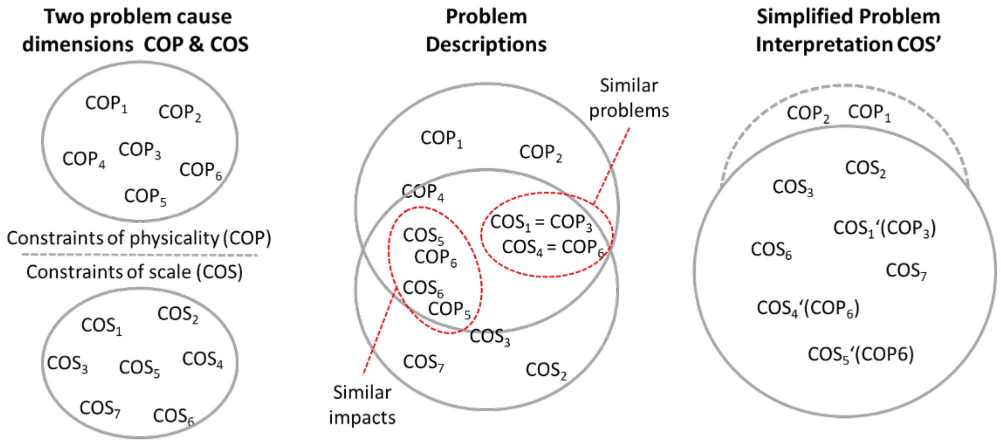
The required **expertise** in the design of physical products requires a larger spectrum of specialisations than in software design throughout the sequential design steps (e.g. prototyping, manufacturing, physical dependencies on component level). The large number of necessary individuals in projects complicate team composition, prioritization, planning, task division and require the cooperation between interdependent teams. To avoid inter team cooperation problems agile teams working on shared tasks require adjusted inter team cooperation mechanisms.

Testing in automotive design is based to a large extent on functional, full-scale **physical prototypes** on a high system integration level. The manufacturing of these prototypes is a complicated and lengthy process that requires additional expertise in prototype manufacturing. These highly integrated prototypes are necessary for various design teams that need to plan and coordinate the shared use. This testing approach also requires adjustments to agile planning practices to better integrate necessary long-time perspectives.

The design of the mass **manufacturing machinery and logistics** presents a large share of the overall design effort in automotive. Unlike software hardware products need to be materialized. In automotive product design, the design of the production and logistics machinery has significant influences on the product design process. It requires additional expertise, capacity and close cooperation with product designer and therefore results in additional inter team dependencies between product and production design teams.

In summary, compared to software design the physicality of the automotive design process results in two distinct process characteristics. First, an **increasing interdependency of the collective design tasks**. Second, a **larger size and heterogeneity of the collective design activities**. In agile product design both characteristics lead to growing numbers of interdependent organization units. These interdependencies result in inter team coordination problems in the analysed data set.

These findings imply that constraints of physicality translate into coordination complications which were originally attributed to constraints of scale. The conclusions encourage a new perspective of constraints of physicality as a subcategory of constraints of scale in this study. Constraints of physicality are therefore viewed as an additional driver of to the existing constraint of scale category (see Figure 22). But this integration of constraints of physicality into constraints of scale is not complete. Some aspects of physicality cannot be attributed to typical scaling problems.



Two problem cause dimensions COP & COS
COP1 COP2 COP3 COP4 COP5 COP6
Constraints of physicality (COP)
Constraints of scale (COS)
COS1 COS2 COS3 COS4 COS5 COS6 COS7
Problem Descriptions
Similar problems
COP1 COP2 COP4
COS1 = COP3
COS4 = COP6
COS5 COP6 COS6 COP5
COS3 COS7 COS2
Similar impacts
Simplified Problem Interpretation COS'
COP2 COP1
COS2 COS3
COS1'(COP3)
COS6 COS7
COS4'(COP6)
COS5'(COP6)

Figure 22: The cause-effect analysis of the experienced problems underlines the relevance of coordination specific problem causes for both constraints of scale and physicality. Therefore, constraints of physicality are viewed as an additional reinforcement of the constraints of scale category to simplify problem understanding and facilitate solution approaches for the thesis at hand.

The integrated problem perspective simplifies problem understanding. It avoids interferences, overlapping and logical gaps between separated problem spaces in the thesis at hand. The unified constraints category facilitates a comprehensive solution approach regarding agile product design for the domain automotive. It allows to apply one theoretical lens to analyse the mechanisms that cause the constraints and construct respective solutions. Coordination theory and the coordination reference model are chosen as theoretical lens to analyse and address this unified constraint category. The respective analysis of the extracted problems from the data set according to the coordination reference model (section 4.1) is presented in chapter 6.1.

6 Discussion

"Speculation and the exploration of ideas beyond what we know with certainty are what lead to progress."
Lisa Randall

The aim of the discussion chapter is to answer research questions two and three. The first section of the discussion 6.1 addresses research question two: What constraints reduce agile design applicability how in automotive design? The summarized problems of agile methods in automotive application contexts of chapter 5 are analysed from a coordination perspective to answer this question. The influence of the automotive application context on the derived agile coordination strategies from chapter 4 is evaluated to determine their functionality in this domain. The functionality assessment allows to further analyse and explain the experienced problems throughout the pilot projects. The first part of the discussion finishes with a summarizing response to research question two.

The second section of the discussion addresses research question three: How to enable agility in automotive product design? The findings of section 6.1 explain the experienced problems with a mismatch between coordination determinants in multiteam automotive design and employed agile coordination strategies. To address these dysfunctionalities of agile coordination strategies in automotive design three scenarios are described. The first scenario supplements the agile coordination strategy with new inter team coordination mechanisms. It also changes the balance of employed coordination modes to match the coordination determinants in automotive. The second scenario relies on new design technologies to improve existing agile coordination mechanisms to provide inter team coordination. The third scenario details how reconfigurations of the product architecture change coordination determinants in automotive and hence increase the applicability of the initially employed agile coordination strategies. The second part of the discussion finishes with a summarizing response to research question three.

6.1 Functionality of agile coordination strategies in automotive design

"We cannot solve problems with the same thinking we used to create them."
Albert Einstein

The results of chapter 5 show that agile methods are less beneficial in automotive design than in their original domain software design. The bottom-up and top-down data analysis in subchapters 5.2 and 5.3 show that both constraints of scale and physicality cause the experienced problems. More precisely, the data analysis clarifies that both factors push towards multiteam design projects. The resulting inter team cooperation networks contradict the original focus of agile design on intra team cooperation and require adjusted coordination structures. Since both constraint categories cause similar inter team cooperation problems, they are merged to facilitate a coordination perspective analysis.

The aim of the first part of the discussion is to better understand the imbalanced agile coordination strategy to answer the *how* question word in research question two: *What constraints reduce agile design applicability how in automotive design?* Unlike in the design theory unspecific bottom-up and top-down data analyses, coordination theory is employed here to analyse the experienced problems regarding the influence of coordination structures on them. The coordination reference model (see 4.1) is used as a theoretical lens to analyse and understand the cause-effect relations between the experienced practical problems and the functionality of the respective agile coordination strategy. The analysis relies on four steps which are discussed in the subchapters of 6.1.

First, automotive design as a generalized application context is categorized according to the coordination determinants of the reference model. Second, the influence of the automotive coordination determinants on the individually employed coordination modes and their mechanisms is demonstrated in relation to the experienced problems from the pilot projects. Third, the influence of the automotive coordination determinants on the mutual connection between agile coordination mechanisms and the self-adjustment of agile coordination strategies are analysed. Fourth, a discussion is presented, how the generalized automotive design application context affects agile product design in response to research question two. Recommendations are presented what coordination modes and mechanisms need adaptions to suit the automotive application context.

6.1.1 Coordination determinants in automotive design

All pilot projects have been conducted in the automotive domain. A coordination-based explanation of the encountered challenges requires a functional representation of this application context according to the presented coordination reference model (see 4.1). The coordination reference model links the applicability of coordination modes to coordination determinants. This allows to evaluate encountered combinations of coordination modes and determinants. The employed coordination determinants are unit size, task dependency and task uncertainty. Different application contexts are remodelled based on combinations of these factors. Since the individual application contexts of the pilot projects varied, an average configuration for automotive design was chosen to assess the employed agile coordination strategies. This average application context was designed according to the experienced project characteristics. The descriptions are in relation to the original agile application context software design in small teams. According to the results from the top-down and bottom-up data analyses the physicality of the product and the scale of the design process were the central influence factors to the automotive coordination determinants (see Figure 23).

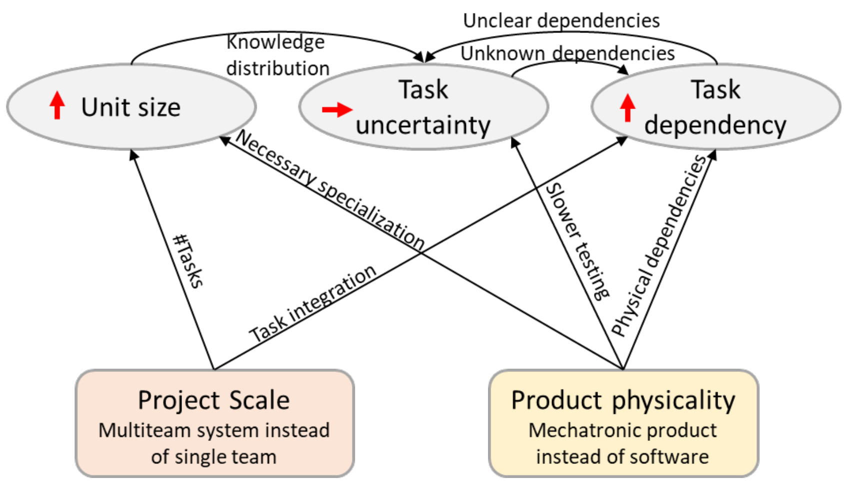


Figure 23: The coordination determinants unit size, task uncertainty and task dependency reflect the project scale and the product physicality in automotive design. While the unit size and task dependency levels increase significantly, the task uncertainty change level remains insignificant. The observed changes of unit size and task dependency mutually enhance each other.

Automotive design projects are characterized by a broad spectrum of divergent design objectives across several interdependent teams. Also, many design activities require a specific sequence. This combination results in complex design systems with both parallel design activities and lengthy overall process sequences. Typical design projects last for several years. This **scale** of automotive design results in multiteam design systems with strong inter team dependencies. Unlike the agile sweet spot intra team software design, they require cooperation between teams. This project scale enlarges the coordination determinant **unit size** because more people and different specializations for system integration and support are necessary. While the total number of teams increases, individual team sizes do not follow proportionally but remain constant under a maximum number of team members. The unit size coordination determinant therefore reflects the number of involved teams as an approximation to the total number of involved individuals. **Task dependency** also increases in scaled projects. The division of large projects into smaller sub projects requires system planning that includes task division and integration to divide and combine the product into sub products and back into the whole product. Each division into sub-projects creates interfaces between teams and results in task dependencies across teams. The project scale does not necessarily affect task uncertainty. But the pilot projects show that in larger projects knowledge is distributed in different teams. If known knowledge is not available to the task responsible designer **task uncertainty** is increased indirectly. Especially unknown knowns (Ramasesh and Browning, 2014) are driven by the larger number of teams while unknown unknowns are not directly affected.

The **physicality** of the automotive product differs from the agile sweet spot application context as well. The design process of physical products implies several phases from conceptualization to manufacturing. It is also more heterogeneous due to physical dependencies between components and the necessary parallel design process of the manufacturing machinery. Both factors result in a broad spectrum of design objectives within the overall process. **Unit size** increases due to the growing number of necessary specializations and due to the larger number of design steps. This affects the number of teams and to a lower degree the size of teams, since more experts need to be integrated. Physical dependencies between components and manufacturing design also increase interlinkage between individual design steps. This significantly increases **task dependency** for physical product design. Most of the design steps in automotive are predetermined through long-term planning, legal obligations, and further restrictions. Therefore, task uncertainty is at a lower level than in less standardized software design. On the other hand, testing of physical systems is less automated and much slower which increases task uncertainty. Regarding task uncertainty these two factors balance each other and require case-specific consideration.

In the pilot projects it has been observed that changing **coordination determinants** also have **mutual influences** onto each other. A larger unit size increases task uncertainty because of recessive knowledge distribution and transparency. Existing team knowledge to solve tasks is not accessible for task responsible teams if inter team communication and exchange channels are not established or if they are inappropriate to the inter team dependency level. Task uncertainty also increases task dependency. If task dependencies are unclear, teams tend to expect additional dependencies to avoid ignoring relevant task dependencies. On the other hand, task dependencies lead to higher task uncertainty since more factors have an influence on the task. Teams are unable to comprehensively understand problems if relevant input from other teams is missing.

The presented coordination determinants in Figure 23 reflect average automotive design projects. But they also vary between projects depending on project scale, physicality of the product maturity and innovation level. The pilot projects show that coordination determinants change according to project dynamics during projects. In summary, coordination determinants in automotive design are influenced by several product and design factors. Compared to agile sweet spot conditions this results in clear changes. The broader spectrum of necessary expertise and the total number of project participants increases the **unit size** significantly. The concept of one growing unit is replaced with a **multiteam system** with growing numbers of interdependent teams but limited team sizes in the coordination reference model. In this study the larger unit size is subcategorized into small, medium, and large regarding the number of interdependent teams. The small unit size represents two teams, the medium unit size up to five teams and the large unit size more than five teams. Such multiteam system require **inter team coordination** concepts. Both physicality and scale increase **task dependency** in automotive design. Relevant factors are physical dependencies between components, physical prototyping, system integration and parallel design of product and manufacturing machinery (and logistics). Like the unit size coordination determinant, the task dependency coordination determinant is also subcategorized into low, medium and high levels in this study. The low dependency level includes dependencies between two teams, the medium dependency level describes dependencies between up to five teams and the high dependency level describes dependencies between more than five teams. In automotive these additional dependencies are well-predictable in general and only few occur unexpectedly. Therefore, unlike the other two coordination determinants **task uncertainty** does not necessarily rise in automotive design. Two opposing tendencies outbalance each other. Project scale and task dependency as well as slower and hardware focused verification often result in decreasing project transparency which increases task uncertainty. But automotive design relies to a large degree on repetitive and predefined activities driven by legal restrictions and long-term plans which improve predictability and hence reduce task uncertainty.

To conclude, coordination determinants in automotive design clearly differ from the agile sweet spot intra team software design. Automotive design is based on **multiteam systems with strong inter team dependencies**. The original focus of agile methods task uncertainty remains on a similar level, while the determinants unit size and task dependency increase significantly. These shifts have significant influences on the applicability of agile coordination strategies which is shown in the following sections.

6.1.2 Functionality of agile coordination modes and mechanisms in automotive design

The next step to understand the problems from the pilot projects is to analyse the influence of the changed coordination determinants onto the agile coordination modes and the respective coordination mechanisms. Cause-effect relations between automotive coordination determinants and employed agile coordination modes are discussed based on the experienced problems from the pilot project.

According to Van de Ven et al. (Ven *et al.*, 1976) and the coordination reference model additional impersonal mode coordination is recommended if the unit size increases (Ven *et al.*, 1976). Still agile **impersonal mode coordination** mechanisms are negatively affected by the increasing **unit size** in automotive in several ways. Broader project specialization and different design objectives complicate and contradict a common terminology in projects. This decreases impersonal communication efficiency. Shared practices and rules that require agreement across teams such as Coding Standards or Simple Design rules lose applicability if design objectives of teams vary largely. Basic agile roles (e.g. Scrum roles) which aim to establish clearly separated responsibilities often oversimplify the more complex automotive design role structures and lead to unclear competencies and responsibilities.

A high level of **task dependency** influences impersonal mode coordination mechanisms as well. Continuous integration and continuous testing practices can hardly be applied due to the higher dependency level between physical subcomponents and the less mature level of automatized testing of hardware in automotive. The manifold verification steps in automotive design mostly rely on physical prototypes to manage task dependencies. Often, sequential and parallel steps are not connected sufficiently to each other. System integration in automotive is more complicated since subcomponents need to be connected into more complex whole products and cannot be employed independently. Task dependency also complicates the application of common Scrum practices like e.g. the Definition of Done. With a growing number of dependencies per task, unfinished parallel tasks prevent task completion. The employed agile methods themselves as overarching impersonal mode coordination mechanisms have severe problems to manage the higher level of task dependency.

Additionally, the level of **task uncertainty** in automotive influences the applicability of agile impersonal mode coordination mechanisms. Since most design activities are well predictable and repetitive the low level of standardization in agile methods contradicts the potential of standardization in automotive design. Agile methods are adjusted to high uncertainty levels. In automotive design this results in unnecessary coordination effort for well-predictable tasks.

In conclusion, the automotive application context decreases the functionality of the lightweight impersonal coordination mode mechanisms in agile methods. Task dependency and unit size have the largest impact on impersonal mode coordination. First, the original lightweight structures are not able to support the task dependency level. The scale and predictability of the process require more efficient integrative impersonal coordination mode mechanisms. Second, the change to multiteam systems contradicts the focus of impersonal coordination mechanisms on intra team mechanisms. Third, automatized impersonal mechanisms such as continuous integration and testing are not available in automotive design. The necessary IT infrastructure and software implementation are not a standard in physical automotive design yet. These finding clarify that even though impersonal mode coordination should be suitable for the larger unit size in automotive, agile methods lack the necessary implementations such as impersonal mode inter team coordination mechanisms.

The reconfiguration of the coordination determinants also affects the applicability of the **boundary spanning** objects, activities, and roles in agile coordination strategies. The larger **unit size** implies more teams with separated design objectives. This includes more specializations in teams and hence larger team sizes. Agile boundary objects such as the Backlog are designed for straightforward implementation, easy updates, and fast uncomplicated knowledge exchange in small projects. In large projects this compactness is not sufficiently versatile to provide transparency across teams and specializations and support interdependent prioritization. Their design does not factor the inter team coordination requirements and the growing number of necessary tasks in automotive design. Regarding the increasing **task dependency** boundary objects are well-suited to enable teams with different design objectives to cooperate efficiently without the need to mutually understand design objectives and terminology completely. Still in the pilot projects this function was overwhelmed by the experienced complexity of task dependencies. Boundary spanning roles such as the Product Owner are adapted to single teams and had difficulty to handle the network of resulting task dependencies, since no specifications are given how to scale the Product Owner role in multiteam systems in the Scrum guide. In summary, most agile boundary spanning coordination mechanisms are adjusted to small projects and therefore showed decreasing functionality in automotive design. That is why agile boundary spanning coordination decreases in automotive application context even though boundary objects are suitable for the larger task dependency in general.

Coordination by mutual adjustment in the **group mode** is essential to all analysed agile coordination strategies. According to the coordination reference model group mode coordination should be suitable for the higher task dependency level. Still, the analysed agile meetings become less efficient in interdependent multiteam systems with experts/designers working in several teams. The relation between useful time and invested time per individual decreases with the number of participants and additional scopes of the meetings. Across the complete design project, the meeting overhead increases disproportionally driven by a larger **unit size** and more participants. Meetings as coordination mechanisms are vulnerable to discussions between individuals that block much larger groups. Group mode coordination also requires complete team presence. Partial presence of teams in meetings results in incomplete knowledge distribution and hence insufficient coordination. Unscheduled meetings are less affected by a larger unit size, since meetings have no fixed number of participants and also apply to subgroups of teams. But the risk of incomplete coordination remains. Additionally, the increase in unit size complicates aspired team characteristics such as co-location which in turn negatively affects the ability for unscheduled meetings. Cross-functionality suffers from the larger expertise spectrum in designers. Especially, the Retrospective a group mode coordination mechanism in most agile frameworks cannot provide its function as central design process adjustment mechanism. It has a project size limit to collect necessary information and generate solutions. The growing **task dependency** in automotive design further affects the applicability of group mode coordination. While intra team dependencies are well managed by the presented meetings, inter team dependencies are not addressed. E.g. Scrum meetings focus completely on intra team dependencies. In the automotive multiteam systems inter team dependencies clearly increase which reduces the applicability of the established agile group mode coordination mechanisms. The experienced changes in **task uncertainty** have little impact on group mode coordination. In summary, automotive coordination determinants severely influence group mode coordination in agile design frameworks. The functionality of the implemented group mode coordination mechanisms decreases significantly even though the coordination mode should be suitable for the larger task dependency level. Like in impersonal mode coordination agile group mode coordination lacks adjusted inter team coordination mechanisms.

In general, **individual mode coordination** is less affected by the presented changes in coordination determinants. The coordination mechanisms are applicable independent of the overall project size because only two designers are connected. Agile frameworks are based on horizontal communication channels. But the larger **unit size** limits their applicability for overall project coordination for efficiency reasons. Personal exchanges of n individuals grow with a quadratic increase of n individuals ((n+1)*n/2). Additionally, some agile individual mode coordination mechanisms such as the On-Site-Customer suffer from locally distributed teams while other practices such as Pair-Programming are not affected. Unlike group mode mechanisms individual mode coordination mechanisms manage increasing **task dependencies** in projects more efficiently. Effort for inter team coordination is limited since dependencies only require two individuals that distribute relevant information in their respective teams. Still this only applies to an inter team dependency complexity limit dependent of the total number of coordination activities. In summary, the coordination determinants in automotive do not restrict agile individual mode coordination mechanisms as much as they affect group mode mechanisms. Changes in unit size and task dependency have little consequences on coordination between individuals. The flexibility and efficacy of these mechanisms allow them to fulfil high complexity coordination tasks. Still, individual mode coordination efficiency decreases for larger projects.

Changes in coordination determinants from the agile sweet spot intra team cooperation to automotive design affect **cognitive mode coordination** most. This implicit form of coordination is based on close intra team cooperation with little team external dependencies. Several requirements of cognitive mode coordination are impaired in automotive design. The greater **unit size** causes inter team dependencies and requires cooperation between teams. Inter team design activities are insufficient to create sustainable, personal inter team relations and trust. They also reduce exchange and personal relations and trust within teams. Distributed teams, multi-project employments and less cooperation time affect proximity and familiarity two main mechanisms of cognitive coordination. The increasing **task dependency** increases complexity of the product and the design process and thus complicates the emergence of shared mental models of the product and the project. Cognitive coordination mechanisms also tend to oversimplify intransparent task dependencies in automotive. The broader spectrum of specializations impedes a common design terminology throughout projects. Still, other cognitive coordination mechanisms such as a shared vision are applicable in automotive design. Briefly, agile cognitive coordination mode mechanisms are affected most by automotive coordination determinants. This influence results in serious problems for the agile frameworks as seen in the pilot projects, because cognitive coordination mechanisms are central to the efficiency of agile coordination strategies.

