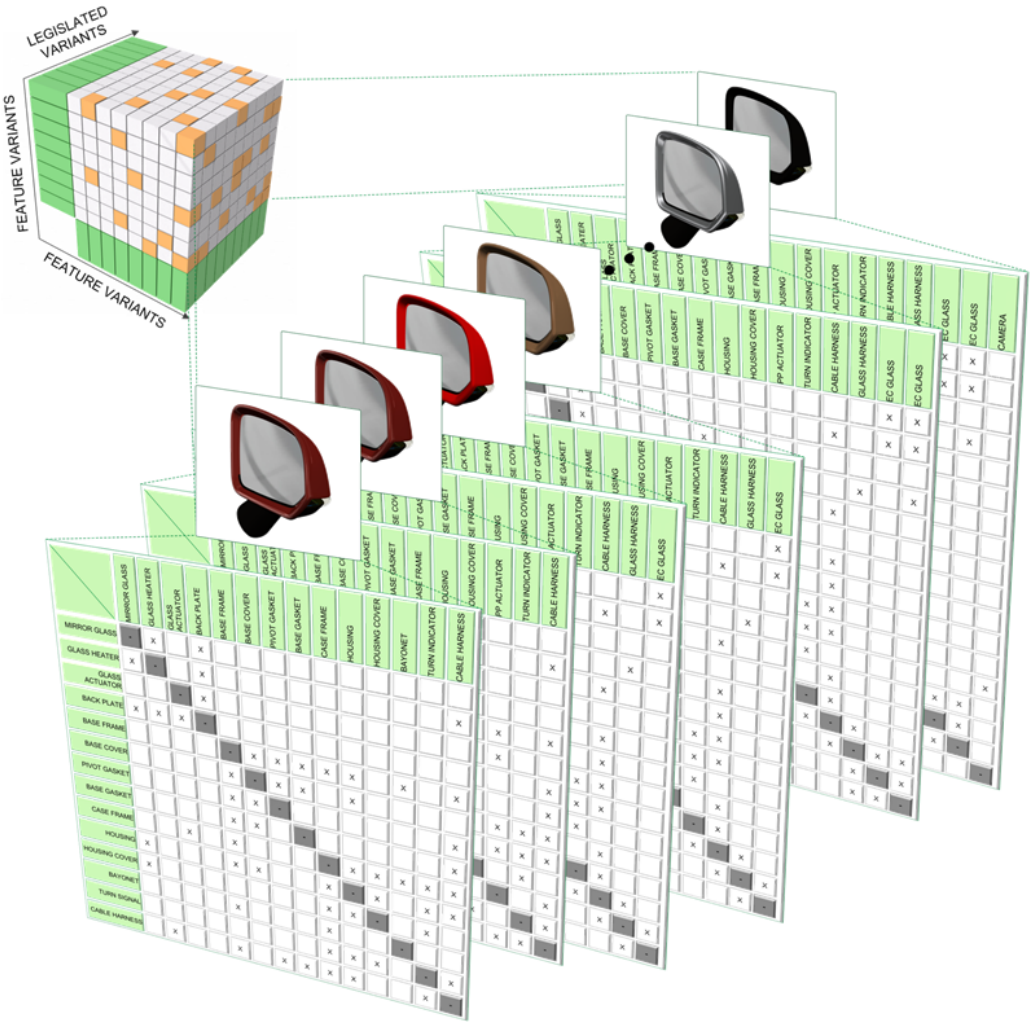
CHALMERS
UNIVERSITY OF TECHNOLOGY



Unveiling Variety-induced Complexity in Vehicle Development

Towards Complexity Management Using Change Prediction

Master of Science Thesis in Product Development

ALBERT CAHLIN

Department of Industrial and Materials Science
Division of Product Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:NN

Unveiling Variety-induced Complexity in Vehicle Development

Towards Complexity Management Using Change Prediction

ALBERT CAHLIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Product Development
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Unveiling Variety-induced Complexity in Vehicle Development
Towards Complexity Management Using Change Prediction
ALBERT CAHLIN

Supervisor: Ola Isaksson, Department of Industrial and Materials Science
Supervisor: Mikael Gustavsson, Volvo Cars
Examiner: Ola Isaksson, Department of Industrial and Materials Science

Master's Thesis 2020:NN
Department of Industrial and Materials Science, Chalmers
Division of Product Development
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31 772 1000

Cover: Illustrative figure of the evolution from feature variants to system variants, each with its own architecture, for exterior vehicle rear-view mirrors.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Preface

During the spring of 2016 I got in contact with the Product Data Management (PDM) group at Volvo Cars. Volvo Cars had a program that offered to combine an internship with a master's thesis. The fact that the group worked in the Product Lifecycle Management (PLM) field was essentially a fluke. The match was done by an external company with little knowledge — if any at all — of the area.