6.1.3 Self-adjustment of agile coordination strategies in automotive design

In 4.3 it has been demonstrated that the efficiency and efficacy of agile coordination strategies relies on the balance and connection between different coordination modes. This self-adjusting coordination system has been identified as a central reason for the success of agile design frameworks. Therefore, a comprehensive analysis of the impact of automotive coordination determinants on agile coordination strategies cannot be based exclusively on the impacts on the individual coordination modes and mechanisms (6.1.2) alone. Additionally, the connections between the coordination modes need to be examined to comprehend the effect on the agile coordination system. The following section examines the impact of the automotive coordination determinants onto this connected coordination system.

First, the links from mutual adjustment mechanisms to other coordination modes are analysed. **Group mode coordination** is central in agile frameworks and provides coordination speed, efficacy, and flexibility. The implemented agile meetings integrate results, task dependencies, information, verification, and validation. These coordination mechanisms were designed as the integrative platform for the other agile coordination mechanisms in agile frameworks. Increases in unit size and task dependency not only impact the functionality of these meetings as coordination mechanisms but also their connection to impersonal mode coordination mechanisms. In the pilot projects the Scrum meetings were not able to integrate inter team dependencies and heterogeneous information as required. This resulted in insufficient connections between agile coordination modes. For example, the information channelling from multiteam systems into boundary objects such as the Backlog was impaired. Inter team dependencies were not addressed in Scrum meetings which affected the generation and continuous employment of common design standards and rules. Retrospective meetings, as the central learning functionality of agile frameworks, had troubles to adjust the agile coordination systems according to project dynamics for large unit sizes. Such incomplete coordination integration prevents emergent coordination strategies that continuously adjust to changing tasks and coordination requirements. Lastly, agile group mode coordination mechanisms are not able to generate intense personal exchange, repeated activities, and mutual trust necessary to maintain cognitive mode coordination mechanisms in automotive design. Unlike agile group mode coordination mechanisms, agile **individual mode coordination** mechanisms are less affected by the changes in design coordination determinants. Consequently, individual mode coordination mechanisms remain able to initiate other coordination mechanisms or adjust to them. Individual mode coordination mechanisms like the Backlog Refinement still trigger Boundary Object coordination mechanisms. Nevertheless, boundary spanning roles such as the Product Owner are affected by an increasing task dependency which complicates her responsibility to prioritize design tasks. In a nutshell, individual mode coordination mechanisms address specific coordination tasks between individuals. But to function in a coordination system they rely on integrative coordination mechanisms to distribute information and provide project transparency.

Second, agile **impersonal mode coordination** mechanisms such as design rules, roles, plans, testing infrastructure and standards integrate other coordination mechanisms in agile frameworks (see 4.2.1.2 and 4.2.2.2). In automotive design these agile impersonal coordination mechanisms have difficulties to connect the coordination system. The impersonal coordination mechanisms standards and blueprints of action answer well to the large unit size, but not the high task dependency level. High task-dependency levels resulted in incomplete connections between impersonal coordination mechanisms and mutual adjustment coordination mechanisms. Continuous integration systems allow to connect impersonal and mutual adjustment coordination mechanisms based on automated product verification. But in automotive design the technology is not applicable yet to the larger unit size and task-dependency. Agile Boundary Object mechanisms such as the Backlog remain effective to structure agile meetings and hence connect well to group mode coordination mechanisms. But agile Boundary Objects also suffered from the task dependency level and were not able to integrate the amount of design objectives from the interlinked teams.

Lastly, automotive design is based on design in multiteam systems. As described, this significantly reduces **cognitive mode coordination mechanisms**. The support of other coordination modes by cognitive mode coordination is therefore limited to few mechanisms such as the shared vision in automotive multiteam systems. After long-term cooperation in a steady network personal relations and trust may be generated in multi-team systems between individuals across teams but in automotive this is unlikely due to the dynamic change of cooperation partners. The resulting cognitive mode inter team coordination mechanisms cannot replace the connection to other coordination modes as intended in the original coordination strategies for single teams. The loss of cognitive coordination mechanisms deprives the agile coordination system of a central connector in automotive application contexts.

The findings demonstrate that automotive coordination determinants reduce the applicability of the employed agile coordination mechanisms. Furthermore, they severely reduce the connections between them. The function of impersonal and group mode coordination to integrate other coordination modes in agile frameworks is less effective. Alternatives to integrate coordination mechanisms into a connected coordination system are necessary. Additionally, implicit cognitive and implicit impersonal coordination mechanisms are unable to replace and support other coordination mechanisms as emphasized in agile frameworks. These dysfunctionalities impair the **self-adjustment** of agile coordination structures in automotive. The coordination system is not able to adapt to project change and dynamics as required anymore. Agile frameworks therefore lose flexibility and efficiency in automotive design applications.

6.1.4 Suitability of agile coordination strategies in automotive

To conclude this subchapter, a summary of the coordination specific data analyses is given. The researched automotive application context differs significantly from agile sweet spot conditions. The coordination determinants unit size and task dependency increase significantly and cause a system of inter team dependencies. The third coordination determinants task uncertainty remains on a similar level compared to the agile sweet spot. These changes in the coordination determinants in automotive design have severe influences on the applicability of agile coordination strategies as shown in Figure 24.

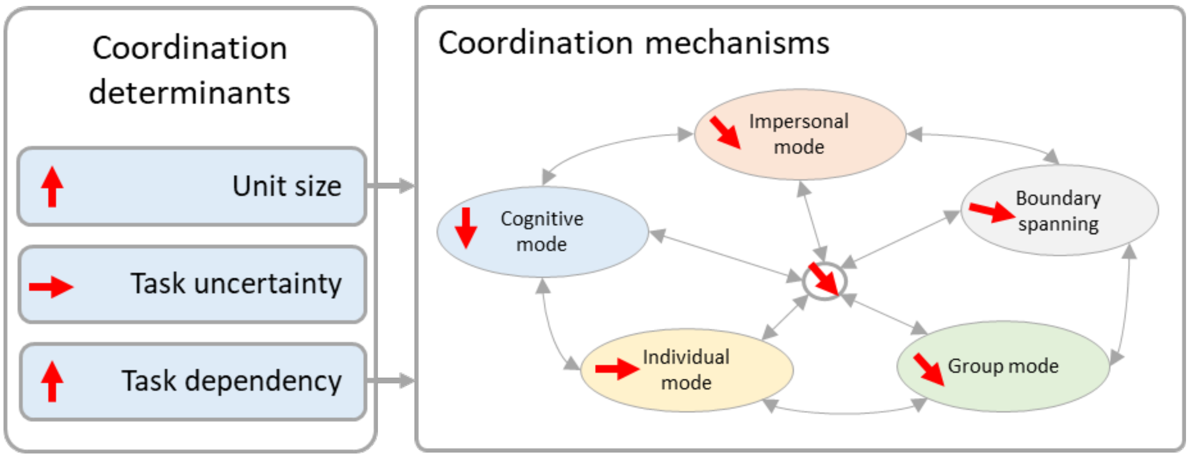


Figure 24: Compared to agile sweet spot conditions (Boehm, 2002; Kruchten, 2013) *automotive design results in different coordination determinants. These changes in coordination determinants result in inappropriate coordination modes, ineffective and insufficient coordination mechanisms and a lack of connectivity for agile coordination strategies. Cognitive mode coordination is affected most, while individual mode coordination and boundary spanning are affected least.*

In the original model of Van de Ven et al. (Ven *et al.*, 1976) increases in unit size are addressed by impersonal mode coordination mechanisms and increases in task dependency by group mode coordination mechanisms. Both are central coordination modes in agile coordination strategies. Contradictory, the data analysis clarifies that the analysed agile group mode and impersonal mode coordination mechanisms are unable to manage coordination in automotive application contexts. Lightweight agile impersonal mode coordination mechanisms are overstrained by the task dependency level while agile group mode coordination mechanisms suffer from the larger unit size. Agile individual mode coordination mechanisms and boundary objects remain mostly functional in automotive application context. Then again implicit cognitive and impersonal mode coordination mechanisms are almost completely inapplicable to automotive multiteam systems.

Additionally, the self-adjusting ability and connection of agile coordination strategies is also limited in automotive application contexts. The ability of central impersonal and group mode coordination mechanisms to integrate other coordination mechanisms is overwhelmed by the number of teams and the level of task dependency. The emphasis of agile coordination strategies on cognitive mode coordination to support other coordination mechanisms is not applicable to automotive design. These findings show that the ability of agile coordination to adapt to project dynamics decreases significantly in automotive design. Figure 24 summarizes the changes of coordination determinants in automotive and the effect of this change onto agile coordination strategies.

In a nutshell, the findings allow to draw two conclusions. First, the analysed agile coordination mechanisms are less suitable to automotive application contexts than to software application contexts. Second, the connection between employed agile coordination mechanisms is weaker in automotive design which affects the ability of the coordination system to adapt to dynamic coordination requirements. Both conclusions are caused by the focus of agile methods on intra team coordination. Nevertheless, these findings do not contradict agile coordination strategies and their emphasis on impersonal mode and mutual adjustment mode coordination in automotive in general. Neither impersonal mode coordination nor group mode coordination face impassable restrictions in automotive design but both lack adjusted coordination mechanisms. Only cognitive mode coordination may turn out unfit as a central coordination mode for large multiteam design systems in automotive independent of the coordination mechanism selection.

To adjust agile coordination strategies to automotive application contexts it is necessary to accept and embrace cooperation across teams. The lack of agile inter team coordination mechanisms is the main reason for its reduced applicability in automotive design. Inter team coordination mechanisms answer directly to larger unit sizes and resulting multiteam systems as well as larger task dependencies and resulting inter team dependencies. Additionally, these inter team coordination mechanisms need to reconnect agile coordination modes to re-establish self-adjusting coordination systems in automotive design. This requires two necessary adaptions of agile coordination systems in automotive design. First, specific agile inter team coordination mechanisms are necessary and must be integrated. Second, cognitive mode coordination needs to be replaced to a large degree by impersonal and individual mode coordination.

6.1.5 Findings in response to research question two

One central objective of the thesis at hand is the analysis whether agile product design approaches are suitable in the domain automotive design. Even though, agile methods and their benefits suit the current challenges and problems of automotive design in theory, agile pilot projects in automotive are necessary to evaluate their performance in practice. Data of eleven pilot projects was analysed regarding the practicability of agile methods in this unfamiliar application context. The focus of research question two are constraints to agility in automotive design. To answer this research question three research streams were connected. Practical problems were summarized and classified according to their statistical frequency across the agile pilot projects. These classified problems were categorized to the existing concepts constraints of scale and physicality via cause-effect relations. Comprehensive understanding of the problem’s roots was enabled through a coordination strategy analysis.

Throughout the agile design pilot projects in automotive design several problems to the methodology reoccurred. A bottom-up data analysis resulted in the following problems. Design teams need to integrate team external stakeholders into the design activities. Additionally, the number of relevant experts often surpasses upper limits of team sizes and hence further increases the number of relevant stakeholders. The large and design phase dependent spectrum of necessary specializations complicates team composition and prevents continuity of team constellations. Agile communication channels are often not sufficiently versatile to connect the necessary network of designers and stakeholder. Large multiteam design projects in automotive result in inter team cooperation problems to the applied agile methods. Agile planning in short iterations gets complicated by the strong interlinkage of design activities. Multiple physical dependencies between components lead to a much more interdependent design process. Task prioritisation and task division suffer from unclear and emergent dependencies between both components and hence tasks. Task dependencies also drive resource allocation problems and redundant work. Distributed teams are unable to remain close personal contact which hampers with agile design paradigms. The documentation granularity is not sufficient to support independent design activities across teams and requires additional exchange between teams. Further problems regarding product architecture and requirements structure are due to management systems that are unable to respond to the speed of agile methods. Automotive design requires elaborate IT and prototyping infrastructure. Integrative design activities necessary to design and provide such infrastructure were not considered sufficiently in the agile pilot projects. Also, slow prototyping in automotive contradicts short iterative design cycles.

A top-down data analysis investigated cause-effect relations between the experienced problems and the physicality of the product and the scale of the design system in automotive. Scaled systems of teams result in characteristics that directly contribute to most of the experienced problems. The same relation is evident for the physicality of the product. These cause-effect relations lead to the answer to research question two that both constraints of scale and constraints of physicality are evident and relevant in automotive design. Additionally, both categories clearly overlap regarding the experienced problems to agile design. The connection of findings of the bottom-up and the top-down data analysis shows that both constraints fields cause similar multiteam cooperation problems. Therefore, in the thesis at hand constraints of physicality are viewed as an additional driver to the constraints of scale category.

This simplification of the problem space allows to analyse the unified problem space regarding the employed agile coordination strategies with the coordination reference model. This analysis clarifies that agile coordination strategies do not function as expected in automotive design. The malfunctions of the coordination strategies directly relate to the experienced problems. The following concatenation of circumstances explains the dysfunctionality of the employed agile coordination strategies and further details the answer to research question two.

First, coordination determinants in large-scale automotive design are distinctly different compared to small scale software design teams. The unit size increases significantly driven by the number of teams, even though the upper limit of team sizes increases only slightly. Task dependency increases due to physical dependencies and a larger spectrum of specializations. Task uncertainty on the other hand increases only to a small degree due to more distributed and less connected knowledge, while basic design activities are more predictable due to well established processes.

Second, these changes in coordination determinants severely influence the suitability of agile coordination mechanisms and respective coordination modes. Typical lightweight impersonal coordination mechanisms are not suited for automotive design. Especially, intra team boundary object and cognitive mode coordination mechanisms are impacted. Even though, the coordination reference model recommends impersonal mode coordination for automotive coordination determinants, agile coordination strategies lack the respective coordination mechanisms. The same applies to agile group mode coordination which lacks inter team coordination mechanisms. Individual mode coordination mechanisms remain functional in automotive application contexts. Implicit mode coordination in agile coordination strategies suffers most from the experienced coordination determinants in automotive. Cognitive coordination mechanisms are inefficient in inter team coordination. The increasing inter team task dependency even decreases the excellent intra team coordination efficiency of cognitive mode coordination mechanisms. Impersonal implicit coordination mechanisms such as continuous integration systems are overstrained by the complexity of the product and the prescribed verification system. In summary, agile coordination modes are either less suitable in automotive design and or lack respective coordination mechanisms. Also, multiteam design systems in automotive require inter team coordination mechanisms, which basically contradicts the original focus of agile methods on intra team cooperation. Especially cognitive mode coordination mechanisms are less applicable in multiteam systems.

Third, the impact on these individual coordination modes and mechanisms severely impairs the connectivity of the agile coordination system. This lack of connection between coordination mechanisms decreases the ability of agile coordination strategies to self-adjust to project dynamics which is elementary to its coordination efficiency and efficacy. Agile meetings as group mode coordination mechanisms cannot provide the interlinkage to other coordination mechanisms any more for multiteam automotive design systems. Especially the balance between group mode coordination mechanisms and boundary objects is overstrained due to number of different parties and the spectrum of specializations. Furthermore, personal exchange necessary for cognitive mode coordination mechanisms decreases with ever larger meetings. Unlike group mode, individual mode coordination mechanisms remain functional in multiteam systems and therefore keep their ability to trigger other coordination modes. Still, boundary spanning roles such as the Product Owner are affected by the larger network of inter team dependencies. Furthermore, within the coordination system individual mode coordination mechanisms suffer from incomplete reactions of other coordination mechanisms to their trigger. High task dependency levels result in incomplete connections between impersonal and mutual adjustment coordination mechanisms. Especially continuous integration infrastructure and boundary objects to structure agile meetings are affected by it. With multiteam systems seriously affecting cognitive mode coordination this leaves only few cognitive mode coordination mechanisms connected to the coordination system.

In summary, the findings clarify why agile coordination strategies are less efficient in automotive design. The experienced project characteristics do not match their requirements. They differ to such a degree that some employed coordination modes become inapplicable, and others lack adjusted coordination mechanism. The connectivity of the employed coordination mechanisms also decreases severely. The change in coordination determinants goes beyond the ability of agile coordination systems to self-adjust to project dynamics since the balance of coordination modes is disrupted and necessary coordination mechanisms are not available. These findings are central to a comprehensive understanding of agile design in automotive design. They provide the base for alterations of agile methods to enable them for automotive application contexts.

6.2 Scenarios to enable agile coordination strategies in automotive design

"[…] well-coordinated teams will […] find an effective mix of mechanisms for the coordination needs of the task they are engaged in."
Alberto Espinosa

The aim of the second part of the discussion is to suggest approaches to counteract the described dysfunctionalities of agile coordination strategies in automotive design in response to research question three: ***How to enable agility in automotive product design?*** *In chapter 5 the practical problems of agile methods across eleven pilot projects have been summarized and compared. The design theory-unspecific data analyses establish the conclusion that the experienced problems are caused by constraints of scale and physicality. Both factors increase inter team cooperation in design activities and hence contradict the focus of agile methods on intra team cooperation. The coordination-specific data analysis in subchapter 6.1 examined how inadequacies of the agile coordination strategies in automotive design caused the experienced problems to agile methods throughout the pilot projects. The original balance of coordination modes and the employed coordination mechanisms in agile coordination strategies are inappropriate for the coordination determinants in automotive design. Figure 25 represents how changes in coordination determinants in comparison to agile sweet spot conditions have resulted in inadequate agile coordination strategies. The swelling river represents the changes in coordination determinants in automotive design in comparison to agile sweet spot conditions. The bridge that crossed the original river but cannot span the swelling river represents the original agile coordination strategies.*

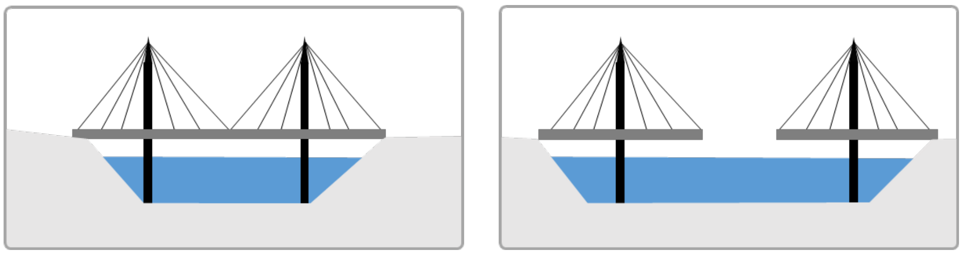


Figure 25: Transcribing sketch regarding the difficulty of agile automotive design. The larger river represents the context automotive design in comparison to the agile sweet spot software design and the bridge represents original agile coordination strategies. While the established bridge was fitted to cross the initial river, it cannot span the enlarged river. The same is true for agile design in the automotive domain. Transferred into the new domain agile product design cannot realize its original functionality.

Three different approaches to answer research question three will be presented. All concepts are based on the coordination perspective on agile design and therefore intend to realize a match between agile coordination strategies and the automotive design domain. To achieve this goal, they focus on different aspects of agile coordination strategies. Figure 26 presents simplified analogies to reflect the opposing ideas behind the three approaches in detail. To recreate the original function of crossing the bridge three different options are available. First, repair and enlarge the bridge. Second, dig a tunnel below the river and avoid the bridge completely. Third, restructure the river so the original bridge can span it again.

In subchapter 6.2.1 a concept to adjust the agile coordination system to match the automotive design coordination determinants is described. This represents the idea to repair and enlarge the original bridge across the larger river. The integration of inter team coordination mechanisms to answer to the requirements of multiteam design systems is the first step towards this approach. The second step is the adjustment of the coordination system connectivity to include the new inter team coordination mechanisms. Lastly, the balance between the employed agile coordination modes needs to be recalibrated to reflect automotive design characteristics. This results in a shift towards coordination modes that better reflect the needs of multiteam design projects.

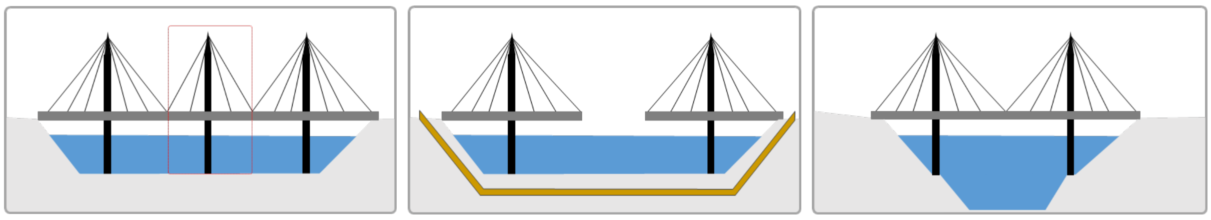


Figure 26: The three sketches (left, middle, right) represent developed scenarios to adjust agile coordination strategies to the coordination determinants in automotive design. In scenario 1 (left) the original bridge is enhanced with additional structures to reestablish its original functionality. It represents the readjustment of agile coordination strategies with additional inter team coordination mechanisms. In scenario two (middle) the larger river is avoided by a tunnel instead of fixing the bridge. It reflects the use of a new design technology to realize agile automotive design. In scenario three (right) not the bridge but the river is adjusted. To carry more water without increasing its width its depth is increased. This approach reflects the idea to adjust the product structure to reestablish agile sweet spot coordination determinants.

In subchapter 6.2.2 the influence of new design technology onto agile coordination mechanisms and strategies is presented. It reflects the idea to build a tunnel below instead of a river above the larger river to get across it. Generative Design as an example of new design technologies is described within the automotive verification and design process. The perspective of Generative Design as a new impersonal mode inter team coordination mechanism is analysed. Furthermore, it is shown how its coordination abilities open the opportunity to change the balance between agile coordination modes and reconfigure the agile coordination strategy.

In subchapter 6.2.3 it is shown how the relation between product architecture and coordination determinants could be used to approximate agile sweet spot conditions in automotive design. This new setting increases the applicability of agile coordination strategies. The concept reflects the idea to change the river width and depth of the swelling river back to its original shape and use the existing bridge to cross it. A modularization strategy is described that structures the product in relation to an agile design enabling organization structure.