Pontus, the group manager, proposed two different topics for this thesis. The first was how to efficiently integrate virtual design with the physical bill-of-material. The second was about the area of product configuration. I choose the latter. There and then it was my first contact with the topic, but I have been hooked ever since.

Now some four and a half year later I am still working with the same challenges, in the same setting. Although the study presented in this thesis was conducted during the second half of 2016 and the beginning of 2017, it is as relevant today as it was then. During the spring of 2017 I was offered a full-time position at the PDM group. The unfinished thesis was placed in a drawer, awaiting the finishing touches. Time went by as I was devoted to applying many of my learnings in practice. Now, at the end of 2020 I am finally ready to publish the work that introduced me to the fascinating world of variants, variants and more variants.

Abstract

The dimension of product variety can turn complicated product development into an extremely complex task. Meanwhile, studies have shown that the development effort grows super-linearly with increased complexity. Differentiated products and speed of change can however be vital to meet the customer on her terms. The term *variety-induced complexity* was invented to describe the costs brought on by differentiation and numerous publications have been dedicated to studying them. Perhaps even more studies have been dedicated to the strategies used to keep them to a minimum. One of the most prolific strategies is design *re-use*, deriving product variants from a common platform.

It is a strategy to allow product variety and *manage* the complexity. Variety is still omnipresent, a fact companies need to deal with. Complexity costs during product development are often obscure and relate to the time design engineers spend on non-value adding activities. Meanwhile, the development task undergoes a small revolution. A 'good' design in an individual product is one that satisfy requirements. In a platform, it is also one with a *positive* contribution to the platform's complexity costs over time. Complexity creates complexity, and if not managed properly it creates an inertia that threaten the long-term success of product platforms.

This thesis studies an established automotive company and identifies that decision are often made in the disfavour of the product platform. The organisation, culture and information structures are in many ways biased to the benefit of individual products. We propose a framework that places the configurable design in the centre of decision-making. The aim is to give design engineers means to model and optimise the design and configuration space of a product platform to minimise complexity costs while meeting the needs of a range of customers.

In early design phases, the *functional bandwidth* the system presents to the customer is defined. By modelling the functions and the different ways to solve them, the system's *design bandwidth* is defined. Adding knowledge about how different solutions interact allows us to evaluate how *coupled* the design is. That is, by simulating changes to the design, and analysing how they propagate, the complexity and robustness of individual system variants can be assessed. This framework was applied to external vehicle Rear-view Mirrors (RVMs). By modelling an existing design, we showed that knowledge can be generated to support a selection of the architectural options and configurations with the least negative effects on complexity costs and robustness. That is, some solutions tend to increase the risks in the system and should be harmonised against their value. The framework also appeared promising to efficiently model and compare many configurations during design space exploration and to identify variants likely to be affected by a re-design.

Keywords: *Product development; Product variety; Platform-based development; Product Configuration; Complexity; Variety-induced complexity; Configurable Components; Function-means modelling; Change propagation analysis; Product Structure*

Acknowledgements

The work presented in this thesis was inquired by Volvo Cars, by the department of PDM Development and was carried out at Volvo Cars in cooperation with the department of product and production development at Chalmers University of Technology. I greatly acknowledge the support of both parties in carrying out the work.

I would like to express my deep gratitude towards my supervisor at Chalmers, Professor Ola Isaksson, for the advice and feedback that has been guiding my work. I am very grateful for your support, your patience and your belief in me. That has been truthfully encouraging.
A special thanks also to PHD Jakob Müller at the division of Product Development. Thank you for introducing the Configurable Component Modeller (CCM), for your support and advice throughout all the hassles. I would also like to thank Magnus Andersson from PE Geometry for his help with the CCM tool. Thanks also to Timos Kipouros of the Engineering Design research group at the University of Cambridge for your support using the Cambridge Advanced Modeller (CAM).