6.2.1 Inter team coordination in agile coordination strategies

The aim of this subchapter is to present adjustments to agile coordination strategies to match the analysed coordination determinants in automotive design and hence avoid the experienced practical problems. These adjustments focus on enabling existing coordination modes with new coordination mechanisms to re-establish the original functionality of agile coordination strategies. The presented analysis of agile coordination strategies in automotive design clarifies that both the coordination determinants unit size and task dependency increase compared to agile sweet spot conditions. In practice this results in design activities that require cooperation between teams and hence need inter team coordination. These multiteam design systems in automotive design cannot support the original premise of agile design to focus on intra team cooperation. They require inter team coordination mechanisms to answer to inter team dependencies. To fill this gap a set of inter team coordination mechanisms were developed, introduced and evaluated across the pilot projects. The development of these design artifacts balanced their individual shape and function and their mutual interlinkages as a prerequisite to analyse the resulting system behaviour. These aspects are necessary to re-establish the efficiency, efficacy and flexibility of agile coordination strategies. Both the design of the inter team coordination mechanisms and their system behaviour evaluation were based on the addressed system dysfunctionalities identified with the coordination reference model (see Figure 24).

The subchapter is divided into four sections. In the first section, agile inter team coordination mechanisms from software design are summarized. In the second section agile inter team coordination mechanisms in automotive are described. The set is limited to inter team coordination mechanisms that have been employed in pilot projects. In the third section the presented set of inter team coordination mechanisms is evaluated regarding their suitability to address the relevant levels of unit size and task dependency in automotive design. The fourth section assesses the connectivity between the adjusted set of agile intra and inter coordination mechanisms. This includes an evaluation if the flexibility and self-adjustment capabilities of the reconfigured agile coordination strategy have been restored.

6.2.1.1 Agile inter team coordination mechanisms in scaled software development

Table 11 summarizes findings of relevant secondary literature (Dingsøyr, Bjørnson, *et al.*, 2018; Edison *et al.*, 2021; Nyrud and Stray, 2017) regarding inter team coordination mechanisms in large scale agile design. In the table coordination mechanisms are categorized according to the coordination modes of Van de Ven et al. **Impersonal mode inter team coordination** mechanisms are central team directives, visualizations of dependencies and deliveries, collaborative tool platforms, common Sprint goals, regular product integration steps across domains, scaled agile roles, strategic roadmaps, shared backlogs and open work areas. **Individual mode inter team coordination** mechanisms are iterative proxy collaboration, team member rotation, instant messaging, and informal ad hoc conversation. **Group mode inter team coordination** mechanisms are synchronized sprint cycles, virtual meetings, mid sprint reviews, theme reviews, agile role coordination meetings, cross team demos, physical proximity of teams, experience forums, architecture teams, management meetings across teams, and scaled agile meetings, including the Retrospective, the product Demos, the Planning, and the Backlog grooming. The distribution the coordination mechanisms shows that impersonal mode and group mode coordination seem to be best suited to provide inter team coordination in scaled software development. Details and descriptions of the coordination mechanisms are added in the following subchapters.

Table 11:Impersonal mode, personal mode and group mode inter team coordination mechanisms in large-scale agile software engineering (Dingsøyr, Moe, *et al.*, 2018; Edison *et al.*, 2021; Nyrud and Stray, 2017).

	Edison 2021	**Dingsøyr, Moe 2018**	**Nyrud and Stray, 2017**
Impersonal mode	Central team directives	Masterplan- common backlog	Agile processes
	Visualization (dependencies, deliveries, IT project portfolio)	Open space technology	JIRA (shared, digital backlog)
	Collaborative (tool) platform	Wiki- architectural guidelines	Rules for QA
	Common sprint goal		Open work area
	Regular full integration of software, hardware, mechanics		
	Scaled agile roles		
	Strategic roadmap		
Individual mode	Iterative proxy collaboration	Instant messaging	Instant messaging
	Ad-hoc communication	Rotation of team members	Informal ad hoc conversations
	Team member rotation		
Group mode	Ad-hoc communication	Central team planning	Stand up meetings
	Synchronized sprint cycle	Open work area	Overall Retrospective
	Virtual stand-up meetings	Experience forum	Overall Demo
	Mid sprint review	Scrum of Scrum	Overall Sprint planning
	Theme review meetings	Technical corner- Briefing of teams by architects	Overall Backlog grooming
	PO coordination meetings	MetaScrum – Management meeting across teams	
	Cross-team demo	Board discussions	
	Physical proximity of teams	Overall demos	
	Scrum of Scrum meetings (Grande SoS, feature SoS)		

The spectrum of inter team coordination mechanisms shows the relevance of the problem. But the presented set of mechanisms has been established in large-scale software design. Its relevance in agile hardware design is to be evaluated yet. Throughout the pilot projects several of the presented inter team coordination mechanisms are employed and evaluated regarding their suitability in automotive design. The following subchapter describes agile inter team coordination mechanisms that have been employed successfully throughout the agile pilot projects. The nomenclature between the secondary sources, scaled agile methods and the use in the pilot projects may differ for some coordination mechanisms. For the thesis at hand the employed names in automotive design were selected.

6.2.1.2 Agile inter team coordination mechanisms in automotive design

Group mode inter team coordination mechanisms

The findings of the data analysis in the results chapter indicate unsuitable **group mode coordination** based on the lack of appropriate scheduled meetings for automotive multiteam systems. The larger unit size in automotive projects increases the length of scheduled meetings which results in non-relevant meeting overhead for individual participants. Regarding the large task dependency in automotive agile meetings answer well to it within teams but not between teams. In a nutshell, the employed agile coordination strategies lack adjusted group mode coordination mechanisms that address inter team coordination demand in automotive design projects. To address these findings adjusted and new group mode coordination mechanisms supporting inter team coordination in the form of scheduled multiteam meetings have been tested and evaluated throughout the pilot projects. These scheduled meetings can be categorized into coordination meetings that connect complete teams, meetings that connect agile roles across teams and meetings that connect individuals or communities across teams with similar responsibility or interest.

Scaled agile meetings (e.g. Scrum of Scrum, multiteam grooming, PI Planning, …) connect several teams and allow to address inter team dependencies before, during and after the Sprint cycle. Several teams cooperate during these meetings. The shared Planning allows to discuss and predict inter team dependencies and prepare accordingly. Tasks are divided and distributed to minimize inter team dependencies. The Scrum of Scrum meeting answers to emergent inter team coordination demand and the shared Review meeting connects components dependencies and presents increments composed of interlinked subcomponents. The multiteam Grooming meeting focuses on inter team dependencies and clarifies them within the Backlog as preparation for a successful planning. Such a multiteam Grooming needn't necessarily include complete teams but rather relevant designers from the respective teams. These scaled agile meetings answer well to medium task dependency levels in automotive, but the number of participating teams should not exceed five teams and hence cannot include complete projects. Applied at larger unit sizes they require extensive preparation and often result in inefficient coordination. The extension of the meetings to accommodate several teams results in multiplied participants which negatively affects coordination efficiency. To outmanoeuvre this tendency new non-linear meeting formats were applied. Instead of one person addressing the complete audience parallel structures are employed to ensure both inter team coordination and maintain overall efficiency and flexibility. Formats include *speed dating* or *fish bowl* during planning meetings or the *market place* and *bazar* formats from *Liberating Structures* ("Liberating Structures", 2022) during review meetings. These adjusted agile meetings are divided into joint and separated parts. They provide overall project coordination in the joint sessions and emphasis inter team dependencies in break out parts including only few teams and hence providing more intense coordination. Intra team dependencies are addressed in team individual meetings. Since the capacity of shared areas is often limited digital versions of the meetings have been tested and approved valuable in automotive.

Agile role bearer meetings (e.g. PO synchronization, SM meeting) connect agile role bearers across teams. Much smaller than scaled agile meetings these meetings enable inter team coordination specific to individual agile roles. Throughout the pilot project the PO synchronization emphasised task prioritization and capacity overview across teams while the SM exchange addressed a concerted working model throughout the complete project. Additionally, inter team conflicts and impediments were discussed, and solutions decided upon. These meetings are applicable at medium and large unit sizes and up to a medium task-dependency level. **Communities of Practice** (COPs) connect fields of specialization across teams. Individuals with the same specialization or similar interest can exchange information through these meetings. In automotive COPs were beneficial in product architecture and the verification system. Depending on the demand of the exchange such meetings are structured very strictly or resemble a lose exchange that dynamically adapts to a changing interest of exchange in the design project. Communities of Practice provide inter team coordination across more than five teams and hence answer to large unit sizes. In automotive they have been proven beneficial in integrative design activities. The emergent character of the meeting is applicable to low inter team dependency levels.

The presented scheduled and unscheduled meetings function as group mode inter team coordination mechanisms. They present arenas that encourage inter team coordination between interdependent teams in automotive. As such these inter team coordination mechanisms answer to both the large unit size and the high task dependency in automotive design. In summary, they re-establish the applicability of group mode coordination in automotive application contexts. Still, these adjusted and new meetings are limited regarding the unit size. They answer well to a medium unit size up to five interdependent teams and suffer from further scaling.

These selected group mode inter team coordination mechanisms match well with the mechanisms presented in Table 11. Edison et al., Dingsøyr et al. and Nyrud and Stray summarize similar scaled agile meetings (e.g. Scrum of Scrum, joint Retrospectives, cross-team Demo, theme Review, overall Backlog Grooming) as valuable inter team coordination mechanisms in software design (Dingsøyr, Moe, *et al.*, 2018; Nyrud and Stray, 2017) based on a synchronized Sprint cycle (Edison *et al.*, 2021). Edison et al. recommend agile role coordination meeting such as the PO coordination meeting (Edison *et al.*, 2021). Communities of practice have been successfully implemented in scaled software design projects (Paasivaara and Lassenius, 2014) and large organizations with strong inter team dependencies (Kahkonen, 2004)

Individual mode inter team coordination mechanisms

Regarding individual mode coordination the findings indicate that the analysed agile coordination mechanisms remain functional in automotive design applications environments (section 4.2). Still, with a growing number of teams, coordination efficiency decreases due to an exponential increase of necessary personal exchanges between interdependent teams. Inter team dependencies between few teams are well manageable but larger projects require additional individual mode inter team coordination mechanisms. Another central drawback of the employed agile individual mode coordination is the need for direct exchange and hence co-location. But distributed teams are unavoidable in automotive design, due to the large overall project size and the number of required teams. Therefore, location independent, individual mode inter team coordination mechanisms are necessary. Several specific individual mode inter team coordination mechanisms were introduced and evaluated in the pilot projects to address this inadequacy of agile coordination strategies in automotive design.

Instant messenger and **video conferencing** tools (e.g. Skype, Microsoft Teams, Slack, …) are digital individual mode inter team mechanisms. They are easy to implement and remove the need for co-located teams regarding individual mode exchange. Direct exchange between individuals of different teams is independent of personal location and team size. Tools like Microsoft Teams integrate personal exchange, documentation, and collaboration between team members of different teams. They are more flexible than emails and function more like oral conversations and therefore improve coordination efficiency in comparison. These tools are applicable to the high dependency levels of other individual mode coordination mechanisms since communication in video conferencing includes language and facial expressions. An alternative approach represents **multiteam working areas** that provide sufficient room for several teams with high dependency levels. These areas include shared areas and team specific sites but require large, available spaces.

Team member rotation describes the temporal exchange of team members between teams. The *Traveller or Scout* are specific inter team exchange roles within teams, who are responsible for inter team coordination tasks. They participate in other teams for a defined period or for specific meetings. They establish knowledge regarding both teams, their dynamics and influence Planning and Review meetings to address inter team dependencies. A less formal option is a personally allocated responsibility for inter team coordination within the team for a limited period. This individual remains in her team but changes her responsibilities towards attending multiteam meetings and channels exchange with other teams. Selection of such coordination team members often includes her specialization being suited to the given task dependency between teams. The coordination responsible team member changes according to coordination dynamics and therefore differs from permanent agile roles. In automotive these coordination roles are suitable for small to medium unit sizes with two to five teams. They are applicable up to medium inter team dependency levels. The individual’s capacity for inter team exchange limits the coordination role concept to a medium task dependency level.

The findings from the pilot projects in automotive match well with earlier studies on individual mode coordination in scaled agile design projects. Instant messaging has been proposed as individual mode inter team coordination mechanisms in several studies as depicted in Table 11 (Dingsøyr, Moe, *et al.*, 2018; Nyrud and Stray, 2017). Team member rotations have been reported beneficial for inter team coordination in several industry reports and review paper (Grewal and Maurer, 2007; Lindlöf and Furuhjelm, 2018).

Impersonal mode inter team coordination mechanisms

Even though impersonal mode coordination is recommended for larger unit sizes, the analysed agile coordination strategies lack suitable inter team coordination mechanisms for automotive application contexts (section 4.2). The presented results clarify that the employed agile lightweight impersonal mode coordination mechanisms are insufficient for multiteam design projects in automotive. The scale and the predictability of the design projects require more efficient impersonal mode inter team coordination mechanisms to connect other coordination mechanisms. Throughout the pilot projects several impersonal mode inter team coordination mechanisms were evaluated, and the findings are discussed in the following paragraph.

Synchronized sprint cycles are a requirement of multiteam meetings (Lindkvist *et al.*, 2016; Martensson *et al.*, 2017) in large design projects and interdependent teams need to agree on a synchronized Sprint rhythm. In automotive synchronized Sprint cycles were applicable independent of the unit size and the task dependency level but substantial differences in design characteristics resulted in opposing ideal Sprint length for some teams. To deal with such contradicting requirements longer sprint cycles were defined as double or three times the length of shorter sprint lengths to ensure simultaneous and synchronized meetings every third or second iteration.

Hierarchies of the agile roles Scrum Master and Product Owner are important inter team coordination mechanisms in automotive design. A leading Product Owner maintains the overview of the product as a system of parts and especially the interfaces between them. The other Product Owners focus on delimited parts at a much more detailed level. The leading product owner mediates conflicts between Product Owners and leads with a strategy for the complete product. She guides the other Product Owners to prevent inter team coordination conflicts. This hierarchy allows to distribute responsibilities between two levels and ensures, that cooperation between Product Owners leads to an overall optimum and not local optima. This connecting role also provides a fast and flexible escalation process if inter team coordination problems cannot be solved between the respective Product Owners of the teams. A similar hierarchy works well for the Scrum Master. A leading agile coach is not assigned to an individual team but addresses continuous system design and deals with impediments on a system level. Additionally, she facilitates cooperation between Scrum Masters and ensures that their input is addressed on a system level. The agile coach also represents the interface towards the Product Owner organization. Another responsibility of the role is the continuous adaption of the agile working system and the coordination strategy to the dynamics of the design projects. Both agile role hierarchies are relevant inter team coordination mechanisms in automotive design projects. They are applicable at large unit size levels and up to a high task dependency level.

Information distribution systems also support inter team coordination. They replace direct personal exchange between individuals and teams with online information access. Such systems support inter team coordination regarding information exchange at a low complexity level, but cannot replace direct, personal exchange coordination for more complex tasks. The findings from the pilot projects show that their function as inter team coordination mechanisms relies on the usability, accessibility, completeness and currency of the system and the relevance of the information. The underlying system and tools must guarantee easy access and usability must be sufficiently well to replace often accustomed direct exchange. The provided information must include the complete system and the underlying data must be current. These factors show that such infrastructure needs continuous care and adjustment to provide inter team coordination. Wikis or the tool Confluence have been proven beneficial in automotive at a large scale. Company specific social networks are additional organization maps that provide information regarding current responsibilities of org units within design projects. They might include hierarchical structures to clarify responsibilities. They positively impact individual mode coordination since access to relevant parties is facilitated. Such tools are applicable independent of the unit size level but only up to medium task dependency level. Automatized impersonal mechanisms such as continuous integration and testing are not commonly available in hardware-intensive automotive design yet. Possible digitalized impersonal mode coordination in automotive are discussed in section 6.4.

The high level of task dependency in automotive design are driven to a large degree by the physical dependencies between components. They result in similar effects that can be categorized into vibrations, acoustics, crash behaviour and durability amongst others. To address these categories in all tasks **automotive specific *Definitions of Done*** have been proven beneficial. Tasks are only accepted as done if these criteria have been evaluated or the respective teams have been contacted. This extension to an existing agile practice results into an easy applicable impersonal mode inter team coordination mechanism.

Boundary spanning inter team coordination mechanisms

The deduced coordination determinants in automotive affect the applicability of **boundary spanning** objects, activities, and roles in the analysed agile coordination strategies severely (section 4.2). The larger **unit size** mirrors more design teams with a divergent spectrum of design objectives. Most of the applied agile boundary objects such as the Backlog are designed for straight-forward implementation, easy updates, and uncomplicated knowledge exchange in small projects. In large projects they are not sufficiently versatile to provide transparency and prioritization across numerous interdependent teams and specializations. Regarding the **task dependency levels**, in theory boundary objects enable teams with different design objectives to cooperate efficiently without the need to mutually understand design objectives and terminology completely. Still, in the pilot projects this function was unable to address the experienced complexity of task dependencies. Most agile boundary spanning coordination mechanisms are adjusted to small projects and showed decreasing functionality in multiteam design projects. To address the resulting inter team dependencies several additional boundary spanning coordination mechanisms have been tested and evaluated throughout the pilot projects.

The **multiteam Backlog** as a boundary-object inter team coordination mechanism has been proven in automotive. Such a Backlog includes all relevant tasks of a design project independent of the number of teams. Like in a single team Backlog tasks are related to each other, to clarify dependencies between tasks and teams to improve inter team coordination. Tasks are subcategorized from team to project level granularity and dependency definitions follow this granularity. Depending on the user or the purpose the shared Backlog provides different views on task and additional information. It facilitates a range of use cases starting from personal task management up to agile multiteam meetings. The Backlog supports documentation during multiteam Planning and Review meetings. The shared Backlog remains adjustable to reflect project dynamics. Such a Backlog transparently depicts the complete design project tasks and their dependencies. This information improves task allocation between teams and enables an efficient overall project coordination. Automotive project sizes require digital tools that provide simultaneous access up to thousands of users. For example, JIRA by Atlassian is such an IT tool that provides task management for very large projects without reducing usability for individual team. The multiteam Backlog concept is well suitable for large unit size levels and up to medium task dependency levels.

The Roadmap and the System Map boundary objects address the large number of teams and a high level of inter team dependencies in automotive design projects. The term map is used here as a visualization of information such as different kind of dependencies that facilitates inter team coordination that is easily available to all designers. Unlike rigid plans or rules the boundary object map is regularly updated during agile meetings or by responsible individuals. Without displaying all the available information focus is put on information that improves inter team coordination. Digitalized versions are implemented in tools to improve accessibility and usability in distributed multiteam systems. Similar visualization boundary objects have been mentioned in various publications (Dingsøyr, Moe, *et al.*, 2018; Middleton *et al.*, 2007; Tripathi *et al.*, 2015).

The **Roadmap** is an extension to the multiteam Backlog that specifies at what time larger increments need to be implemented to ensure overall system compatibility or strategic goals. It strengthens the Backlog's ability to estimate future design activities that can be well planned. It also supports effective capacity estimation and hence improves Backlog prioritization. In agreement with the Backlog principles these future activities needn't be specified in detail. These Backlog items additionally include known dependencies to earlier and later design activities. These dependencies improve Backlog prioritization and allow to reduce or transparently display inter team dependencies to improve inter team collaboration. Like other Backlog items the Roadmap items are adjusted according to relevant findings during the Refinement meetings. Especially in automotive this boundary object enhancement is beneficial since large parts of the overall design effort consist of the product verification and product manufacture design. Both design activities are well-predictable and strongly interlinked. The roadmap is applicable up to a medium unit size level and a medium task dependency level.

The boundary object **System or Integration map** integrates the product structure and procedural information. It reflects the overall product architecture including components, modules, and the (physical) dependencies between them. This product structure is connected to the respective design activities and their sequence. The corresponding visualization in the System Map improves transparency of dependencies between components and hence tasks during product design. Teams can rely on the system map during the Backlog Refinement meeting and the Planning meetings to avoid unnecessary inter team dependencies and to define remaining inter team dependencies. This information is valuable to the affected teams since they can adjust their mutual synchronization activities to address such inter team dependencies in advance. Even during the iteration, the System Map facilitates understanding and management of unexpected dependencies between teams. As a boundary object the System Map relies on continuous updates which must be incorporated into the design system to reflect the current system. The system map improves agile design particularly in large and complex automotive design projects. Especially in the complicated automotive verification process the System Map is beneficial since many teams must cooperate on highly interdependent tasks. The System Map is applicable to medium unit size levels and up to large task dependency levels. Teams needn't understand the complete system and can focus on design not coordination activities. Still benefits of the System Map must be balanced with the effort to construct it and keep it up to date. Tripathi et al. proposed a similar digital visualization tool for multiteam Kanban systems in software engineering to answer to multisite organizations (Tripathi *et al.*, 2015).

Boundary spanning roles are also suitable inter team coordination mechanisms in automotive design. Responsibility to ensure coordination between teams or different design objectives in a project is located within a personal role. The SAFe framework proposes the Release Train Engineer and Solution Train Engineer STE roles to extend the Scrum Master role to larger projects. A central responsibility of such a **Project Scrum Master** is inter team coordination. The role connects different teams and facilitates coordination between them. The role is not attached to a specific team but to the system and a close partner of the respective PO level. It is based on a system understanding regarding organization structure, design process and product structure. This system understanding facilitates inter team coordination if teams cannot solve problems independently. The boundary spanning role is applicable up to a medium unit size level and a medium task dependency level. The project guide role is close to Project Scrum Master role relying on more traditional project management tools. A similar role is the **system architect** role. The role might be implemented by an individual or a team of experts. The architect predefines the product architecture and proposes architecture guidelines. The role balances the need for overall architecture interfaces and the ability of the architecture to adjust to findings throughout design. The definition of the product structure and interfaces between components has a significant influence on inter team coordination. Specifications of interfaces may lead teams that need to cooperate. The role is necessary in larger projects to enable teams to focus on specific design tasks. The architect is guided by agile design principles to avoid inter team dependencies and still allow emergent architecture. The architect role is applicable to the large unit size level and the high task dependency level.

Cognitive mode inter team coordination mechanisms

The large unit size and the high task dependency in automotive design result in highly interdependent multiteam systems. The analysed agile cognitive mode coordination mechanisms (section 4.2) are not suitable to answer to this coordination demand between teams. Personal relations and trust between members of different teams in multiteam design systems are at a much lower level than required for cognitive mode coordination since personal exchange happens mostly within and not between teams. Furthermore, the high level of task dependency in automotive complicates the emergence of shared mental models of the product and the design process for members of different teams. In a nutshell, the analysed agile cognitive mode coordination mechanisms are suited for intra team and not inter team coordination. This results in two drawbacks regarding automotive multiteam design systems. First, cognitive mode coordination mechanisms within teams are less efficient, since exchange and cooperation within teams are reduced by a larger share of design activities between teams. Second, team centred cognitive mode coordination mechanisms are not suitable for inter team coordination. To address this flaw in agile coordination strategies in automotive design, cognitive mode inter team coordination mechanisms have been tested and evaluated throughout the pilot projects.