I am very grateful by the reception and encouragement I have received from Volvo Cars. To the department of PDM Development, thank you for taking me in and giving me everything I needed during my work. To Pontus Albrektsson, thank you for opening the door to this subject, for your positivity and your support.
I would also express my gratitude towards my supervisor, Mikael Gustavsson for his relentless support. You always had an answer to every possible question.

This work would not have been possible without the candidness shown by all the people volunteering to participate in interviews and in discussions. I would like to thank all the people at Volvo Cars that in some way have answered my opinionated and intricate questions, may it have been in interviews or in another context. Some of you have really shown a great effort to be available with enthusiastic answers even during one of the most hectic times at Volvo Cars to date.

Albert Cahlin, Gothenburg, 2020

Table of Contents

List of Acronyms

BOM	Bill of Materials
C	Constraint
CAD	Computer Aided Design
CAM	Cambridge Advanced Modeller
CC	Configurable Component
CCM	Configurable Component Modeller
CE	Composition Element
CI	Control Interface
CPM	Change Propagation Methodology
DR	Design Rationale
DRM	Design Research Methodology
DS	Design Solution
DSM	Design Structure Matrix
EC	Engineering Change
ECU	Electrical Control Unit
EF-M	Enhanced Function-means
F-MT	Function-means Tree
FoV	Field-of-View
FR	Functional Requirement
HMI	Human-machine Interface
iib	is influenced by
iw	interacts with
MC	Mass Customisation
pd	product development
PDM	Product Data Management
PLM	Product Lifecycle Management
RVM	Rear-view Mirror
VP	Variant Parameter
VPV	Variant Parameter Value

List of Figures

List of Tables

1

Introduction

The ever-changing global markets of the twenty-first century have redefined the competitive arena for many companies; with fiercer competition, shorter product life-cycles and time-to-market as well as increasing product variety (Alford et al., 2000; Ben-Arieh et al., 2009; Qu et al., 2011; Sanderson & Uzumeri, 1997). In addition, there has been an explosion in interactions both within and between technological products, largely driven by the recent software explosion. New technology artefacts are steadily incorporated together with an increasing heterogeneity in functionality. Vehicles are prime examples of such highly complex products.

The postmodern customer is often associated with extreme individualism (Cova, 1996). Pine (1999) highlighted *product variety* more than three decades ago. He established that products must be differentiated and target each customer's needs in order to improve customer satisfaction on a saturated market. Product variety — the opposite to the standardisation efforts driven by traditional mass production — is a key root-cause to the complexity characterising modern pd. This is a widely reported strategy to maintain economy of scale advantages, while satisfying diverse customer needs through customisation (Huffman & Kahn, 1998; Jiao et al., 2007; Pine, 1999; Salvador et al., 2009). Salvador and Forza (2004, p. 275) highlight that the necessity in this paradigm is to *'understand and fulfil each individual customer's increasingly diverse wants and needs — while meeting the coequal imperative for achieving low cost'*.

The impact of Mass Customisation (MC) in the automotive industry are well documented (Alford et al., 2000). With hundreds of attributes and tens of thousands of components, car manufacturers typically face the possibility to theoretically offer over 10^{100} unique product configurations (Tidstam & Malmqvist, 2015)[1]. The challenge for any manufacturing firm is to supply an optimum balance of variety that customers want — *commercial variety* — and variety introduced into the industrial system — *internal variety* due to diminishing returns fuelled by complexity costs (Qu et al., 2011; Simpson, 2004; Wortmann et al., 1996).

Managing variety — supplying high product variety with low complexity in development and manufacturing — is thus an imperative capability for success (Blecker et al., 2004). Variety has largely overturned how products are developed today. Commonly used strategies for variety management include *modularization*, *platform-based development* and *product family design* (Jiao et al., 2007; Meyer & Lehnerd, 1997; Simpson, 2004; Simpson et al., 2001). Ultimately, the incentive for any manufacturing firm is to be able to accurately position their offered commercial variety supported by innovative product architectures, processes and methodologies to capitalise on, rather than to fail with, complexity. This is however an incentive poorly supported by many ingrained information flows, structures and methodologies in mature pd organisations.