Teams across teams represent the opportunity to transfer cognitive mode coordination from intra team coordination to inter team coordination. Such teams include role bearers, or team members across teams and integrate them into stand-alone or additional agile teams. Within these cross-section teams cognitive mode intra team coordination mechanisms are applicable and result in inter team coordination, since the team members remain interfaces to their original teams. Requirements of cognitive mode coordination such as mutual knowledge, personal trust, shared mental models, shared goals longevity and common grounding (Cannon-Bowers and Salas, 2001; Kang *et al.*, 2006) must be provided within these teams. The cross-section team concept is based on the compromise that some team members are part of two teams. This allows to transfer cognitive intra team coordination mechanisms into inter team coordination mechanisms but also results in drawbacks. Affected individuals must divide their attention between two teams and attend two meeting structures, which decreases efficiency of cognitive mode intra team coordination mechanisms in both teams. The balance between intra team and inter team coordination must be chosen according to project characteristics. Nevertheless, agile role bearer teams and specialized design teams have been proven successful as cognitive mode inter team coordination mechanisms throughout the pilot projects.

Agile role bearer teams transfer agile role responsibilities from individuals towards a team. Such teams function like regular design teams and address excessive workload and complexity that are not manageable by individuals. But foremost these agile role teams answer to strong inter team dependencies that require close collaboration between design teams. Bass and Haxby describe tailored Product Owner teams that address inter team coordination in large scale agile design projects (Bass, 2015; Bass and Haxby, 2019). In the pilot projects Product Owners of different design teams collaborated closely within Product Owner teams to manage design projects that consist of multiple components and respective teams. Especially if dependencies between these components are high such close collaboration of the Product Owner role is beneficial. Like other agile teams the Product Owner teams follow agile rules and require an agile coach and a leading Product Owner. Even though, the Product Owners collaborate within their cross-section teams and fulfil their role as a team, the connection to their own teams remains unchanged to ensure that the agile design teams remain unaffected. The close team setup within the Product Owner teams allows to address inter team dependencies at the Product Owner level. Dependencies are considered during generation and distribution of tasks to facilitate cooperation on the team level or avoid inter team dependencies. Scrum Master or agile coach teams follow the same concept have been proven beneficial as well. Agile role bearer teams improve inter team coordination for high task dependency and medium unit size levels.

Specialized, integrative design teams consist of a cross-section of experts of a similar functionality from other design teams. They ensure certain characteristics for the whole product across design teams in a design project. Like agile role bearer teams these specialized design teams are agile teams with a much higher level of collaboration and exchange than Communities of Practice. In automotive design integration teams that are responsible for product integration and verification have been proven very beneficial. Further specialized design teams have been architecture teams that specify and maintain a common product architecture throughout the design project. Often team members of the architecture team spend more time in the architecture teams than within their original teams, because of the relevance of a common product architecture. Other teams are generated for a limited period like system teams that establish necessary infrastructure such as a tooling system to improve design activities across their original design teams. Even though these teams are limited in time they require at least several months to grow into a team and benefit from the cognitive mode intra team coordination mechanisms.

6.2.1.3 Applicability agile inter team coordination mechanisms in automotive design

In section 6.1.1 coordination determinants have been deduced for automotive design projects in comparison to software design projects. Especially higher unit size and task dependency levels have decreased the applicability of agile coordination strategies in automotive since respective inter team coordination mechanisms were lacking. Throughout the pilot projects agile inter team coordination mechanisms were evaluated regarding their applicability in automotive design. The spectrum of presented agile inter team coordination mechanisms across coordination modes are summarized and evaluated regarding applicability for unit size and task dependency levels in Figure 27.

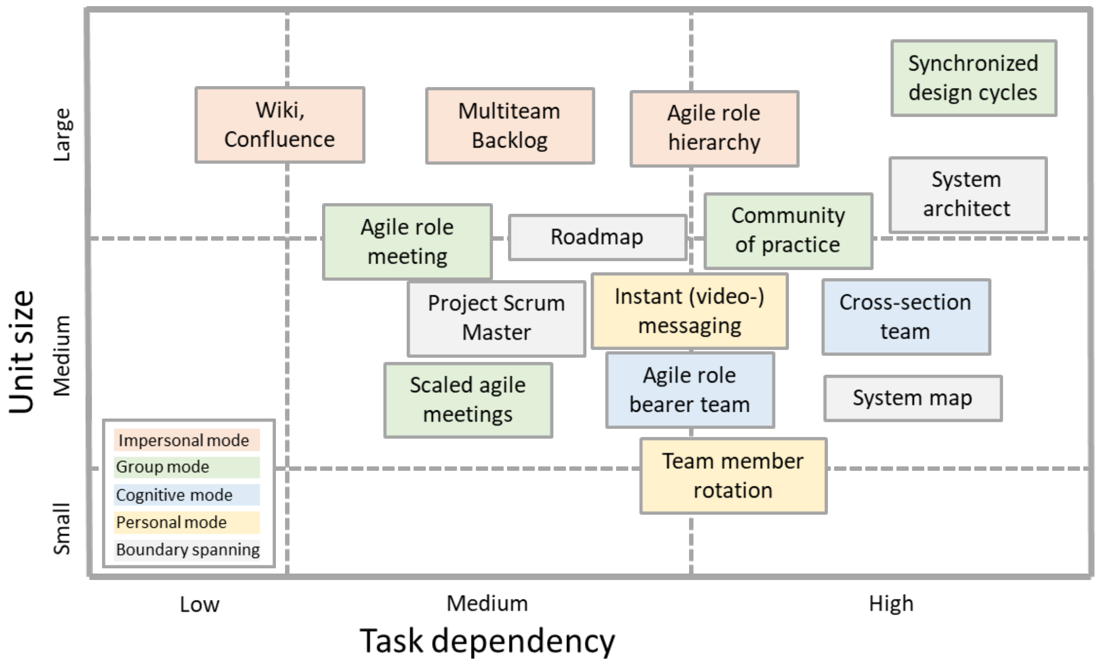


Figure 27: Overview of suitable agile inter team coordination mechanisms in automotive design according to task dependency and unit size levels of the design project. The location in the graph reflects the upper limit of the applicability regarding both task dependency and unit size.

Figure 27 shows that a spectrum of agile inter team coordination mechanisms have been applied and evaluated to answer to the coordination demand in agile automotive design projects. As a set they answer to both large unit size and task dependency levels but within the underlying coordination modes their applicability is distinct. Impersonal mode inter team coordination mechanisms improve coordination for large unit size levels in large multiteam projects, but they are limited to medium task dependency levels. While they provide information independent of the number of participating teams, they lack detail for more complicated task dependency activities. Group mode inter team coordination mechanisms answer well to medium unit size and task dependency levels. These large meetings are limited regarding the number of teams that can be integrated due to the group-based information exchange which is inefficient for large teams. The community of practice concept allows to increase the number of teams since only team representatives are invited but is also limited for very large design projects. The synchronized design cycle concept is independent of the unit size and task dependency a generally applicable facilitator for other group mode inter team coordination mechanism. Boundary spanning objects and roles such as the Roadmap and System map improve inter team coordination up to high task dependency levels. But they are limited to a medium unit size level with some boundary spanning roles such as the system architect being able to address even larger systems. As boundary spanners these inter team coordination mechanisms are limited to a range of design objectives between teams and cannot be extended further since they turn unspecific in the process and lose coordination value to all teams. Individual mode inter team coordination mechanisms are suitable for medium to high task dependency levels but only up to medium unit size levels. Personal exchange is facilitated between teams independent of their location and the total number for high task dependency levels. For larger number of teams this inter team coordination mechanisms needs to be linked to more efficient coordination modes.

The overlapping between inter team coordination mechanisms and modes opens the opportunity to apply different coordination mechanisms for similar design activities. This allows to choose coordination mechanisms not only according to their general applicability but also regarding further aspects such as project dynamics or coordination system connectivity.

6.2.1.4 Connectivity between agile inter team coordination mechanisms

In section 4.3 it has been shown that benefits of agile product design can be derived from the underlying coordination system. The efficiency and flexibility of an agile coordination systems is based on the connectivity of different coordination mechanisms to establish a coordination system that flexibly answers to dynamic project characteristics. Coordination mechanisms must be interlinked to support this property. Therefore, to re-establish agile coordination strategies in multiteam automotive design projects both inter team coordination mechanisms, and the systemic behaviour of the adjusted coordination systems are relevant. This means that inter team coordination mechanisms must be connected to other inter team and intra team coordination mechanisms to ensure coordination system consistency and hence flexibility.

In multiteam automotive design group and impersonal mode inter team coordination mechanisms are connected centrally and provide the foundation of the adaptive agile coordination system as depicted in Figure 28. In the following section the connections between these coordination mechanisms are described.

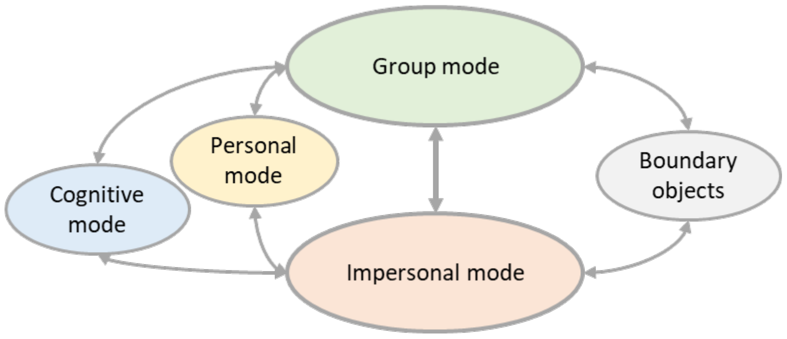


Figure 28: Interlinkage system of inter team coordination mechanisms in agile automotive design. Group mode and impersonal mode are the crucial coordination modes.

The agile group mode inter team coordination mechanisms are multiteam agile meetings, agile role bearer meetings and communities of practice. These meetings are directly connected to the impersonal mode inter team coordination mechanisms synchronized design cycle, multiteam backlog, hierarchies of agile roles and information and distribution systems. Multiteam coordination meetings are only possible if all participating teams agree on a synchronized design cycle with the multiteam meetings as continuous pacemaker between and within the cycles. Furthermore, the multiteam meetings are directly connected to the multiteam backlog. Throughout the meetings the interdependent issues within the multiteam backlog get either refined, prioritized, distributed, or reviewed. The multiteam backlog is the central input source for the multiteam meetings and requires the meetings as adjustment mechanisms. Findings and decisions from the meetings are documented within the Backlog or Wikis. The multiteam meeting structure also mirrors the agile role hierarchy. Throughout the shared sessions the leading PO structures design activities across teams, while the other POs are responsible throughout the parallel sessions with fewer teams participating. Differences between multiteam meetings, agile role bearer meetings and communities of practice allow to address different unit size and task dependency levels as shown in Figure 27. The participation of the same roles, or teams and the application of the same boundary objects ensures that the meetings are interlinked as well.

Boundary spanning inter team coordination mechanisms are directly connected to the agile multiteam meetings. The boundary objects Roadmap and the System map are used to transparently depict organizational, procedural and architectural dependencies in the meetings which facilitates task priorization and distribution. Like the multiteam backlog, the boundary objects are adjusted throughout and after the meetings according to the latest findings. This mutual dependency between multiteam meeting and boundary object coordination mechanisms ensures both efficient meetings despite little personal exchange between teams with divergent design objectives as well as updated boundary objects. The multiteam meetings also connect the Roadmap, System map and the multiteam backlog. Far reaching and less specific information from the latter objects is transferred through the multiteam refinement and planning into short-team, unambiguous design activities in the multiteam backlog. Boundary spanner roles are also integrated into the multiteam meetings. The project SM is responsible for the multiteam meetings and structures them to improve the overall coordination system.

But the multiteam meetings are also connected closely to the cognitive mode inter team coordination mechanisms. Cross-section design teams participate in the multiteam meetings like regular design teams. These meetings are ideal connectors to directly address inter team dependencies with all involved design teams. The agile role meetings on the other hand are necessary for agile role bearer teams to collaborate and synchronize between teams. The less structured group mode inter team coordination mechanisms communities of practice can be transformed into cross-section teams if the task dependency level increases and vice versa if the task dependency level decreases. Unlike in intra team coordination, cognitive mode inter team coordination mechanisms are not suitable to provide a central integrative coordination function in multiteam systems. Cross-section teams and role bearer teams are cognitive mode inter team coordination mechanisms but they cannot provide the same implicit coordination integration as in small scale applications. Individual mode inter team coordination mechanisms require group or impersonal mode inter team coordination mechanisms to distribute information. Multiteam meetings or Wikis are both suitable mechanisms.

In summary, agile inter team coordination mechanisms show a similar ability to generate interlinked coordination systems as intra team coordination mechanisms do in autonomous teams. This ability allows to realize coordination flexibility and efficiency in large scale applications. Furthermore, the agile inter team coordination mechanism are connected to the established agile intra team coordination mechanisms. This connection realizes a balance between intra and inter team coordination to ensure the continuity of high-performance characteristics of agile, autonomous teams. Bass and Haxby emphasis that "*as soon as self-organizing teams work together, they must sacrifice some level of autonomy*" (Bass and Haxby, 2019). Therefore the alignment between autonomous teams and their focus on intra team coordinating mechanisms with a multiteam system goal and its focus on inter team coordination remains a compromise that affects the coordination efficiency of both (Moe *et al.*, 2021).

6.2.2 Technological enablement of agile coordination strategies

The aim of this subchapter is to describe how new design technologies such as Generative Design can enable existing agile coordination mechanisms to match the coordination determinants in automotive design. This extension to the set of inter team coordination mechanisms from section 6.2.1 is presented as an alternative scenario to answer research question three: How to enable agility in automotive product design? The subchapter is based on the publication of Schrof et al. about technology enablement of agile automotive design (Schrof et al., 2019). Such technology driven enhancements of coordination mechanisms allow to maintain the simplicity of existing agile coordination strategies regarding the number of coordination mechanisms and hence avoid unexpected side effects of new coordination mechanisms. Furthermore, the technological empowerment of individual coordination mechanisms leads to a shifted balance of coordination modes within agile coordination strategies. Additionally, this subchapter exemplarily shows how similar design technologies might be analysed and evaluated regarding their impact on agile coordination strategies. Such an evaluation improves design technology selection and combination for practitioners.

The subchapter is subdivided into four sections. First, a comparison to the previous coordination strategy adjustments is drawn to further differentiate the opposing concepts. Second, the product verification process and the applied coordination strategy in automotive design is recapitulated to clarify obstacles to test-driven development, continuous integration, and continuous testing design practices. Third, Generative Design as new design technology is described. Lastly, Generative Design is analysed regarding its benefits to an agile coordination strategy.

The presented adjustments of agile coordination strategies to automotive design determinants focused on additional inter team coordination mechanisms within the original combination of coordination modes. These coordination mechanisms supplement the original set of agile coordination determinants and hence enlarge the initially lightweight agile coordination systems. An alternative approach is to employ new design technology to extend and improve existing coordination mechanisms to meet the requirements of automotive design. This approach allows to enhance the original agile coordination strategy and avoid additional inter team coordination mechanisms. In the presented case the employed technology Generative Design increases the functionality of test-driven development, continuous integration and testing as impersonal mode coordination mechanisms in automotive even beyond their original level in software design. The coordination mechanism improvements are of such relevance that the complete coordination mode balance shifts toward impersonal mode coordination. In Figure 29 the tunnel instead of a somehow modified bridge across a broader river exemplarily shows how a new functionality provides an unexpected solution space.

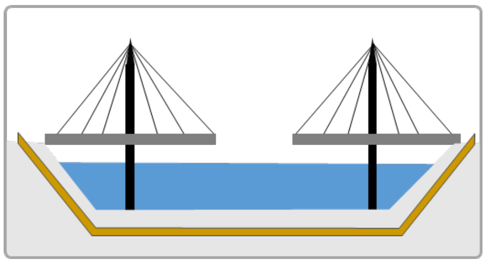


Figure 29: The second scenario to enable agility in automotive design introduces a new design technology that allows to develop the automotive product like software despite its physicality. It opens a new solution space “below” the original functionality of agile coordination strategies.

The complexity and interdependence of the automotive product and the scale of the respective design system require an **interconnected verification system** that must integrate a spectrum of verification objectives of different design disciplines. Both digital and hardware prototypes are necessary to ensure the required product quality. In practice, this results in a complex verification system which requires cooperation between all involved teams. Throughout this integration and verification process group mode and individual mode coordination mechanisms are employed as dominant coordination modes to answer to the level of complexity and number of stakeholders. Additionally, impersonal mode coordination mechanisms such as rules, standards are applied. Together these mechanisms result in a multi-layered and not necessarily transparent coordination strategy in automotive product verification. Consequently, the coordination system slows overall design progress and is prone to distribute incomplete information. Experiences throughout the pilot projects have shown that **automotive design lacks the necessary technology and IT systems for automated test-driven development**

(Beck, 2003). Causevic et al. also emphasis such domain particularities and insufficient tool support as central prohibitors of test-driven development (Causevic et al., 2011). Findings from a case study of a scaled agile product design project confirm that immature software integration tools and infrastructure can result in serious efficiency losses in overall product design (Sutherland and Frohman, 2011). In automotive design necessary continuous integration and testing (Beck, 2003) platforms are not standardized like in software design. For this reason, only a small share of the digital verification system is automated and continuously connected. Their size is not sufficient to reduce coordination effort and increase coordination efficiency significantly.

Despite the lack of standardized continuous integration and continuous testing infrastructure in automotive design new design technologies such as **Generative Design** are developed that provide similar functionality. Generative Design is employed as an umbrella term for the combination of automated design practices such as topology optimization (Bendsoe and Sigmund, 2004) and Vertex Morphing to design 3D geometries (Tyflopoulos *et al.*, 2018). Instead of manually starting and evaluating each design cycle, algorithms automatically run a defined number of iterations according to prespecified objective functions and restrictions until the desired optimization has been reached or unexpected results require interference by a designer. Interfaces to neighbouring components are addressed in the design restriction. Such design practices go beyond the original continuous integration and testing practices since the need for designers to plan, start and evaluate each iteration is no longer necessary.

Without the designer interaction active coordination between designers is not necessary anymore. Relevant information remains within the optimization system until manual adjustment is necessary or an interim result requires evaluation. The digitalized design system also reduces the number of necessary stakeholders regarding the spectrum of relevant specializations since they are integrated into the algorithm. It allows to start the design optimization with incomplete objective and restrictive functions. They are adjusted throughout the optimization which increases the flexibility of the design process. Another benefit is the ability to provide digital prototypes at any time to for example improve costumer integration. Such automated design systems are continuously expanded to integrate a larger share of the automotive product. This allows to reduce explicit coordination activities between teams that used to be responsible for separated parts of the product. In automotive such new design technologies require a digitalized design process and a continuous tool chain. The current combination of digital and hardware prototypes limits the possibilities of digitalized design approaches.

Generative Design or similarly revolutionary design approaches such as Additive Manufacturing have a tremendous influence on agile coordination strategies. From a **coordination perspective** Generative Design is the equivalent of test-driven development, continuous integration and testing and hence an implicit impersonal mode coordination mechanism. Its automatization of design activities provides very efficient coordination. It integrates intra and inter team dependencies. Furthermore, it replaces currently necessary intra and inter team coordination activities since explicit coordination is only requested within and between teams if necessary. Its ability to provide results and prototypes as requested connects it well with group mode coordination mechanisms such as project meetings and boundary object coordination mechanisms based on prototypes. Furthermore, it triggers task specific individual mode coordination between the relevant designers further increasing coordination efficiency. This goes beyond the functionality of the original coordination mechanisms continuous integration and testing as intended in the agile method XP. These capabilities allow to change the agile coordination system to a next level.

Implicit impersonal mode coordination replaces group mode coordination as central integrative coordination mode. The agile coordination strategy balance moves towards impersonal mode coordination since functionalities of former group mode coordination are integrated in impersonal mode coordination. In general, coordination efficiency increases once the system is installed and running. Still, some effort is necessary to update and maintain the system. Coordination efficacy also increases since more coordination happens implicitly within the system and explicit coordination is only requested if necessary and not as a standard. The flexibility of the new coordination strategy remains high since the system directly responds to bugs or changes. This coordination flexibility confirms the request of dynamic coordination systems of Jarzabkowski et al. (Jarzabkowski *et al.*, 2012). Nevertheless, larger updates to the system might negatively affect the coordination flexibility.

In summary, new design technologies might re-establish or even improve existing agile coordination mechanisms to match changes in coordination determinants. This facilitates the transfer of existing agile coordination strategies to new application contexts. In the case of automotive design, the technology Generative Design allows introduce test-driven development, continuous integration and testing to the application context in spite of the existing constraints of scale and physicality. Similarly, technologies like Additive Manufacturing also the potential to improve agile coordination strategies.

6.2.3 Product architecture influence on agile coordination strategies

The aim of this subchapter is to explain how changes to the product architecture present an alternative strategy to enable agile coordination strategies in automotive design. The in subchapters 6.2.1 and 6.2.3 presented strategies to enable agile automotive design were based on adaptions of agile coordination strategies to match automotive coordination determinants. Simply put, they adapted the solution to a changed problem. The inverse strategy would be to adjust the problem and maintain the existing solution. In Figure 30 the increasing water flow is kept in the same river width by increasing its depth.

More specific this alternative implies the adjust automotive coordination determinants to agile sweet spot conditions (Boehm, 2002; Kruchten, 2013) and employ existing agile coordination strategies. If the coordination requirements are closer to the original software application context agile coordination strategies will probably function as expected. Changes in the product structure are an indirect option to influence the coordination determinants. The product structure has a large influence on dependencies between components and hence on dependencies between tasks and teams. It also influences team composition and unit size due to the necessary expertise to design components and operate interfaces between components. This approach to enable agility in automotive design is straight forward for practitioners. Changes to the modularization strategy of the product might are easier to implement than the introduction and maintenance of additional inter team coordination mechanisms or new design technologies. It opens an alternative opportunity to improve agility in product design.

This subchapter is divided into six parts. First, the connection between product architecture and organization structure is explained. Second, the definitions of product architecture and modularization in product design are described. Third, agile core principles are matched with corresponding modularization characteristics to realize a suitable modularization strategy. Fifth, a combination of existing modularization methods is described to realize the developed modularization strategy. Sixth, the changes of the modularization of the product are analysed regarding their influence on agile coordination strategies.

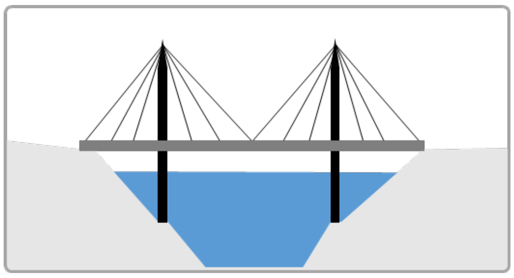


Figure 30: The third scenario to enable agility in automotive design focuses on changing the experienced coordination determinants. The deeper river reflects changes to the modularization structure of the product to support agile core principles and hence recreate agile sweet spot condition coordination determinants.

In 1968 M. Conway published a correlation between the organizational and the product architecture:

"[...] organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations." (Conway, 1968).

This correlation is known as **Conway's law** today and has been proven empirically by MacCormack et al. in their mirroring hypotheses publication (MacCormack *et al.*, 2012). The described correlation is bidirectional. The technical product architecture also influences the functionality of the organization structure. Hatch et al. showed that changes in technical product structure require changes to the corresponding organizational structure (Hatch *et al.*, 2001; Henderson and Clark, 1990). Bowman et al. emphasized the importance between organizational and technical structures since the organizational structure determines the distribution and availability of tacit knowledge (Bowman and Holt, 1998) which predetermines possible coordination structures. To take advantage of this correlation Schrof and Paetzold (Schrof and Paetzold, 2019) propose a product architecture modularization approach that reflects agile organization principles in the product modularization. As a result, it minimizes organizational inter team dependencies and reduces unit size by changing the given product structure. A similar approach has been termed "Inverse Conway Manoeuvre" by Skelton and Pais (Skelton and Pais, 2019).

The **product architecture** defines the properties, functionalities and characteristics of a product based on the product structure and functional structure (Krause and Gebhardt, 2018). The product structure is defined as the hierarchical decomposition of a product into physical modules and components and the interfaces between them (Pahl and Beitz, 2021). The functional structure connects product functions to components. Figure 31 depicts the relation between product architecture, product structure and functional structure. Modular product structures emphasise intra-modular coupling and inter-modular decoupling of dependencies. Göpfert divides four classes of product modularization (Göpfert, 1998). Integral product architectures have low functional and structural independence between modules. Modular product architecture on the other hand aim for structural and functional independence between modules. Between both bookends are functionally modular product architectures which are physically coupled and physically modular product architectures that are functionally coupled. The modularization process structures components and combines them into modules. It includes the definition of physical and functional interfaces between modules. Module driver represent certain characteristics of the aspired structure and influence the modularization process. They emphasis functional, technical or organizational aspects during the modularization (Göpfert, 1998). Krause et al. present an overview of different modularization methods (Krause and Gebhardt, 2018).

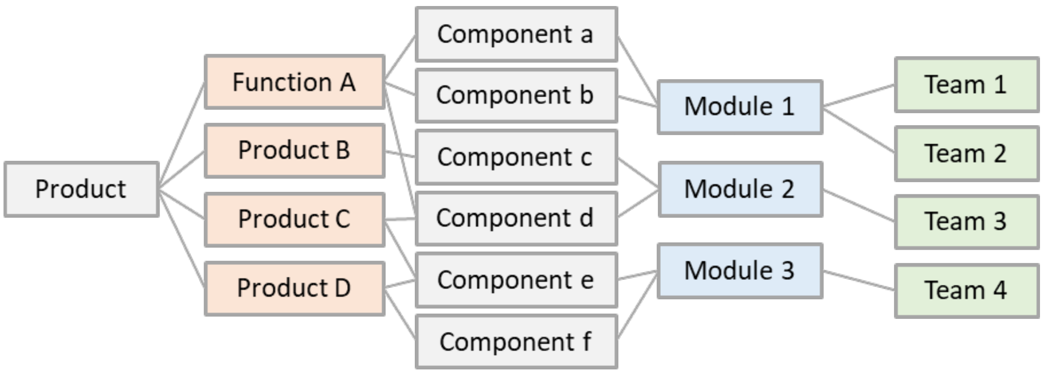


Figure 31: The product architecture maps functions to different components which are combined into modules. Inter team coordination problems (e.g. team 3 and four in module 1) are avoided if modules and teams are matched.

Agile product design is based on core concepts (Baham and Hirschheim, 2021) and principles (Beck and Beedle, 2001) which depend directly on the product architecture. The following section details requirements of agile design on product architecture and module structure features. **Compact, self-organized, responsible, cross-functional, co-located and stable teams** are at the core of agile product design. Ideally, these agile teams are responsible for adequately sized product modules regarding both necessary expertise and workload. In automotive this would translate to smaller modules and a limitation of required specialization spectrum for each module. To allow self-organized and independent teams such modules must be largely decoupled, and remaining dependencies or external expertise requirements must be transparently described in interface definitions. Ideally, the module structure defines clear interfaces and given requirements around any module to enable independence between teams responsible for modules. **Short iterations** that are employed in a continuous cadence for synchronization are emphasized in most agile frameworks. Module and hence increment size have a strong influence on the practicability of such short time periods. The **continuous and automated integration** of increments in product verification as an impersonal coordination mechanism requires precise interface definitions to facilitate automated product integration and testing. The respective verification system would have a strong influence on the overall product quality since it guarantees a functioning system of modules despite their independence on module level. It would also alert designers if systems of or interfaces between modules deviate from expected performance. Continuous **customer validation** on the other hand requires the module structure to mirror costumer value to establish a direct link to customer integration. Additionally, agility in product design emphasises the ability to **adapt quickly to internal or external change**. The product architectures must be able to answer to changes and adapt its structure accordingly. This implies change within given module structures but also includes change of the modularization structure. Such adaptions require several modules to change collectively and might result in additional modules, removal of modules and change of existing module structure. The overall product architecture must also provide uniform product maturity across modules to ensure findings are exchanged between modules as long as changes are still feasible.

In summary, the ideal automotive product structure to enable agility in product design consists of compact, decoupled modules that suit the expertise and capacity of agile teams and allow relevant increments in the chosen iteration cadence. Standardized interfaces between these modules facilitate continuous integration and testing of the overall product in iterative product verification. These interfaces also standardize exchange between modules and teams to facilitate inter team cooperation. The chosen module structure must also represent costumer value to guarantee customer validation. The whole product structure must be able to change according to internal or external impulses.

Two suitable **modularization methods** to realize these requirements are the *Integration Analysis Methodology* of Pimmler and Eppinger (Pimmler and Eppinger, 1994) and the *Methodological support for system building* of Göpfert and Steinbrecher (Göpfert and Steinbrecher, 2000). The Integration Analysis Methodology evaluates the coupling between components and generates modularization concepts. The regarded connections at interfaces are local distribution, energy, information and material. These criteria are evaluated with the Design Structure Matrix (Steward, 1981). Every combination of components is evaluated in the weighted four dimensions. The resulting design structure matrix allows to define modules accordingly.

The *Methodological support for system building* modularization method aspires functional, technical, and organizational decoupling of modules. It relies on five sequential steps. First, definition of premises (requirements, interfaces, organizational suitability). Second, generation of technical (functional and structural) modularization alternatives. Third, evaluation and selection of a technical alternative. Fourth, generation of organizational modularization alternatives, based on the technical modularizations. Fifth, evaluation and selection of overall solution. To adapt the automotive product architecture according to the presented agile design principles a combination of both modularization methods is proposed. The structure of such a hybrid method follows Göpfert's combined modularization of organization and product, while the evaluation of the possible modularization options relies on the Integration Analysis Methodology of Pimmler. The evaluation criteria of the Design Structure Matrix are complemented by the module driver agility in design which reflects the presented agile design principles. This implies that module sizes and connections between modules are structured in a way that facilitates agile coordination structures significantly. Göpfert's five step approach is adapted as a frame for the updated Integration Analysis Methodology. Assuming given agile organization structures implies that certain specifics of the organizational modularization are fixed before the technical modularization. Step four and step three are therefore merged and reflect technical and organizational priorities in the selection of a shared modularization approach.

From a **coordination perspective** the modularization of the product architecture is crucial to changes in coordination determinants. Regarding **task dependency** the decoupling of the modules decreases inter team dependencies and emphasises intra team dependencies. This allows a focus on original agile coordination strategies based on mutual adjustment and cognitive mode coordination within teams. The modularization according to team expertise further reduces inter team dependencies since less team external expertise needs to be integrated. The standardization of interfaces between modules allows to implement automated impersonal mode coordination mechanisms for inter team dependencies. Modules in automotive cannot be decoupled completely due to the described physical dependencies and the necessary integration and verification system. But standardized interfaces allow to channel coordination between responsible teams more efficiently. The effective **unit size** is also affected by the changes. While the overall project size remains unaffected, more compact module sizes allow for smaller teams per module. Still, standardized module interfaces allow a reduction of inter team cooperation efforts. While the total number of teams increases, the coordination determinant unit size decreases since less designers work in interdependent projects. In summary, the presented modularization strategy reduces the coordination determinants task dependency and unit size close to their original agile sweet spot levels. This reorganisation of the product architecture improves the suitability of agile coordination strategies to automotive application contexts.

6.2.4 Findings in response to research question three

In the introduction chapter automotive design is described as being object to an increasingly faster changing design environment. Conventional and established design practices are no longer able to answer to these dynamics. Agile design was developed to answer to a similar challenge in software design two decades ago. The objective of the thesis at hand is to determine if agile design can be a solution for automotive design despite the differences between both industries. Research question three asks for strategies to realize agility in automotive design that account for the fundamental differences: *How to enable agile product design in the automotive domain?*

The answer to research question three requires a linkage of the earlier findings. Chapter 4 shows that the reported benefits of agile methods are related to the respective coordination strategies. The lightweight agile coordination strategies ensure flexibility, efficiency, and efficacy in product design. Contrary to these findings chapter 5 summarizes negative experiences and problems of employed agile methods in automotive design. It emphasises that constraints of physicality and scale reduce the applicability of agile methods in automotive design. Both require multiteam design cooperation in contradiction to the original agile focus intra team cooperation.

The experienced problems are further differentiated in subchapter 6.1 with a focus on the dysfunctionality of the respective agile coordination strategies in automotive design. The findings clearly indicate that the constraints of scale and physicality are linked to coordination determinants in automotive that differ significantly from agile sweet spot conditions. The original balance of agile coordination modes and the respective set of coordination mechanisms does not match these automotive coordination determinants. More specifically, the employed agile coordination strategies lack inter team coordination mechanisms to apply to the multiteam design system. Furthermore, the connectivity of the agile coordination systems suffers which results in a decreasing self-adjustment ability of the coordination system. The coordination specific analysis allows to explain and differentiate the experienced loss in flexibility, efficiency, and efficacy with the dysfunctionality of agile coordination strategies in automotive design.

In subchapter 6.2 three distinct scenarios are discussed to re-establish agile coordination strategies and hence agility in automotive design. They respond directly to research question three. These scenarios differ regarding their basic approach to enable agile coordination strategies in automotive design. Scenario one intends to recreate agile coordination strategies in accordance with automotive coordination determinants. Agile inter team coordination mechanisms are introduced and the connectivity of the coordination system is re-established. Scenario two employs a new design technology to enable existing impersonal mode coordination mechanisms to match coordination determinants in automotive design. Additionally, the balance of agile coordination modes is reconfigured. Scenario three addresses automotive coordination determinants. Changes to the product architecture allow to adjust automotive coordination determinants to better match the original agile coordination strategies.

Table 12: Implemented and evaluated agile inter team coordination mechanisms throughout pilot projects.

Group mode	Individual mode	Impersonal mode	Boundary spanning	Cognitive mode
Scaled agile meeting	Instant messaging, video conferencing	Agile role hierarchy	Roadmap, System map	Agile role bearer team
Agile role bearer meeting	Team member rotation	Documentation, and information distribution system	Boundary spanning role	Specialized design team
Community of Practice		Automotive specific Definition of Done	Multiteam Backlog	Integrative design team

Scenario one addresses the lack of agile inter team coordination mechanisms in multiteam project settings in automotive design. It presents agile inter team coordination mechanisms which have been tested and evaluated throughout the agile pilot projects. Table 12 summarizes the implemented inter team coordination mechanism according to the respective coordination mode. The employed inter-team coordination mechanisms have been compared to similar practices from scaled agile methods such as SAFe and LeSS and to the current scientific literature of large-scale agile software development. Compared to agile sweet spot conditions automotive coordination determinants differ in the larger unit size and task dependency. The set of agile inter team coordination mechanisms has been evaluated and matched regarding their applicability on both unit size variations and task dependency variations in multiteam design system. Additionally, the connectivity of the new set of intra and inter team coordination mechanisms has been reaffirmed to ensure the flexibility and self-adjustment ability of the resulting agile coordination strategies. These characteristics are central to the capability of agile coordination strategies to adjust to changing project dynamics.

Scenario two employs Generative Design as a design technology to automatically and independently run iterative design cycles to generate 3D topologies according to prespecified objective functions and restrictions. From a coordination perspective this technology allows to realize continuous integration and testing coordination mechanisms across several teams. Unlike the original agile practices which include test driven development and requires designers to design and evaluate every iteration manually the new approach allows the system to run as many iterative design cycles as necessary to realize the desired output. Designers are only addressed if manual interaction is required, or interim results need evaluation. This implies that continuous and frequent coordination activities between different teams that were necessary to realize components cooperatively are no longer necessary, since the design algorithm integrates the relevant design objectives. Coordination efforts are reduced, and coordination efficiency increases significantly while coordination efficacy remains strong. Furthermore, the new design technology reduces necessary inter team coordination and hence allows to adjust the balance of coordination modes within agile coordination strategies from mutual adjustment towards impersonal mode coordination like the approach in software only design.

Unlike scenario one and two scenario three aims to change the coordination determinants in automotive design and not the coordination systems. Changes to the existing product modularization structure are implemented to match it with agile organization setups. Module size, necessary specializations for module design activities and module interdependencies are adjusted to reflect the core concepts of agility in design. Existing modularization methods are changed to reflect agile design as an additional modularization driver. The implemented change to the product modularization approximates automotive to agile sweet spot coordination determinants. This improves suitability of existing and proven agile coordination strategies in automotive design.

The three scenarios are presented in subchapters 6.2.1, 6.2.2, and 6.2.3 as independent and distinct concepts to enable agile coordination strategies and hence agility in automotive design. Nevertheless, they are not exclusive and may well be combined in practical applications to obtain optimal results. One paradox of product design as proposed by Paetzold et a. (Paetzold *et al.*, 2017) states that the same exact results of a development process is never expected twice, because development processes are different in each case. Therefore, a single ideal solution is improbable for a spectrum of slightly varying application contexts. Whereas a set of approaches allows to tailor a suiting solution for specific requirement. Furthermore, pilot projects indicate a beneficial mutual influence of scenarios one and three onto each other.

Regarding the core concepts of agility scenario one and two and to a minor degree scenario three considerably reduce the autonomy of self-organized teams to enable cooperation across teams as predicted by Bass and Haxby for large agile projects (Bass and Haxby, 2019). Even though, this implies a reduction of the agility of individual teams it increases the overall agility of multiteam design systems and therefore presents a possible optimum in automotive design. In such a system behaviour the benefits of autonomous, empowered teams are inferior to the benefits of alignment as questioned by Moe et al. (Moe *et al.*, 2016).

From a research perspective the presented scenarios one, two and three are iteratively designed research artifacts connecting the theoretical product design fields coordination theory and agile product design. These artifacts might well be analysed from different theoretical perspectives for further evaluation and borrowed to further empiric application fields.

7 Conclusion

"Science never solves a problem without creating ten more."
George Bernard Shaw

This chapter concludes the thesis by summarising the key research findings in relation to the research questions and discussing the value and contribution thereof. The aim of this thesis is to investigate and enable agility in automotive design contexts. An Action Research methodology was chosen to approach this aim. Agile product design methods and practices were employed, evaluated, and adjusted throughout eleven automotive design projects. The findings show that agile methods are object to constraints in the automotive domain. These constraints are driven by the physicality of the product and scale of the development process. Both factors result in multiteam design systems instead of independent autonomous teams and hence require inter team coordination mechanisms. The research highlights the efficiency of agile coordination strategies to explain the empirically proven benefits of agile methods. Nevertheless, original agile coordination strategies were adjusted to autonomous teams and therefore lack inter team coordination mechanisms necessary in automotive design. To realize agility in automotive design three adjustment scenarios of agile coordination strategies are introduced to cope with the domain specific coordination requirements.

The conclusion chapter is subdivided into five sections. In the first section, the findings are summarized in relation to the research questions. The second section defines the research contribution and emphasises what new knowledge has been added. The third section addresses research limitations, and the fourth section provides opportunities for future research. The last section concludes the thesis with a closing summary.

7.1 Response to the research questions

The **first research question "*How to explain agility and its benefits theoretically?*"** was approached by three sequential steps. First, the selection of an appropriate design theory that reflects agile design characteristics and functionality. Second, the development of a design theory grounded research model that suits the research aim, data availability, and the research project restrictions. Third, the evaluation and comparison of popular agile methods and their empirically reported benefits according to the research model.

Dingsøyr et al. stated in a review of agile design that the theory behind agile design is multifarious and a holistic explanation why agile works does not exist (Dingsøyr *et al.*, 2012). Different design theories have been applied successfully to explain specific aspects of agile design. Several studies rely on **coordination theory** to reflect on and explain agility in design (Dingsøyr, Bjørnson, *et al.*, 2018; Pries-Heje and Pries-Heje, 2011; Strode *et al.*, 2011). The four core concepts of agility involve and specify coordination between changing parties which underlines its importance (Baham and Hirschheim, 2021). The focus on personal exchange and communication in the Agile Manifesto for Software development (Fowler and Highsmith, 2001) underlines the relevance of coordination to realize agility in design. Coordination theory is also central to evaluate and enable inter team cooperation which is crucial in automotive design. More specifically, coordination theory enables to analyse and optimize the balance between team autonomy and system dynamics in agile multiteam systems. Additionally, coordination theory models respect dynamic system behaviour which allows to analyse how change and disturbances affect design systems. This ability is key to inflict change and understand its impacts in the chosen Action Research methodology. It also aides to explain the experienced constraints in agile automotive design. For these reasons coordination theory was chosen as the underlying design theory for this study.

The practical application of coordination theory in the research project required the development of a research model. This **coordination reference model** was shaped according to the coordination strategy concept (Li and Maedche, 2012; Strode *et al.*, 2012). The coordination strategy defines coordination determinants in relation to project characteristics and relates suitable coordination modes and respective mechanisms to realize coordination efficiency. Changes to the coordination determinants or the integration conditions lead to adjustments of the coordination mechanisms. The research model is based on the model of Van de Ven et al.

(Ven *et al.*, 1976) and follows its selection of the coordination determinants task uncertainty, task dependency and unit size. Still, alterations to the original model were added to better reflect agile design characteristics. Cognitive mode coordination (Espinosa *et al.*, 2004) was integrated to better reflect cooperation in close teams and boundary spanning coordination through boundary objects (Star and Griesemer, 1989) was added to explain the functionality of design artefacts in agile methods. Additionally, the model was designed to account for mutual connections between coordination mechanisms to analyse the system behaviour of interlinked agile practices. These adjustments enable the coordination reference model to analyse coordination strategies of agile methods. The deduced coordination strategies allow to decompose the system behaviour of the respective agile methods and explain reported benefits. Furthermore, the coordination reference model enables to explain dysfunctionalities in agile coordination strategies caused by external impacts or unsuitable coordination determinants. Both aspects are crucial to trace agile constraints.

The application context independent analysis of **agile coordination strategies** revealed characteristic patterns which reoccur across popular agile methods. They differ distinctively from conventional automotive design methods regarding their focus on undisturbed collaboration in autonomous, self-organized teams. The avoidance of external dependencies and the focus on intra team coordination results in excellent intra team coordination efficiency. This set-up ideally supports cognitive mode coordination mechanisms in daily cooperation. They require few explicit coordination activities and result in efficient, fast, and flexible coordination within teams. Besides implicit cognitive mode coordination mechanisms explicit group mode coordination mechanisms are emphasized to structure design projects. They provide efficacy, synchronization, knowledge exchange and learning in teams. Impersonal mode coordination mechanisms in the form of boundary objects are also complemented to ensure coordination efficiency. Some agile methods realize implicit impersonal mode coordination such as continuous integration and testing through the employment of respective IT systems. Unlike implicit cognitive mode coordination mechanisms, they are not restricted to small teams but also apply to multiteam systems.

Nevertheless, the deduced selection of coordination modes alone could not sufficiently explain the experienced benefits. The **connectivity between the coordination mechanisms** is crucial to explain the experienced benefits. It enables the system of coordination mechanisms to self-adjust to changing design requirements. It relies on several characteristics of agile methods. First, the compact length and iterative nature of the implemented design cycles allow to readjust the coordination settings according to changing project dynamics with every new iteration in short time lapses. Second, the practices in agile methods actuate and connect coordination mechanisms without manual adjustment. Third, the coordination mechanisms are designed to mutually activate each other answering to project dynamics. The composition of coordination mechanisms in agile coordination strategies is designed for a spectrum of requirements. The coordination strategy remains lightweight through changing projects dynamics since the adjustment is performed implicitly by straight forward design practices. This self-adjusting coordination system results in both very effective and efficient coordination in agile design projects.

The second research question "*What constraints reduce agile design applicability how in automotive design?*" provides context-specific empirical data and analyses it based on the established coordination reference model. It was addressed by two sequential steps. First, the collection and classification of context-specific problems that emerge if existing agile methods are employed in automotive design. This step included the comparison of the collected data to the established constraints of scale and physicality concepts. Second, the explanation of the experienced problems based on dysfunctionalities of agile coordination strategies in automotive design. Data collection was based on eleven agile pilot projects in automotive design contexts. These pilot projects were situated in typical application contexts to include a representative spectrum of automotive design requirements.