[1] Mercedes-Benz reported more than 10^{103} possible configurations for one vehicle family (Kübler et al., 2010).

1.1 Background

Mass Customisation (MC) is challenging in practice and companies accustomed to mass production may have difficulties adapting to it (Blecker & Abdelkafi, 2006). Ferguson et al. (2014) conclude that there is a lack of innovations that enable customisation at sufficient economic efficiency. The full potential of MC is thus yet to be realised. The authors emphasise the need for new tools that support design engineers. Jiao et al. (2007) request effective tools to analyse and manage both the profit from variety and the costs of complexity in product portfolio decision-making. While most of the MC literature focusses on the impact of product variety in the manufacturing system, few studies explore the in-depth implications in pd.

1.1.1 Variety and Complexity in Product Development

Sinha and de Weck (2013) ascribe many of the challenges with today's pd to the increasing complexity of large-scale engineered systems. Complexity correlates to development efforts (Bashir & Thomson, 2001). For example Sinha and de Weck (2016) shows that it grow super linearly with increasing *structural complexity*. This refers to complexity that arises from the number of system elements and their dependency structure. Internal variety adds another dimension — with effects on operations, structures and information flows (Saeed & Young, 1998) — labelled *variety-induced complexity* (Blecker & Abdelkafi, 2006).

Complex, large-scale systems have historically been designed pre-eminently based on trial-and-error and empiricism, without objective means to understand and assess complexity (Suh, 2005). Sinha and de Weck (2013) ascribe large cost overruns in development to an inability to *'characterise, quantify and manage associated complexity'*. Narrowed down to variety, empiricism and heuristics are seen favoured over a systematic design space exploration supported by an assessment of the individual variant's contribution to the overall architecture's complexity.

1.1.2 The Role of Product Architectures in Mass Customisation

Several authors distinguish *commercial variety* (*'functional variety'* by Du et al. (2001) and *'external variety'* by Anderson (1997)) — related to customer satisfaction — from *internal variety* (Anderson, 1997) (*'technical variety'* by Du et al. (2001)) — related to the costs of complexity in the industrial system. Jiao et al. (2007) highlight several strategies under the umbrella of design for technical variety that aims at minimising the complexity costs. They mostly aim at optimising the *product architecture* — the *'scheme by which the function of a product is allocated to physical components'* (Ulrich, 1995) — with respect to e.g. re-use.

The architecture can to a great extent impact the scope and costs of product variety (Ulrich & Eppinger, 2012). Modular product architectures can minimise complexity as they aim at finding a maximum of distinct product variants, with minimal internal variation (Landahl, 2016). A modular architecture is composed by a collection of independent parts (treatable as logical units), that can be designed independently yet function together as a whole (Baldwin & Clark, 1997; Newcomb et al., 1996). Automotive architectures are however pre-dominantly integral, characterised by complex mappings between functional elements and physical structures (Simpson, 2004). In addition, they need to be *maintained* with new requirements and technologies to be competitive over time (Lewandowski et al., 2015). Architectural flexibility is required to leverage commonality in the long-term (Landahl, 2016; Muffatto & Roveda, 2000).

Product family design and *platform-based development* provide a strategic umbrella under which commonality is captured and utilised as specific parts are re-used across a range of products, minimising costs (Aigbedo, 2007; Jiao et al., 2007). Meyer and Lehnerd (1997) define a *product*

family as the set of similar products, yet possessing specific features or functionality meeting specific customer requirements, that are derived from a common platform. Each instance within a product family can be termed a *product model*, each possible to configure into a set of unique product variants. Within a family, some designs will be shared while others are specific to individual product variants (Tidstam & Malmqvist, 2015).