The bottom-up data analysis of the pilot projects showed that the employed agile methods face a repeating set of problems in automotive design. Numerous inter team dependencies, overwhelming numbers of stakeholders and demanding team composition in multiteam design systems, unclear task division, slow prototyping and incomplete product integration in short iterations were the most prominent among them. A corresponding top-down data analysis of the same data set matched the experienced problems to the agile constraints categories scale of design system and product physicality. Based on cause-effect relations both categories were evident in agile automotive design. Their impacts overlap and mainly cause inter team coordination problems in multiteam design systems. Therefore, in the thesis at hand constraints of physicality are viewed as an additional driver to the well-established constraints of scale category. This simplification of the problem space allowed to comprehensively analyse it with the coordination reference model.

The respective analysis identified dysfunctionalities of agile coordination strategies in automotive design that directly relate to the experienced problems. Coordination determinants in large-scale automotive design differ from agile sweet spot conditions. The number of involved teams increases the unit size level, while physical and procedural interdependencies drive the task interdependency level. The task uncertainty level remains invariant in automotive design compared to agile sweet spot conditions. These coordination determinants decrease the suitability of agile coordination mechanisms and respective coordination modes. Most affected are implicit coordination mechanisms. Implicit cognitive coordination mechanisms rely on close teamwork and are therefore inefficient for inter team coordination. The necessary technology for implicit impersonal coordination mechanisms such as continuous integration and testing is not able to manage the level of task interdependency in automotive design yet. Even though, the coordination reference model recommends impersonal mode coordination for larger unit sizes, agile coordination strategies lack the respective coordination mechanisms. Instead, agile lightweight impersonal mode coordination mechanisms such as boundary object mechanisms are overstrained by the number of parties and their opposing design objectives. The same applies to agile group mode coordination mechanisms. Agile coordination mechanisms were shaped to improve intra team coordination. They lack inter team coordination mechanisms able to deal with the task interdependency and unit size level in multiteam design systems.

The impact of automotive coordination determinants on agile coordination mechanisms also impairs the connectivity of the agile coordination system. The disturbed connection between coordination mechanisms decreases the ability of agile coordination strategies to self-adjust to project dynamics. The balance between group mode coordination mechanisms and boundary objects is overstrained due to number of different parties and the spectrum of specializations. Personal exchange necessary for cognitive mode coordination mechanisms decreases with ever larger meetings. Unlike group mode, individual mode coordination mechanisms remain functional in multiteam systems and therefore keep their ability to activate other coordination modes. Still, boundary spanning roles such as the Product Owner are affected by the larger network of inter team dependencies. The high task dependency level impairs the connection between group mode and impersonal mode coordination mechanisms. Boundary objects that structure meetings and get updated by the results are not suitable for the task dependency level anymore. With multiteam systems seriously affecting cognitive mode coordination this leaves only few cognitive mode coordination mechanisms connected to the coordination system.

Research question three asks for alternative approaches to introduce agile product design to automotive despite the earlier findings: ***How to enable agility in automotive product design?*** The research project generated three distinct scenarios to compensate the dysfunctionalities of agile coordination strategies in automotive design. While scenario one and two focus on the coordination system to match automotive coordination determinants, scenario three adjusts the product structure to reconfigure automotive coordination determinants to interlock with agile coordination strategies.

Scenario one adjusts agile coordination strategies to meet the coordination requirements of automotive design. The selection of agile coordination modes and respective mechanisms is reconfigured to match automotive coordination determinants. Agile inter team coordination mechanisms are introduced. These additional coordination mechanisms not only address the lack of inter team coordination but also re-establish the connectivity of the agile coordination system. Group mode inter team coordination is provided by scaled agile meetings, agile role bearer meetings and communities of practice. Individual mode inter team coordination relies on instant messaging, video conferencing and team member rotation to address the larger distances between teams. Impersonal mode inter team coordination mechanisms are agile roles hierarchies, automated documentation, information distribution systems, and a Definition of Done employment that reflects the complex verification process in automotive. Boundary spanning inter team coordination mechanisms are Product roadmaps, System maps, boundary spanning roles and multiteam backlogs. Cognitive mode inter team coordination mechanisms include agile role bearer teams, specialized design teams and integration design teams. These additional inter team coordination mechanisms re-establish agile coordination strategies in automotive multiteam design contexts. They have been tested and evaluated throughout the agile pilot projects. According to project settings and coordination requirements subsets of the presented inter team coordination mechanisms might be suitable.

Scenario two also adjusts agile coordination strategies to match automotive coordination determinants. But unlike scenario one the adjustment is not based on additional inter team coordination mechanisms. Instead, the impact of the new design technology Generative Design on agile coordination strategies in automotive is described. Generative Design allows to automatically and independently run iterative design cycles according to prespecified objective functions and restrictions. From a coordination perspective this technology realizes continuous integration and testing coordination mechanisms in multiteam design system. Dependencies between components are mostly managed within the digitalized design system. Designers from the responsible teams are only addressed if manual interaction is required, or interim results need evaluation. Therefore, the need for continuous and frequent coordination activities between teams to realize components cooperatively is reduced. Coordination efficiency increases significantly while coordination efficacy remains high. The Generative Design technology shifts the focus of agile coordination strategies from mutual adjustment towards impersonal mode coordination.

In contrast to scenario one and two, **scenario three** adjusts the automotive coordination determinants to re-establish existing agile coordination strategies. It changes the product modularization structure to match it with ideal agile organisation setups. While the overall product remains unchanged its decomposition into modules is reorganized. The average module size and the quantity of necessary specializations for module design activities are reduced to match the output of individual agile teams. Additionally, module decoupling is emphasized to increase team independency of responsible teams. These changes in the module structure recreate agile sweet spot conditions and the respective coordination determinants in automotive design which increase the suitability of existing agile coordination strategies.

The three scenarios are described independently but they do not exclude each other. Data from agile pilot projects indicate that especially scenarios one and three complement each other. Therefore, the scenarios might well be combined to improve their impact. Their combined introduction allows to realize agility in automotive design despite the much larger task dependency and unit size level compared to agile sweet spot conditions. The chosen coordination strategy approach is not constrained to a specific application domain.

7.2 Research contribution

The research contributes to the theoretical field of agile product design by its conceptualization of agility. It is based on two complementing perspectives. Agility as an attribute focuses on a conceptualization of agility. It characterizes the system behaviour and central traits of agility from an outside in perspective. Agility as a construct focuses on how to realize agility. It describes crucial elements and complete frameworks to realize agility in product design from an inside out perspective. The combination of both perspectives provides an unambiguous understanding of agility to avoid the Guru problem in practical applications and research.

The research includes rich case descriptions of agile design in automotive application contexts. It summarizes and categorizes repeating challenges of popular agile methods in the domain and grounds them theoretically. It proves the existence and interference of the established concepts agile constraints of scale and physicality within the application environment. The research compares and differentiates the cause-effect relations between both constraints and shows a strong overlap between their effects. This allows to position constraints of physicality as a subcategory of constraints of scale to simplify problem understanding for the selected research case. The connection of both constraints exemplarily shows how opposing characteristics of domains do not necessarily cause opposing constraints to agility. It underlines the broad applicability of agile design and avoids premature domain or product biased specific exclusions of agile product design.

The research employs coordination theory as theoretical lens to decompose and analyse agile methods to explain their empirically proven functionality and benefits. This proceed provides theoretical grounding to empirically validated design methods. While coordination theory has been employed in the software domain no comparable use is known to the author in the automotive domain. The research contributes a coordination reference model that is specifically designed to mirror agile design characteristics to allow for precise analyses.

Unlike earlier coordination models it connects selected theories from coordination research in the fields of organization research, team research and multiteam cooperation research. The coordination reference model allows to analyse individual agile practices and agile system behaviour in relation to the application domain based on the coordination strategy concept.

The coordination reference model employs the coordination strategy as theoretical concept to connect dynamically coupled application environment characteristics with coordination practices. This allows to connect experienced practical problems of agile design with flaws in the respective coordination strategies. It underlines the influence of the application context on the coordination strategy. The concept accounts for dynamic changes in the balance and hence represents the core benefits of agility: its adjustability and flexibility in direct relation to project dynamics. The approach allows to adjust both agile design practices and or characteristics of the application context to enable agility in automotive design. For the automotive domain several scenarios to avoid agile constraints are presented. This proceed is not limited to automotive design but opens opportunities to expand agile design to further domains.

7.3 Research limitations

The findings of this study have to be seen in light of some limitations. This section announces and reflects on those limitations. In the research outlook recommendations how to avoid the described limitations are supplemented.

The first limitation set concerns the novelty of the research area as a recent phenomenon which impairs the design of the research aim and the respective research questions. The selected research area agile automotive design lacks previous secondary literature. The existing publications in the automotive domain consist mainly of practice-oriented experience reports. Comprehensive literature reviews are missing, and few publications include theoretical grounding. Unlike the automotive domain agile software development is well researched and based on a rich research stream. This imbalance towards software development translates into a software bias in the research community which translates into the work at hand.

The lack of literature specific to the research phenomenon was addressed by comparably broad research questions that address the complete domain automotive design and the functionality of agile design in general. More narrow research questions would have facilitated data collection and improved findings sensitivity. In retrospect a focus on the early phase of automotive design would have resulted in more conclusive data sets. Furthermore, the research questions do not account for the significant change within the domain. Even within the limited time horizon of the research project the transformation from conventional mechatronic design towards software or software-alike design was evident and had a relevant influence on the results. Regarding the chosen coordination reference model, the given coordination determinant Unit Size did not adequately reflect the multiteam design systems in the automotive domain.

The second limitation set reflects the chosen research approach. The selected Action Research methodology is beneficial for parallel problem understanding and practice relevant solution design. This advantage in research on novel research phenomena from practice comes with drawbacks compared to more precise and adjustable research methods. It impairs the realized research depth in particular aspects (e.g. the sensitivity of the efficiency measurement of coordination practices or the comparability of the pilot projects). Additionally, Action Research data sets are context and case specific which complicates the generalization of the findings from specific pilot projects to the complete domain. Still, typical Action Research methodology based limitations such as a strong researcher bias or focusing on action and neglecting research and hence being little more than consultancy (Avison and Wood-Harper, 1991) were addressed through the balanced research approach. The coordination theory grounded data analysis as well as the systematic and comprehensive data collection ensured validity of the data and rigour in the methodology which are often criticised in Action Research (Baskerville and Wood-Harper, 1996).

The research is also affected by a sample bias since the selection of the pilot projects was influenced by research unspecific requirements such as irrevocable management wishes. Furthermore, the set of eleven pilot projects does not reflect the comprehensive activities in automotive design and therefore only realizes a partial research phenomenon coverage. Generalizing results to the automotive domain based on the Action Research pilot projects was only possible with limitations. Therefore, scenarios instead of generally applicable frameworks were presented in the discussion. Basing the study in larger sample size could have generated more accurate results. But the necessary effort for pilot project support impeded a larger set of cases. Furthermore, the research was limited to one company which limits the validity to the complete domain.

Lastly, the scope of the discussion suffers from incomplete scenario evaluation. While scenario one and two were evaluated to a certain degree in practice, scenario three remains a theoretical construct based on applicable theories and comparable cases from different contexts. These limitations of the discussion were also driven by the restricted time horizon of the research project.

7.4 Further work

The thesis at hand opens several opportunities for future research. Regarding the presented scenarios to overcome agile constraints in the automotive domain the theoretical scenario three opens the opportunity for comprehensive practical validation in automotive design. Little research has been published that connects product modularization with the applicability of agile design throughout the design process. Additionally, the combination of all three scenarios should lead to interesting findings and help to evaluate the scenarios in dimensions ranging from ease of application towards practical benefits to reduce agile constraints.

The current set of the pilot projects leaves research opportunities to evaluate the findings in other automotive design contexts and throughout later phases of automotive design. Large and continuous projects provide the chance to increase the relevance of the findings to complete automotive design projects. Since the presented research project was limited to one partnering automotive OEM further validations with other OEMs or TIER 1 supplier will increase the significance of the findings. Further research opportunities are not restricted to the automotive domain. The design of the coordination reference model and the scenarios to avoid agile constraints applies to other large-scale hardware product domains. This opens the possibility to transfer the findings to a spectrum of industries that face similar challenges of increasing change driven by global phenomena such as digitalization. Even in software development the application of the coordination reference model could be studied in comparison to existing coordination models to address the experienced challenges of scale in large software development programs.

Regarding the second research limitation set due to the selected research methodology the application of more standardized qualitative methods such as case studies or the application of quantitative research methods such as surveys might result in additional and more precise findings. Furthermore, research variables such as confirmed efficiency gains with partnering companies could improve research acceptance and its transferability to practice. Especially a precise definition and measurement of the efficiency of the employed coordination strategies would allow a better selection of suitable inter team coordination mechanisms for multiteam design context conditions.

7.5 Closing summary

Within this conclusion chapter a brief summary of the PhD thesis at hand was given. It included the original research aim, the selected research questions and the chosen research methodology. The main findings were presented in relation to the respective research questions to show the consistency of the research project. Three distinct scenarios to realize agility in automotive design despite the verified constraints to agility in this domain were presented. Additionally, limitations of the research were announced and reflected and opportunities for future research were presented. To finalize this PhD thesis the author has selected a quote to underline both the inherent incompleteness of research and the necessity to conclude anyways.

"A great dissertation is a finished dissertation"
Ancient Grad Student Proverb

Acronyms

APD	Agile Product Development/Design
COP	Community of practice
PD	Product Development
PO	Product Owner
SM	Scrum Master
XP	Extreme Programming
SAFe	Scaled Agile Frameworks
LeSS	Large Scale Scrum
OEM	Original Equipment Manufacturer
TIER1 supplier	First level supplier

List of figures

List of tables

References

Abbas, N., Gravell, A.M. and Willis, G.B. (2008), “Historical roots of agile methods: Where did ‘Agile thinking’ come from?”, *International Conference on Agile Processes and Extreme Programming in Software Engineering.*, Springer, Berlin.

Abrahamsson, A.P., Salo, O. and Ronkainen, J. (2002), “Agile Software Development Methods : Review and Analysis”, *VTT Publication*.

Agile Alliance. (2021), “Agile 101”, available at: https://www.agilealliance.org/agile101/.

AgileAlliance. (2021a), “Scrum”, available at: https://www.agilealliance.org/glossary/scrum.

AgileAlliance. (2021b), “XP”, available at: https://www.agilealliance.org/glossary/xp.

Aken, J.E. van. (2004), “Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules”, *Journal of Management Studies*, Wiley Online Library, Vol. 41 No. 2, pp. 219–246.

Allen, T.J. (1977), “Managing the Flow of Technology”, *MIT Press Cambridge MA*.

Alqudah, M. and Razali, R. (2016), “A Review of Scaling Agile Methods in Large Software Development”, *International Journal on Advanced Science, Engineering and Information Technology*, Vol. 6 No. 6, p. 828.

Alvesson, M. and Sandberg, J. (2011), “Generating research questions through problematization”, *Academy of Management Review*, Academy of Management Briarcliff Manor, NY, Vol. 36 No. 2, pp. 247–271.

Atzberger, A., Nicklas, S., Schrof, J., Weiss, S. and Paetzold, K. (2020), *Agile Entwicklung Physischer Produkte - Eine Studie Zum Aktuellen Stand Der Industriellen Praxis*, Munich, available at:https://doi.org/10.18726/2020_5.

Atzberger, A. and Paetzold, K. (2019), “Current challenges of agile hardware development: What are still the pain points nowadays?”, *Proceedings of the International Conference on Engineering Design, ICED*, Vol. 2019-Augus No. AUGUST, pp. 2209–2218.

Atzberger, A., Simon, N., Schrof, J., Weiss, S. and Paetzold, K. (2020), *Agile Entwicklung Physischer Produkte: Eine Studie Zum Aktuellen Stand in Der Industriellen Praxis*, Universität der Bundeswehr München, Neubiberg, Munich, available at: https://athene-forschung.unibw.de/doc/133438/133438.pdf.

Avison, D.E. and Wood-Harper, A.T. (1991), “Conclusions from Action Research: The Multiview Experience”, *Systems Thinking in Europe*, Springer US, Boston, MA, pp. 591–596.

Babüroglu, O.N. and Ravn, I. (1992), “Normative action research”, *Organization Studies*, SAGE Publications Sage UK: London, England, Vol. 13 No. 1, pp. 19–34.

Baham, C. and Hirschheim, R. (2021), “Issues, challenges, and a proposed theoretical core of agile software development research”, *Information Systems Journal*, p. isj.12336.

Ballard, D.I. and Seibold, D.R. (2003), “Communicating And Organizing In Time”, *Management Communication Quarterly*, Vol. 16 No. 3, pp. 380–415.

Baltes, G. and Selig, C. (2017), “Organisationale Veränderungsintelligenz – Wachstumsfähigkeit mit strategischer Innovation erneuern”, *Veränderungsintelligenz*, available at:https://doi.org/10.1007/978-3-658-04889-1_2.

Banks, J.A. and Hughes, E.C. (1959), “Men and Their Work”, *The British Journal of Sociology*, available at:https://doi.org/10.2307/587716.

Barlow, J., Giboney, J.S., Keith, M.J., Wilson, D.W., Schuetzler, R.M., Lowry, P.B. and Vance, A. (2011), “Overview and Guidance on Agile Development in Large Organizations Overview and Guidance on Agile Development in Large Organizations”, *Communications of the ASsociation for Information Systems*, Vol. 29 No. July 2011, pp. 25–44.

Baron, P. and Hüttermann, M. (2010), *Fragile Agile: Agile Softwareentwicklung Richtig Verstehen Und Leben*, Hanser Verlag.

Baskerville and Myers. (2004), “Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice: Foreword”, *MIS Quarterly*, Vol. 28 No. 3, p. 329.

Baskerville, R. and Pries-Heje, J. (1999), “Grounded action research: A method for understanding IT in practice”, *Accounting, Management and Information Technologies*, Vol. 9 No. 1, pp. 1–23.

Baskerville, R.L. and Wood-Harper, A.T. (1996), “A critical perspective on action research as a method for information systems research”, *Journal of Information Technology*, available at:https://doi.org/10.1080/026839696345289.

Bass, J.M. (2015), “How product owner teams scale agile methods to large distributed enterprises”, *Empirical Software Engineering*, Vol. 20 No. 6, pp. 1525–1557.

Bass, J.M. and Haxby, A. (2019), “Tailoring Product Ownership in Large-Scale Agile Projects: Managing Scale, Distance, and Governance”, *IEEE Software*, available at:https://doi.org/10.1109/MS.2018.2885524.

Bassey, M. (1990), *ON THE NATURE OF RESEARCH IN EDUCATION.*, HOLLINSHEAD, A. ED2280/ED3843.

Bechky, B.A. (2006), “Gaffers, gofers, and grips: Role-based coordination in temporary organizations”, *Organization Science*, available at:https://doi.org/10.1287/orsc.1050.0149.

Beck, K. (1999), *Extreme Programming Explained: Embrace Change*, *XP Series*.

Beck, K. (2003), *Test-Driven Development: By Example*, *The Journal of Object Technology*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, United States, available at:https://doi.org/10.5381/jot.2003.2.2.r1.

Beck, K. (2004), *Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series)*, Vol. 2.

Beck, K. and Beedle, M. (2001), “Manifesto for Agile Software Development”, available at: http://agilemanifesto.org/history.html.

Benbasat, I. and Zmud, R.W. (1999), “Empirical research in information systems: The practice of relevance”, *MIS Quarterly: Management Information Systems*, Vol. 23 No. 1, pp. 3–16.

Bendsoe, M.P. and Sigmund, O. (2004), *Topology Optimization. Theory, Methods and Applications.*, Springer-Vlg., Bln., Heidelberg.

Bennett, N. and Lemoine, G.J. (2015), “What a difference a word makes : Understanding threats to performance in a VUCA world”, *Business Horizons*, “Kelley School of Business, Indiana University”, p. 8.

Bentley, C. (2005), *PRINCE2 Revealed*, Routledge, available at:https://doi.org/10.4324/9780080458823.

Berger, C. and Eklund, U. (2015), “Expectations and Challenges from Scaling Agile in Mechatronics-Driven Companies – A Comparative Case Study”, in Lassenius, C., Dingsøyr, T. and Paasivaara, M. (Eds.), *Lecture Notes in Business Information Processing*, Vol. 212, Springer International Publishing, Cham, pp. 15–26.

Bergman, M., Mark, G. and Lyytinen, K. (2007), “Boundary Objects in Design: An Ecological View of Design Artifacts”, *Journal of the Association for Information Systems*, Vol. 8 No. 11, pp. 546–568.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A. and Heinzl, A. (2018), “Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings”, *IEEE Transactions on Software Engineering*, available at:https://doi.org/10.1109/TSE.2017.2730870.

Blessing, L.T.M. and Chakrabarti, A. (2009), *DRM, a Design Research Methodology*, *ICED 1995: International Conference on Engineering Design*, Springer London, London, available at:https://doi.org/10.1007/978-1-84882-587-1.

Boehm, B. (2002), “Get ready for agile methods, with care”, *Computer*, Vol. 35 No. 1, pp. 64–69.

Boehm, B. and Turner, R. (2004), “Balancing agility and discipline: evaluating and integrating agile and plan-driven methods”, *Proceedings. 26th International Conference on Software Engineering*, IEEE Comput. Soc, pp. 718–719.

Boehm, B.W. (1988), “A spiral model of software development and enhancement”, *Computer*, Vol. 21 No. 5, pp. 61–72.

Boell, S.K. and Cecez-Kecmanovic, D. (2015), “On being ‘Systematic’ in Literature Reviews in IS”, *Journal of Information Technology*, Vol. 30 No. 2, pp. 161–173.

Bogner, A., Littig, B. and Menz, W. (2014), *Qualitative Sozialforschung*, *Qualitative Sozialforschung*, Oldenbourg Wissenschaftsverlag Verlag, München, available at:https://doi.org/10.1524/9783486717594.

Boote, D.N. and Beile, P. (2005), “Scholars before researchers: On the centrality of the dissertation literature review in research preparation”, *Educational Researcher*, Sage Publications Sage CA: Thousand Oaks, CA, Vol. 34 No. 6, pp. 3–15.

Bowman, I. and Holt, R. (1998), “Software Architecture Recovery Using Conway’s Law”, *Conference of the Centre for Advanced Studies on Collaborative Research*.

Briers, M. and Wai, F.C. (2001), “The role of actor-networks and boundary objects in management accounting change: A field study of an implementation of activity-based costing”, *Accounting, Organizations and Society*, available at:https://doi.org/10.1016/S0361-3682(00)00029-5.

Brown, T.E. (2013), “Skunk works: a sign of failure, a sign of hope?”, *Innovation, Entrepreneurship and Culture*, Edward Elgar Publishing, available at:https://doi.org/10.4337/9781845420550.00012.

Burrel, G. and Morgan, G. (1979), *Sociological Paradigms and Organizational Analysis*, Heineman.

Burris, B.H. and Henderson, K. (2001), “On Line and On Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering”, *Contemporary Sociology*, Vol. 30 No. 1, p. 38.

Campanelli, A.S. and Parreiras, F.S. (2015), “Agile methods tailoring - A systematic literature review”, *Journal of Systems and Software*, available at:https://doi.org/10.1016/j.jss.2015.08.035.

Cannon-Bowers, J.A. and Salas, E. (2001), “Reflections on shared cognition”, *Journal of Organizational Behavior*, available at:https://doi.org/10.1002/job.82.

Cannon-Bowers, J.A., Salas, E. and Converse, S. (1993), “Shared mental models in expert team decision making”, *Individual and Group Decision Making*.

Causevic, A., Sundmark, D. and Punnekkat, S. (2011), “Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review”, *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, IEEE, pp. 337–346.

Checkland, P. and Holwell, S. (1998), “Action Research: Its Nature and Validity”, *Systemic Practice and Action Research*, Vol. 11 No. 1, pp. 9–21.

Cockburn, A. (2006), *Agile Software Development: The Cooperative Game*, Pearson Eduction.

Cohn, M. (2010), *Succeeding with Agile: Software Development Using Scrum*, Pearson Eduction.

Combs, J.P., Bustamante, R.M. and Onwuegbuzie, A.J. (2010), “An interactive model for facilitating development of literature reviews”, *International Journal of Multiple Research Approaches*, Taylor & Francis, Vol. 4 No. 2, pp. 159–182.

Conboy, K. (2009), “Agility from First Principles : Reconstructing the Concept of Agility in Information Systems Development”, *Information Systems Research*, Vol. 20(3) No. August 2018, pp. 329–354.

Conforto, E.C., Amaral, D.C., da Silva, S.L., Di Felippo, A. and Kamikawachi, D.S.L. (2016), “The agility construct on project management theory”, *International Journal of Project Management*, Vol. 34 No. 4, pp. 660–674.

Conforto, E.C., Salum, F., Amaral, D.C., da Silva, S.L. and de Almeida, L.F.M. (2014), “Can Agile Project Management be Adopted by Industries Other than Software Development?”, *Project Management Journal*, Vol. 45 No. 3, pp. 21–34.

Conway, M.E. (1968), “How do committees invent”, *Datamation*, Vol. 14 No. 4, pp. 28–31.

Cooper, R.G. (1983), “The new product process: an empirically-based classification scheme”, *R&D Management*, Vol. 13 No. 1, pp. 1–13.

Cooper, R.G. (1990), “Stage-Gate Systems: A new Tool for Managing New Products”, No. June, available at:https://doi.org/10.1016/0007-6813(90)90040-I.

Cramton, C.D. (2001), “The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration”, *Organization Science*, available at:https://doi.org/10.1287/orsc.12.3.346.10098.

Cronen, V.E. (2001), “Practical theory, practical art, and the pragmatic-systemic account of inquiry”, *Communication Theory*, Oxford University Press, Vol. 11 No. 1, pp. 14–35.

Crowston, K., Rubleske, J. and Howison, J. (2006), “Coordination theory: A ten-year retrospective”, *Human-Computer Interaction and Management Information Systems*, available at:https://doi.org/ISBN: 0765614871.

Davison, R., Martinsons, M.G. and Kock, N. (2004), “Principles of canonical action research”, *Information Systems Journal*, Wiley Online Library, Vol. 14 No. 1, pp. 65–86.

Denning, S. (2012), “How Agile can transform manufacturing: the case of Wikispeed”, *Strategy & Leadership*, Vol. 40 No. 6, pp. 22–28.

Denning, S. (2016), “Explaining Agile”, *Forbes*, available at: https://www.forbes.com/sites/stevedenning/2016/09/08/explaining-agile/?sh=7c9e0416301b.

Denning, S. (2017), “The age of Agile”, *Strategy and Leadership*, available at:https://doi.org/10.1108/SL-12-2016-0086.

Denning, S. (2020), “Why And How Volvo Embraces Agile At Scale”, *Forbes*, available at: https://www.forbes.com/sites/stevedenning/2020/01/26/how-volvo-embraces-agile-at-scale/?sh=3d4ee9f94cf0.

Díaz, C.E.G. (2011), “Vehicle Architecture and Lifecycle Cost Analysis: In a New Age of Architectural Competition”, Universitätsbibliothek der TU München.

Dietrich, P., Kujala, J. and Artto, K. (2013), “Inter-Team Coordination Patterns and Outcomes in Multi-Team Projects”, *Project Management Journal*, Vol. 44 No. 6, pp. 6–19.

Dikert, K., Paasivaara, M. and Lassenius, C. (2016), “Challenges and success factors for large-scale agile transformations: A systematic literature review”, *Journal of Systems and Software*, Elsevier Inc., Vol. 119, pp. 87–108.

Dingsoeyr, T., Falessi, D. and Power, K. (2019), “Agile Development at Scale: The Next Frontier”, *IEEE Software*, Vol. 36 No. 2, pp. 30–38.

Dingsøyr, T., Bjørnson, F.O., Moe, N.B., Rolland, K. and Seim, E.A. (2018), “Rethinking coordination in large-scale software development”, *Proceedings - International Conference on Software Engineering*, pp. 91–92.

Dingsøyr, T., Fægri, T.E. and Itkonen, J. (2014), “What is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development”, in Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J. and Raatikainen, M. (Eds.), *Product-Focused Software Process Improvement*, Vol. 8892, Springer International Publishing, Cham, pp. 273–276.

Dingsøyr, T., Moe, N.B., Fægri, T.E. and Seim, E.A. (2018), “Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation”, *Empirical Software Engineering*, Vol. 23 No. 1, pp. 490–520.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2012), “A decade of agile methodologies: Towards explaining agile software development”, *Journal of Systems and Software*, Vol. 85 No. 6, pp. 1213–1221.

Dingsøyr, T., Olav, B.F., Schrof, J. and Sporsem, T. (2021), “Transitioning from a first- to a second-generation large-scale agile development method: A longitudinal explanatory case study of coordination in a very large development programme", *Empirical Software Engineering*.

Dingsøyr, T., Rolland, K., Moe, N.B. and Seim, E.A. (2017), "Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development", *Procedia Computer Science*, Vol. 121, pp. 123–128.

Donaldson, L. (2001), *The Contingency Theory of Organizations*, *The Contingency Theory of Organizations*, SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States, available at:https://doi.org/10.4135/9781452229249.

Douglass, B.P. (2016), "What Are Agile Methods and Why Should I Care?", *Agile Systems Engineering*, Elsevier, pp. 41–84.

Ebel, B. and Hofer, M.B. (2014), "Automotive Management: Strategie und Marketing in der Automobilwirtschaft: Springer Berlin Heidelberg", *Online Verfügbar Unter Https://Books. Google. de/Books*.

Eden, C. and Huxham, C. (1996), "Action research for the study of organizations.", *Handbook of Organization Studies*, pp. 526–542.

Edison, H., Wang, X. and Conboy, K. (2021), "Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review", *IEEE Transactions on Software Engineering*, pp. 1–1.

Ehrlenspiel, K. and Meerkamm, H. (2013), *Integrierte Produktentwicklung – Denkabläufe, Methodeneinsatz, Zusammenarbeit.*, Carl Hanser Verlag GmbH Co KG.

Eigner, M. (2021), "Engineering 4.0–Grundlagen der Digitalisierung des Engineerings", *System Lifecycle Management*, Springer, pp. 29–104.

Eklund, U. and Berger, C. (2017), "Scaling agile development in mechatronic organizations - A comparative case study", *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, pp. 173–182.

Erickson, J., Lyytinen, K. and Siau, K. (2005), "Agile modeling, agile software development, and extreme programming: The state of research", *Journal of Database Management*, Vol. 16 No. 4, p. 88.

Espinosa, J.A., Armour, F. and Boh, W.F. (2010), "Coordination in enterprise architecting: An interview study", *Proceedings of the Annual Hawaii International Conference on System Sciences*, available at:https://doi.org/10.1109/HICSS.2010.450.

Espinosa, J.A., Lerch, F.J. and Kraut, R.E. (2004), "Explicit versus implicit coordination mechanisms and task dependencies: One size does not fit all.", *Team Cognition: Understanding the Factors That Drive Process and Performance.*, American Psychological Association, Washington, pp. 107–129.

Espinosa, J.A., Slaughter, S.A., Kraut, R.E. and Herbsleb, J.D. (2007), "Team knowledge and coordination in geographically distributed software development", *Journal of Management Information Systems*, available at:https://doi.org/10.2753/MIS0742-1222240104.

Faraj, S. and Sproull, L. (2000), "Coordinating expertise in software development teams", *Management Science*, available at:https://doi.org/10.1287/mnsc.46.12.1554.12072.

Faraj, S. and Xiao, Y. (2006), "Coordination in fast-response organizations", *Management Science*, available at:https://doi.org/10.1287/mnsc.1060.0526.

Fayol, H. (1949), "General and Industrial Management", *English Translation*, Pitman Publishing Company, London.

Feak, C.B. and Swales, J.M. (2009), *Telling a Research Story: Writing a Literature Review*, University of Michigan Press.

Feldman, M.S. (2000), "Organizational Routines as a Source of Continuous Change", *Organization Science*, available at:https://doi.org/10.1287/orsc.11.6.611.12529.

Firth, B.M., Hollenbeck, J.R., Miles, J.E., Ilgen, D.R. and Barnes, C.M. (2015), "Same Page, Different Books: Extending Representational Gaps Theory to Enhance Performance in Multiteam Systems", *Academy of*

Management Journal, Vol. 58 No. 3, pp. 813–835.

Fowler, M. and Highsmith, J. (2001), *The Agile Manifesto*.

Friedman, K. (2003), “Theory construction in design research: criteria: approaches, and methods”, *Design Studies*, Elsevier, Vol. 24 No. 6, pp. 507–522.

Furuhjelm, J., Segertoft, J., Justice, J. and Sutherland, J.J. (2017), “Owning the Sky with agile: Building a Jet Fighter Faster, Cheaper, Better with Scrum”, pp. 1–4.

Galbrath, J.R. (1973), *Designing Complex Organizations*, Addison-Wesley, Reading, MA.

Gausemeier, J., Michels, J.S., Orlik, L. and Redenius, A. (2004), “Modellierung und Planung von Produktentstehungsprozessen”, *VDI Berichte*.

Gilbert, K. (2000), *The Emotional Nature of Qualitative Research*, CRC Press.

Gittell, J.H. (2000), “Paradox of Coordination and Control”, *California Management Review*, Vol. 42 No. 3, pp. 101–117.

Goldkuhl, G. and Lind, M. (2010), “A multi-grounded design research process”, *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6105 LNCS, pp. 45–60.

Göpfert, J. (1998), *Modulare Produktentwicklung*, *Modulare Produktentwicklung*, Deutscher Universitätsverlag, Wiesbaden, available at:https://doi.org/10.1007/978-3-663-08152-4.

Göpfert, J. and Steinbrecher, M. (2000), “Modulare Produktentwicklung leistet mehr”, *Harvard Business Review*, Vol. 3 No. 089, pp. 1–17.

Greenhalgh, T. and Taylor, R. (1997), “How to read a paper: Papers that go beyond numbers (qualitative research)”, *BMJ*, Vol. 315 No. 7110, pp. 740–743.

Gregory, P., Barroca, L., Taylor, K., Salah, D. and Sharp, H. (2015), “Agile challenges in practice: A thematic analysis”, *Lecture Notes in Business Information Processing*, available at:https://doi.org/10.1007/978-3-319-18612-2_6.

Grewal, H. and Maurer, F. (2007), “Scaling Agile Methodologies for Developing a Production Accounting System for the Oil &amp; Gas Industry”, *AGILE 2007 (AGILE 2007)*, IEEE, pp. 309–315.

Gustavsson, T. (2016), “Benefits of agile project management in a non-software development context - A literature review”, *Project Management Development – Practice and Perspectives: Fifth International Scientific Conference on Project Management in the Baltic Contries April 14-15, Riga, University of Latvia*.

Gustavsson, T. (2020a), “Dynamics of inter-team coordination routines in large-scale agile software development”, *27th European Conference on Information Systems - Information Systems for a Sharing Society, ECIS 2019*.

Gustavsson, T. (2020b), *Inter-Team Coordination in Large-Scale Agile Software Development Projects*, Karlstad University Studies, available at:https://doi.org/978-91-7867-155-7.

Haberfellner, R. and de Weck, O. (2005), “Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering”, *International Council On Systems Engineering (INCOSE)*, Vol. 15, pp. 1449–1465.

Hage, J., Aiken, M. and Marrett, C.B. (1971), “Organization Structure and Communications”, *American Sociological Review*, available at:https://doi.org/10.2307/2093672.

Hammer, M. (2001), “Seven insights about processes”, *Conference on Strategic Power Process Ensuring Survival Creating Competitive Advantage*, Boston, MA.

Hatch, N.W., Baldwin, C.Y. and Clark, K.B. (2001), “Design Rules, Volume 1: The Power of Modularity”, *The Academy of Management Review*, Vol. 26 No. 1, p. 130.

Hellenbrand, D. (2013), “Transdisziplinäre Planung und Synchronisation mechatronischer Produktentwicklungsprozesse”, Technische Universität München.

Henderson-Sellers, B. and Serour, M.K. (2005), “Creating a dual-agility method: The value of method engineering”, *Journal of Database Management*, available at:https://doi.org/10.4018/jdm.2005100101.

Henderson, R.M. and Clark, K.B. (1990), “Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms”, *Administrative Science Quarterly*, Vol. 35 No. 1, p. 9.

Hensel, T. (2011), *Modellbasierter Entwicklungsprozess Für Automatisierungslösungen*, Vol. 258, Herbert Utz Verlag.

Hevner Alan, R. (2007), “A Three Cycle View of Design Science Research”, *Scandinavian Journal of Information Systems*, Vol. 19 No. 2, pp. 87–92.

Hevner, March, Park and Ram. (2004), “Design Science in Information Systems Research”, *MIS Quarterly*, Vol. 28 No. 1, p. 75.

Highsmith, J. (2009), *Agile Project Management: Creating Innovative Products*, Pearson Eduction.

Highsmith, J. and Cockburn, A. (2001), “Agile software development: the business of innovation”, *Computer*, Vol. 34 No. 9, pp. 120–127.

Hoda, R., Salleh, N. and Grundy, J. (2018), “The Rise and Evolution of Agile Software Development”, *IEEE Software*, Vol. 35 No. 5, pp. 58–63.

Hoegl, M. and Gemuenden, H.G. (2001), “Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence”, *Organization Science*, available at:https://doi.org/10.1287/orsc.12.4.435.10635.

Hoegl, M., Weinkauf, K. and Gemuenden, H.G. (2004), “Interteam Coordination, Project Commitment, and Teamwork in Multiteam R&D Projects: A Longitudinal Study”, *Organization Science*, available at:https://doi.org/10.1287/orsc.1030.0053.

Hohl, P., Münch, J., Schneider, K. and Stupperich, M. (2016), “Forces that Prevent Agile Adoption in the Automotive Domain”, *Product-Focused Software Process Improvement (PROFES)*, Vol. 10027, pp. 468–476.

Holmstrom, H., Conchúir, E.Ó., Ågerfalk, P.J. and Fitzgerald, B. (2006), “Global software development challenges: A case study on temporal, geographical and socio-cultural distance”, *Proceedings - 2006 IEEE International Conference on Global Software Engineering, ICGSE 2006*, available at:https://doi.org/10.1109/ICGSE.2006.261210.

Iivari, J. (2007), “Scandinavian Journal of Information A Paradigmatic Analysis of Information Systems As a Design Science A Paradigmatic Analysis of Information”, *Scandinavian Journal of Information Systems © Scandinavian Journal of Information Systems*, Vol. 19 No. 2, pp. 39–64.

Iivari, J., Hirschheim, R. and Klein, H.K. (1998), “A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies”, *Information Systems Research*, Vol. 9 No. 2, pp. 164–193.

Janes, A. and Succi, G. (2012), “The dark side of agile software development”, *SPLASH 2012: Onward! 2012 - Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pp. 215–227.

Jarzabkowski, P.A., Lê, J.K. and Feldman, M.S. (2012), “Toward a theory of coordinating: Creating coordinating mechanisms in practice”, *Organization Science*, Vol. 23 No. 4, pp. 907–927.

Kahkonen, T. (2004), “Agile Methods for Large Organizations - Building Communities of Practice”, *Agile Development Conference*, IEEE, pp. 2–11.

Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Könnölä, K., Mäkilä, T. and Lehtonen, T. (2013), “Agile methods for embedded systems development - a literature review and a mapping study”, *EURASIP Journal on Embedded Systems*, Vol. 2013 No. 1, available at:https://doi.org/10.1186/1687-3963-2013-15.

Kang, H.R., Yang, H.D. and Rowley, C. (2006), “Factors in team effectiveness: Cognitive and demographic similarities of software development team members”, *Human Relations*, available at:https://doi.org/10.1177/0018726706072891.

Kazanjian, R.K., Drazin, R. and Glynn, M.A. (2000), “Creativity and technological learning: The roles of organization architecture and crisis in large-scale projects”, *Journal of Engineering and Technology Management - JET-M*, available at:https://doi.org/10.1016/S0923-4748(00)00026-6.

Kerzner, H.R. (2009), *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 11th ed., New Jersey: Wiley.

Kniberg, H. (2014a), “Spotify Engineering Culture (part 1)”, *Spotify Labs*.

Kniberg, H. (2014b), “Spotify engineering culture (part 2)”, *Spotify Labs*.

Komus, A. (2017), *Abschlussbericht: Status Quo Agile 2016/2017.*

Komus, A. and Kuberg, M. (2020), *Status Quo (Scaled) Agile 2019/20*, *4. Internationale Studie Zu Nutzen Und Erfolgsfaktoren (Skalierter) Agiler Ansätze*.

Kortus-Schultes, D., Laufner, W., Hadry, A., Hasler, D., Markes, N., Powalka, V. and Stähler, L. (2014), “Das Auto als Smartphone: Konvergenz von Geschäftsmodellen der Automobil-Hersteller und der Telekommunikationsanbieter”, *Radikale Innovationen in Der Mobilität*, Springer Fachmedien Wiesbaden, Wiesbaden, pp. 117–142.

Krause, D. and Gebhardt, N. (2018), *Methodische Entwicklung Modularer Produktfamilien*, Springer Berlin Heidelberg, Berlin, Heidelberg, available at:https://doi.org/10.1007/978-3-662-53040-5.

Krauss, R.M. and Fussel, S.R. (1990), “Mutual Knowledge and Communicative Effectiveness”, *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*.

Kraut, R.E. and Streeter, L.A. (1995), “Coordination in software development”, *Communications of the ACM*, Vol. 38 No. 3, p. 69+.

Kruchten, P. (2013), “Contextualizing agile software development”, *Journal of Software: Evolution and Process*, Vol. 25 No. 4, pp. 351–361.

Kurtz, C.F. and Snowden, D.J. (2003), “The new dynamics of strategy: sense-making in a complex and complicated world”, *IEEE Engineering Management Review*, Vol. 31 No. 4, pp. 110–110.

Kvale, S. and Brinkmann, S. (2009), *Interviews: Learning the Craft of Qualitative Research*, *California, US: SAGE*.

Laanti, M., Similä, J. and Abrahamsson, P. (2013), “Definitions of Agile Software Development and Agility”, *Communications in Computer and Information Science*, pp. 247–258.

Lan, C. and Ramesh, B. (2007), “Agile software development: Ad hoc practices or sound principles?”, *IT Professional*, available at:https://doi.org/10.1109/MITP.2007.27.

Larman, C. and Vodde, B. (2009), *Scaling Lean & Agile Development Thinking and Organizational Tools for Large-Scale Scrum*, *Scaling Lean & Agile Development*.

Larman, C., Vodde, B. and Jensen, B. (2017), *Large-Scale Scrum*, Dpunkt.

Lee, G. and Xia, W. (2010), “Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility”, *MIS Quarterly: Management Information Systems*, available at:https://doi.org/10.2307/20721416.

Lefèvre, J., Charles, S., Bosch-Mauchand, M., Eynard, B. and Padiolleau, É. (2014), “Multidisciplinary modelling and simulation for mechatronic design”, *Journal of Design Research 9*, Inderscience Publishers Ltd, Vol. 12 No. 1–2, pp. 127–144.

Leffingwell, D. and Kruchten, P. (2007), *Scaling Software Agility: Best Practices for Large Enterprises*.

Lévárdy, V. and Browning, T.R. (2009), “An Adaptive Process Model to Support Product Development Project Management”, *IEEE Transactions on Engineering Management*, Vol. 56 No. 4, pp. 600–620.

Levina and Vaast. (2005), “The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems”, *MIS Quarterly*, Vol. 29 No. 2, p. 335.

Li, Y. and Maedche, A. (2012), “Formulating effective coordination strategies in agile global software development teams”, *International Conference on Information Systems, ICIS 2012*.

“Liberating Structures”. (2022), , available at: https://liberatingstructures.de/.

Lindemann, U. (2009), *Methodische Entwicklung Technischer Produkte : Methoden Flexibel Und Situationsgerecht Anwenden*, *Methodische Entwicklung Technischer Produkte*.

Lindkvist, L., Bengtsson, M., Svensson, D.-M. and Wahlstedt, L. (2016), “Replacing old routines: how Ericsson software developers and managers learned to become Agile”, *Industrial and Corporate Change*, p. dtw038.

Lindlöf, L. and Furuhjelm, J. (2018), “Agile beyond software - A study of a large scale agile initiative”, *Proceedings of International Design Conference, DESIGN*, available at:https://doi.org/10.21278/idc.2018.0411.

Loasby, B.J., Nelson, R.R. and Winter, S.G. (1983), “An Evolutionary Theory of Economic Change.”, *The Economic Journal*, Vol. 93 No. 371, p. 652.

Luckel, J., Koch, T. and Schmitz, J. (2000), “Mechatronik als integrative Basis für innovative Produkte”, *Mechatronik - Mechanisch/Elektrische Antriebstechnik: Tagung Wiesloch, 29. Und 30. März 2000*.

Lückel, J., Koch, T., Schmitz, J., Gausemeier, J., Czubayko, R., Lemke, J., Anderl, R., *et al.* (2000), “Computer-Aided Design of Mechatronic Systems, Exemplified by the Integrated Wheel Suspension of an Innovative Service Vehicle”, *IFAC Proceedings Volumes*, available at:https://doi.org/10.1016/s1474-6670(17)39240-6.