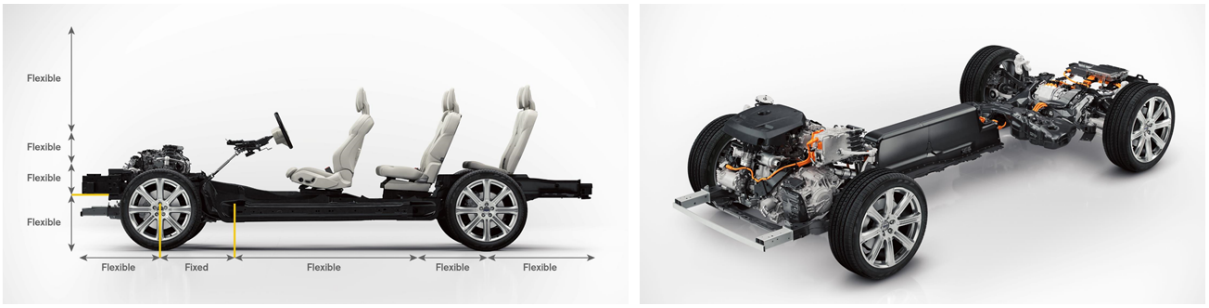


Figure 1.1: Illustration of the company's product platform. It is scalable to different lengths and heights, and modular to for instance carry different engine options.

While a product family targets a certain market segment, each product variant addresses a specific subset of customer needs in that segment (Jiao et al., 2007). According to Meyer and Lehnerd (1997), most companies have historically developed new products one at the time, focusing only on a small set of customers, resulting in a *'failure to embrace commonality, compatibility, standardisation or modularization among different products or product lines'*. Even with the aforementioned strategies, residuals from this tradition remain in many companies.

1.1.3 Company Description

The global automotive company represented in this study has nearly a hundred-year-old tradition of producing cars. Behind its recent success is a credible global brand and a major transformation with vast investments in R&D. Not more than five years ago, the company re-aligned its strategy and repositioned its brand towards the premium segment to increase profitability and to ensure long term growth. All new cars are based on two new *product platforms*, one engine family and one transmission family. Figure 1.1 illustrated one of the platform. This transformation is still in its early days, and tradition continue to influence of the vehicles are developed.

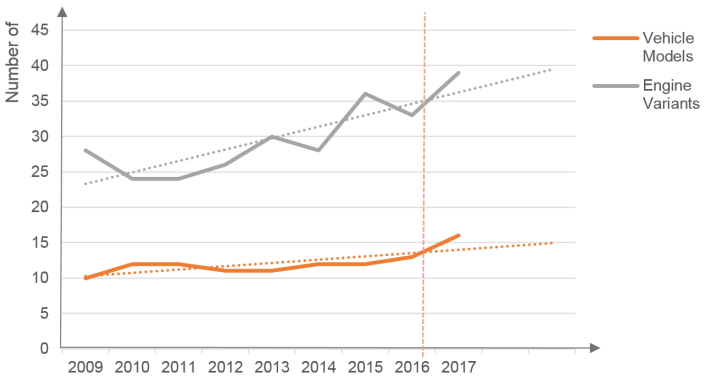


Figure 1.2: The number of car models and engine variants on the market as developed over the last seven years.

The platforms are the result of a strategy to target complexity costs. Investments in a broad and renewed product portfolio is key in reaching the company's ambitious sales targets. New vehicle models are, and will be, continuously introduced at a steady pace, along with a continuous stream of face lifts, Figure 1.2.

The automotive industry is anticipated to transform significantly the next decades. The company