Lyytinen, K. and Rose, G.M. (2006), “Information system development agility as organizational learning”, *European Journal of Information Systems*, Vol. 15 No. 2, pp. 183–199.

MacCormack, A., Baldwin, C. and Rusnak, J. (2012), “Exploring the duality between product and organizational architectures: A test of the ‘mirroring’ hypothesis”, *Research Policy*, Vol. 41 No. 8, pp. 1309–1324.

Mackenzie, A. and Monk, S. (2004), “From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice”, *Computer Supported Cooperative Work: CSCW: An International Journal*, available at:https://doi.org/10.1023/B:COSU.0000014873.27735.10.

Malone, T.W. and Crowston, K. (1990), “What is coordination theory and how can it help design cooperative work systems?”, *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, CSCW 1990*, No. October, pp. 357–370.

Maples, C. (2009), “Enterprise Agile Transformation: The Two-Year Wall”, available at:https://doi.org/10.1109/agile.2009.62.

March, J.G. and Simon, H.A. (1958), “Organizations.”, *NY: Wiley, New York*.

Marks, M.A., DeChurch, L.A., Mathieu, J.E., Panzer, F.J. and Alonso, A. (2005), “Teamwork in Multiteam Systems.”, *Journal of Applied Psychology*, Vol. 90 No. 5, pp. 964–971.

Marks, M.A., Mathieu, J.E. and Zaccaro, S.J. (2001), “A temporally based framework and taxonomy of team processes”, *Academy of Management Review*, available at:https://doi.org/10.5465/AMR.2001.4845785.

Martensson, T., Hammarstrom, P. and Bosch, J. (2017), “Continuous Integration is Not About Build Systems”, *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, pp. 1–9.

Mathiassen, L. (2017), “Designing Engaged Scholarship: From Real-World Problems to Research Publications”, *Engaged Management ReView*, Vol. 1 No. 1, pp. 17–28.

Mathieu, J.E., Marks, M.A. and Zaccaro, S.J. (2001), “Multiteam systems.”, in Anderson, N., Ones, D., Sinangil, H. and Viswesvaran, C. (Eds.), *Handbook of Industrial, Work and Organizational Psychology*, London: Sage, pp. 289-313.

McEvily, B., Perrone, V. and Zaheer, A. (2003), “Trust as an organizing principle”, *Organization Science*, available at:https://doi.org/10.1287/orsc.14.1.91.12814.

McGrath, J.E., Arrow, H. and Berdahl, J.L. (1999), “Cooperation and conflict as manifestations of coordination in small groups”, *Polish Psychological Bulletin*.

McIntyre, R.M. and Salas, E. (1995), “Measuring and managing for team performance: Emerging principles from complex environments.”, *Team Effectiveness and Decision Making in Organizations*.

Meißner, M. and Blessing, L. (2006), “Defining an adaptive product development methodology”, *9th International Design Conference, DESIGN 2006*.

Middleton, P., Taylor, P.S., Flaxel, A. and Cookson, A. (2007), “Lean principles and techniques for improving the quality and productivity of software development projects: a case study”, *International Journal of Productivity and Quality Management*, Vol. 2 No. 4, p. 387.

Mintzberg, H. (1989), *Mintzberg on Management: Inside Our Strange World of Organizations*, Simon and Schuster.

Moe, N.B., Olsson, H.H. and Dingsøyr, T. (2016), “Trends in large-scale agile development: A summary of the 4th workshop at XP2016”, *ACM International Conference Proceeding Series*, available at:https://doi.org/10.1145/2962695.2962696.

Moe, N.B., Šmite, D., Paasivaara, M. and Lassenius, C. (2021), “Finding the sweet spot for organizational control and team autonomy in large-scale agile software development”, *Empirical Software Engineering*, Vol. 26 No. 5, p. 101.

Moen, R. and Norman, C. (2009), “Evolution of the PDCA Cycle”, *Society*, pp. 1–11.

Nerur, S. and Balijepally, V. (2007), “Theoretical reflections on agile development methodologies.”, *Communications of the ACM 50.3*.

Nyrud, H. and Stray, V. (2017), “Inter-team coordination mechanisms in large-scale agile”, *Proceedings of the XP2017 Scientific Workshops on - XP ’17*, ACM Press, New York, New York, USA, pp. 1–6.

Oestereich, B. and Weiss, C. (2008), *Agiles Projekt Management*, dpunkt.verlag.

Okhuysen, G.A. (2001), “Structuring change: Familiarity and formal interventions in problem-solving groups”, *Academy of Management Journal*, available at:https://doi.org/10.2307/3069416.

Okhuysen, G.A. and Bechky, B.A. (2009), “10 Coordination in Organizations: An Integrative Perspective”, *The Academy of Management Annals*, Vol. 3 No. 1, pp. 463–502.

Okoli, C. and Schabram, K. (2010), “A Guide to Conducting a Systematic Literature Review of Information Systems Research”, *SSRN Electronic Journal*, Vol. 10 No. 2010, available at:https://doi.org/10.2139/ssrn.1954824.

Ottosson, S., Björk, E., Holmdahl, L. and Vajna, S. (2006), “Research approaches on product development processes”, *9th International Design Conference, DESIGN 2006*, pp. 91–102.

Ovesen, N. (2012), “The Challenges of Becoming Agile: Implementing and Conducting SCRUM in Integrated Product Development”, p. 200.

Paasivaara, M. and Lassenius, C. (2014), “Communities of practice in a large distributed agile software development organization - Case Ericsson”, *Information and Software Technology*, available at:https://doi.org/10.1016/j.infsof.2014.06.008.

Paasivaara, M., Lassenius, C. and Heikkilä, V.T. (2012), “Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?”, *International Symposium on Empirical Software Engineering and Measurement*, pp. 235–238.

Paetzold, K., Biffl, S., Lüder, A. and Gerhar, D. (2017), “Product and Systems Engineering/CA* Tool Chains”, *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, Springer, p. 36.

Pahl, G. and Beitz, W. (2013), *Pahl/Beitz Konstruktionslehre*, edited by Feldhusen, J. and Grote, K.-H.*Pahl/Beitz Konstruktionslehre*, 8., Springer Berlin Heidelberg, Berlin, Heidelberg, available at:https://doi.org/10.1007/978-3-642-29569-0.

Pahl, G. and Beitz, W. (2021), *Pahl/Beitz Konstruktionslehre*, edited by Bender, B. and Gericke, K.*Pahl/Beitz Konstruktionslehre*, Springer Berlin Heidelberg, Berlin, Heidelberg, available at:https://doi.org/10.1007/978-3-662-57303-7.

Pennings, J.M. (1975), “Interdependence and Complementarity-The Case of a Brokerage Office”, *Human Relations*, available at:https://doi.org/10.1177/001872677502800904.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. and Still, J. (2008), “The impact of agile practices on communication in software development”, *Empirical Software Engineering*, Vol. 13 No. 3, pp. 303–337.

Pimmler, T.U. and Eppinger, S.D. (1994), “Integration Analysis of Product Decompositions”, *ASME Design Theory and Methodology Conference*, Sloan School of Management Massachusetts Institute of Technology 38 Memorial Drive, E56-390 Cambridge, MA 02139.

Pinto, M.B., Pinto, J.K. and Prescott, J.E. (1993), “Antecedents and Consequences of Project Team Cross-Functional Cooperation”, *Management Science*, available at:https://doi.org/10.1287/mnsc.39.10.1281.

PMI. (2008), *A Guide to the Project Managemet Body of Knowledge (PMBoK Guide).*, 4th ed., Newtown Square: Project Management Institute.

Poth, A. and Wolf, F. (2017), “Agile Procedures of an Automotive OEM – Views from Different Business Areas”, *Uluslararasi Iliskiler*, Vol. 5, pp. 513–522.

Pries-Heje, L. and Pries-Heje, J. (2011), “Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India”, *2011 AGILE Conference*, IEEE, Salt Lake City, Utah, United States, pp. 20–28.

Prosci. (2021), “Stop Confusing agile with Agile”, available at: https://www.prosci.com/resources/articles/stop-confusing-agile-with-Agile.

Ramasesh, R. V. and Browning, T.R. (2014), “A conceptual framework for tackling knowable unknown unknowns in project management”, *Journal of Operations Management*, Elsevier B.V., Vol. 32 No. 4, pp. 190–204.

Ramesh, B., Pries-Heje, J. and Baskerville, R. (2002), “Internet software engineering: A different class of processes”, *Annals of Software Engineering*, available at:https://doi.org/10.1023/A:1020557725165.

Randy Evans, W. and Carson, C.M. (2005), “A social capital explanation of the relationship between functional diversity and group performance”, *Team Performance Management: An International Journal*, Vol. 11 No. 7/8, pp. 302–315.

Rathor, S., Xia, W., Batra, D. and Zhang, M. (2016), “What constitutes software development agility?”, *AMCIS 2016: Surfing the IT Innovation Wave - 22nd Americas Conference on Information Systems*.

Reagans, R., Argote, L. and Brooks, D. (2005), “Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together”, *Management Science*, available at:https://doi.org/10.1287/mnsc.1050.0366.

Rico, R., Sánchez-Manzanares, M., Gil, F. and Gibson, C. (2008), “Team Implicit Coordination Processes: A Team Knowledge–Based Approach”, *Academy of Management Review*, Vol. 33 No. 1, pp. 163–184.

Rigby, D.K., Sutherland, J., Noble, A. and Sutherland, J. (2018), “Agile at Scale- How to get from a few teams to hundreds”, *Harvard Business Review*, No. June.

Robson, C. (2011), *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*, *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*.

Rolland, K., Dingsoyr, T., Fitzgerald, B. and Stol, K.-J. (2016), “Problematizing agile in the large: alternative assumptions for large-scale agile development”, *37th International Conference on Information Systems (ICIS 2016)*, Dublin.

Ronkainen, J. and Abrahamsson, P. (2003), “Software development under stringent hardware constraints: do agile methods have a chance?”, *… and Agile Processes in Software …*, pp. 1012–1012.

Rook, P. (1986), “Controlling software projects”, *Software Engineering Journal*, Vol. 1 No. 1, p. 7.

Rosemann, M. and Vessey, I. (2008), “Toward improving the relevance of information systems research to practice: The role of applicability checks”, *MIS Quarterly: Management Information Systems*, Vol. 32 No. 1, pp. 7–22.

Royce, D.W.W. (1970), “Managing the Development of large Software Systems”, *Ieee Wescon*, No. August, pp. 1–9.

SAFe. (2021), “Scaled Agile Framework SAFe”, available at: https://www.scaledagileframework.com/.

Salas, E., Sims, D.E. and Burke, C.S. (2005), “Is there a ‘Big Five’ in Teamwork?”, *Small Group Research*, Vol. 36 No. 5, pp. 555–599.

Scheerer, A., Hildenbrand, T. and Kude, T. (2014), “Coordination in large-scale agile software development: A multiteam systems perspective”, *Proceedings of the Annual Hawaii International Conference on System Sciences*, IEEE, pp. 4780–4788.

Schmidt, T.S. (2019), *Towards a Method for Agile Development in Mechatronics*, Universität der Bundeswehr München.

Schmidt, T.S., Atzberger, A., Schrof, J.I., Gerling, C., Weiss, S. and Paetzold, K. (2019), *Agile Development Of Physical Products - An Empirical Study about Potentials Transition and Applicability*, available at: www.unibw.de/produktentwicklung-en.

Schömann, S.O. (2011), *Produktentwicklung in Der Automobilindustrie: Managementkonzepte Vor Dem Hintergrund Gewandelter Herausforderungen*, Springer-Verlag.

Schrof, J. and Paetzold, K. (2020), “Agile Produktentwicklung in einer zunehmend dynamischen Automobilwirtschaft: Potentiale und Grenzen”, *Wiener Motoren Symposium*, pp. 1–15.

Schrof, J., Schmidt, T.S. and Paetzold, K. (2018), “Eignungsanalyse agiler Prinzipien für die Entwicklung physischer Produkte”, *Design for X (DfX) Symposium*, Munich, p. 12.

Schrof, J.I., Atzberger, A., Paetzold, K. and Efthymios, P. (2019), “Potential of Technological Enablement for Agile Automotive Product Development”, *ICE 2019: 24th International Conference on Engineering, Technology and Innovation*, Nice, pp. 1–8.

Schrof, J.I. and Paetzold, K. (2019), “Product modularization requirements in agile automotive product development”, *DFX 2019: Proceedings of the 30th Symposium Design for X, 18-19 September 2019, Jesteburg, Germany*, The Design Society, Hamburg, p. 12.

Schwaber, K. (2013), “UnSAFe at any speed”, available at: https://kenschwaber.wordpress.com/2013/08/06/unsafe-at-any-speed/.

Schwaber, K. and Sutherland, J. (2020), *Scrum Guide*, available at: https://scrumguides.org/scrum-guide.html.

Scott, W.R. and Davis, G.F. (2015), *Organizations and Organizing: Rational, Natural and Open Systems Perspectives*, *Organizations and Organizing: Rational, Natural and Open Systems Perspectives*, available at:https://doi.org/10.4324/9781315663371.

Sein, Henfridsson, Purao, Rossi and Lindgren. (2011), “Action Design Research”, *MIS Quarterly*, Vol. 35 No. 1, p. 37.

Sekitoleko, N., Evbota, F., Knauss, E., Sandberg, A., Chaudron, M. and Olsson, H.H. (2014), “Technical Dependency in Large-Scale Agile Software Development”, *International Conference on Agile Software Development*, pp. 46–61.

Sharp, H., Robinson, H. and Petre, M. (2009), “The role of physical artefacts in agile software development: Two complementary perspectives”, *Interacting with Computers*, Vol. 21 No. 1–2, pp. 108–116.

Simon, H.A. (1996), *The Sciences of the Artificial (Vol. 136)*, MIT press.

Simsarian Webber, S. (2002), “Leadership and trust facilitating cross-functional team success”, *Journal of Management Development*, available at:https://doi.org/10.1108/02621710210420273.

Skelton, M. and Pais, M. (2019), *Team Topologies: Organizing Business and Technology Teams for Fast Flow*, IT

Revolution, available at: https://teamtopologies.com/.

Snape, D. and Spencer, L. (2003), *Qualitative Research Practice: A Guide for Social Science Students and Researchers*, *Edited by Jane Ritchie & Jane Lewis*.

Socha, D., Folsom, T.C. and Justice, J. (2013), “Applying agile software principles and practices for fast automotive development”, *Lecture Notes in Electrical Engineering*, available at:https://doi.org/10.1007/978-3-642-33738-3_8.

Socha, D. and Walter, S. (2006), “Is designing software different from designing other things?”, *International Journal of Engineering Education*, Vol. 22, pp. 540–550.

Stacey, R.D. (2007), *Strategic Management and Organisational Dynamics: The Challenge of Complexity to Ways of Thinking About Organisations*, *The Challenge of Complexity to Ways of Thinking About Organisations*.

Star, S.L. and Greisemer, J.R. (1989), “Institutional Ecology, ‘Translations’ and Boundary Objects: Amateurs and Professionals in Berekeley’s Museum of Verebrate Zoology”, *Social Studies of Science*, Vol. 19 No. 3, pp. 387–420.

Star, S.L. and Griesemer, J.R. (1989), “Institutional Ecology, ‘Translations’ and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907–39”, *Social Studies of Science*, available at:https://doi.org/10.1177/030631289019003001.

Stelzmann, E. (2012), “Contextualizing Agile Systems Engineering”, No. May, pp. 17–22.

Steward, D. V. (1981), “Design Structure System: a Method for Managing the Design of Complex Systems.”, *IEEE Transactions on Engineering Management*, Vol. EM-28 No. 3, pp. 71–74.

Stojanov, I., Turetken, O. and Trienekens, J.J.M. (2015), “A Maturity Model for Scaling Agile Development”, *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, IEEE, pp. 446–453.

Stray, V., Moe, N.B. and Hoda, R. (2018), “Autonomous agile teams: Challenges and directions for future research”, *Proceedings of the 19th International Conference on Agile Software Development: Companion*, ACM, New York, NY, USA, pp. 1–5.

Strode, D. (2014), “Measuring Coordination in Agile Software Development”, *Management*, p. 2.

Strode, D.E. (2005), *The Agile Methods: An Analytical Comparison of Five Agile Methods and an Investigation of Their Target Environment*, Massey University, Palmerston North, New Zealand.

Strode, D.E. (2016), “A dependency taxonomy for agile software development projects”, *Information Systems Frontiers*, Vol. 18 No. 1, pp. 23–46.

Strode, D.E., Hope, B., Huff, S.L. and Link, S. (2011), “Coordination effectiveness in an agile software development context”, *PACIS 2011 - 15th Pacific Asia Conference on Information Systems: Quality Research in Pacific*.

Strode, D.E., Huff, S.L., Hope, B. and Link, S. (2012), “Coordination in co-located agile software development projects”, *Journal of Systems and Software*, Vol. 85 No. 6, pp. 1222–1238.

Susman, G.I. (1983), “Action research: a sociotechnical systems perspective”, *Beyond Method: Strategies for Social Research*, Sage Beverly Hills, CA, Vol. 95, p. 113.

Sutherland, J. and Frohman, R. (2011), “Hitting the Wall: What to Do When High Performing Scrum Teams Overwhelm Operations and Infrastructure”, *2011 44th Hawaii International Conference on System Sciences*, IEEE, pp. 1–6.

Takeuchi, H. and Nonaka, I. (1986), “The New New Product Development Game”, *Harvard Business Review*, Vol. 64 No. 1, pp. 137–146.

Takeuchi, H., Rigby, D. and Sutherland, J. (2016), “Embracing Agile”, *Harvard Business Review*.

Taylor, F.W. (1916), “The principles of scientific management”, *Bulletin of the Taylor Society. Reprinted in Shafritz, J.M., & Ott, J.S. (Eds), Classic Organization Theory.*, Wadsworth Publishing Company, Belmont, CA, pp. 66–79.

Thomke, S. and Fujimoto, T. (2000), “The Effect of ‘Front-Loading’ Problem-Solving on Product Development Performance”, *Journal of Product Innovation Management*, available at:https://doi.org/10.1111/1540-5885.1720128.

Thomke, S. and Reinertsen, D. (1998), “Agile Product Development: Managing Development Flexibility in Uncertain Environments”, *California Review Management*, Vol. 41.

Thompson, J.D. (1967), *Organizations in Action: Social Science Bases of Administrative Theory.*, McGraw-Hill.

Thompson, J.D., Zald, M.N. and Scott, W.R. (2017), *Organizations in Action*, *Organizations in Action: Social Science Bases of Administrative Theory*, Routledge, available at:https://doi.org/10.4324/9781315125930.

Tripathi, N., Rodríguez, P., Ahmad, M.O. and Oivo, M. (2015), “Scaling Kanban for Software Development in a Multisite Organization: Challenges and Potential Solutions”, *Agile Processes in Software Engineering and Extreme Programming*, pp. 178–190.

Tseng, Y.H. and Lin, C.T. (2011), “Enhancing enterprise agility by deploying agile drivers, capabilities and providers”, *Information Sciences*, available at:https://doi.org/10.1016/j.ins.2011.04.034.

Tyflopoulos, E., Flem, D.T., Steinert, M. and Olsen, A. (2018), “State of the art of generative design and topology optimization and potential research needs”, pp. 1–15.

Ueding, B. (2014), *Automobilindustrie: Flop-Risiken Anhand von Praxisbeispielen*, Diplomica Verlag.

Ulrich, K.T. and Eppinger, S.D. (2015), *Product Design and Development*, 6th ed., New York.

Uludag, O., Kleehaus, M., Caprano, C. and Matthes, F. (2018), “Identifying and structuring challenges in large-scale agile development based on a structured literature review”, *Proceedings - 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference, EDOC 2018*, pp. 191–197.

VDI 2206. (2004), “VDI 2206”, VDI-Gesellschaft Produkt- und Prozessgestaltung.

Ven, A.H. Van De, Delbecq, A.L. and Koenig, R. (1976), “Determinants of Coordination Modes within Organizations”, *American Sociological Review*, Vol. 41 No. 2, p. 322.

Venkatesh, V., Thong, J.Y.L., Chan, F.K.Y., Hoehle, H. and Spohrer, K. (2020), “How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills”, *Information Systems Journal*, Vol. 30 No. 4, pp. 733–761.

VersionOne. (2020), *14th Annual STATE OF AGILE REPORT*, *Annual Report for the STATE OF AGILE*, available at: https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report%0Ahttps://stateofagile.com/#.

Vidgen, R. and Wang, X. (2009), “Coevolving Systems and the Organization of Agile Software Development”, *Information Systems Research*, Vol. 20 No. 3, pp. 355–376.

van Vliet, H. (2008), *Software Engineering: Principles and Practices*, 13th ed., John Wiley & Sons, Hoboken, NJ.

Walls, J.G., Widmeyer, G.R. and El Sawy, O.A. (1992), “Building an information system design theory for vigilant EIS”, *Information Systems Research*, INFORMS, Vol. 3 No. 1, pp. 36–59.

Wang, X., Conboy, K. and Pikkarainen, M. (2012), “Assimilation of agile practices in use”, *Information Systems Journal*, available at:https://doi.org/10.1111/j.1365-2575.2011.00393.x.

Weber, S. (2015), *Agile in Automotive - State of Practice 2015*.

Wedeniwski, S. (2015), *Mobilitätsrevolution in Der Automobilindustrie*, Springer.

Wegner, D.M. (1995), “A Computer Network Model of Human Transactive Memory”, *Social Cognition*, available at:https://doi.org/10.1521/soco.1995.13.3.319.

Wells, D. (2021), “When should Extreme Programming be Used?”, available at: http://www.extremeprogramming.org/when.html.

Williams, L. and Cockburn, A. (2003), “Agile software development: it’s about feedback and change”, *IEEE Computer*, Vol. 36 No. 6, pp. 39–43.

Wittenbaum, G.M. and Stasser, G. (1996), “Management of information in small groups.”, Sage Publications, Inc.

Wood, S., Michaelides, G. and Thomson, C. (2013), “Successful extreme programming: Fidelity to the methodology or good teamworking?”, *Information and Software Technology*, Vol. 55 No. 4, pp. 660–672.

Xu, P. (2009), “Coordination In Large Agile Projects”, *Review of Business Information Systems (RBIS)*, Vol. 13 No. 4, pp. 29–44.