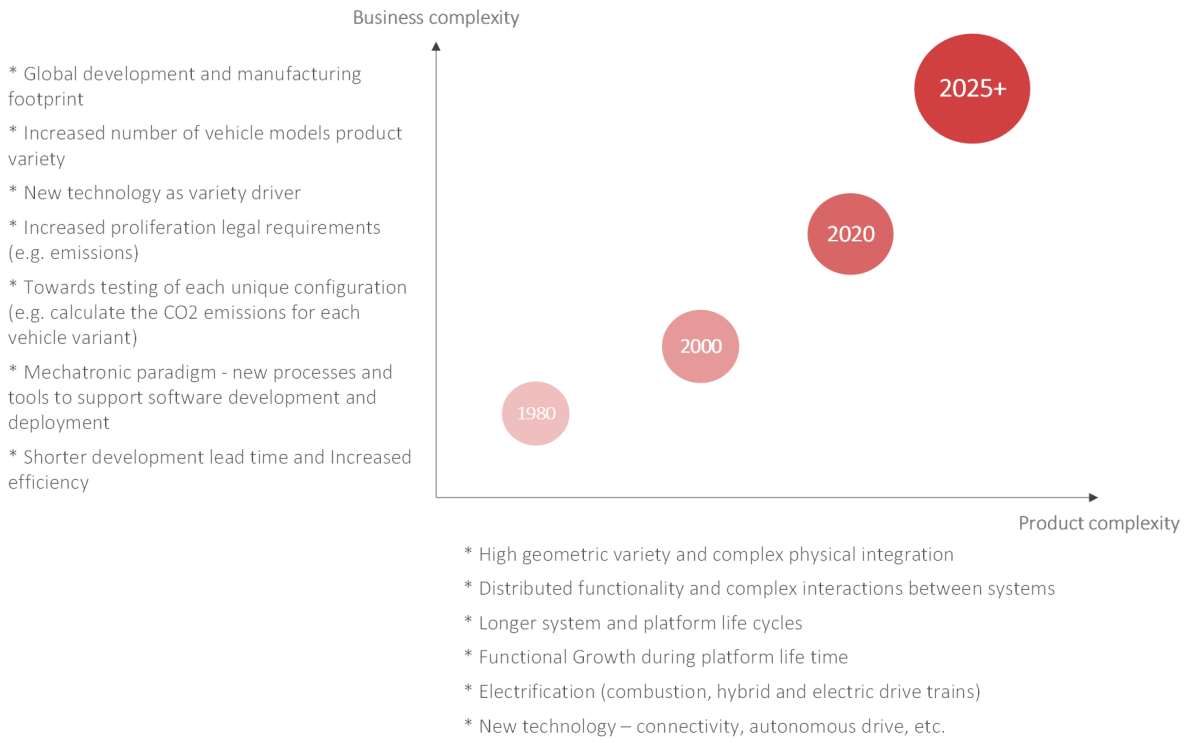


Figure 1.3: Trends, as foreseen by the company, which impose new challenges on pd.

is, like most automotive companies emphasising on electric and hybrid vehicles, autonomous drive and connectivity. These are technology trends that together new industry entrants and expectations on novel functionality faster and with increased quality change the competitive landscape. Many such trends drive business- and product-related complexity, Figure 1.3.

The company's product architecture is separated into A-, B- and C-systems in order to integrate all systems in the right order, Figure 1.4. Critical systems need to mature earlier as they define many critical design pre-conditions. Approximately 90 per cent of all A-systems belong to the *product platform*[2] which are mainly defined in the *architectural definition phase*. Systems not belonging to these groups are typically referred to as part of the *top hat*. Figure 1.5 conceptually describe the different development phases for a product family at the company[3].

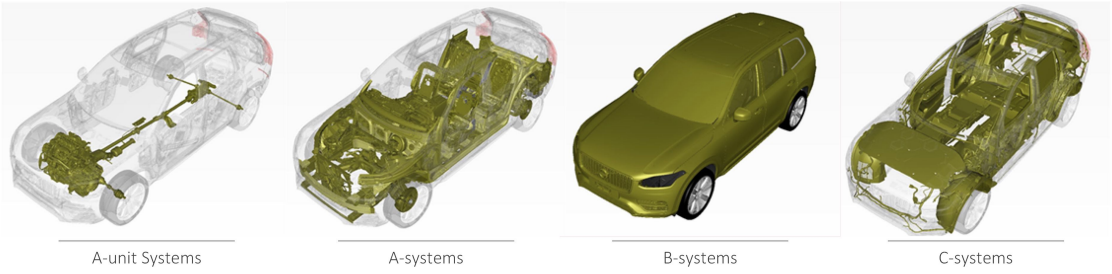


Figure 1.4: Different types of systems, categorised based on complexity, lead times and dependencies.

[2]A platform is internally defined as *a common basis for a number of cars with a high level of commonality.* However, it has no fixed proportion of common solutions. It usually consists of: *1* sub-frame; *2* part of chassis (wheel, wheel suspension, steering gear); and *3* power train (engine, gearbox and driving shaft).

[3]As the platform is not fine-tuned for the next vehicle family, an architectural phase precedes that development project as well where the already mature platform is modified.

As the design pre-conditions start to mature industrial design (henceforth referred to as *styling*) and the sales department gain influence over development. Late concept phases, engineering and industrialisation are characterised by continuous iterations between design engineers (including software developers), test engineers, vehicle and system integration and styling. A project takes approximately 32 months to finish. Adding the architecture phase it takes around 44 months. The time, uncertainty and number of people involved add to the many challenges of developing vehicles in the MC, many of which are new to the company.

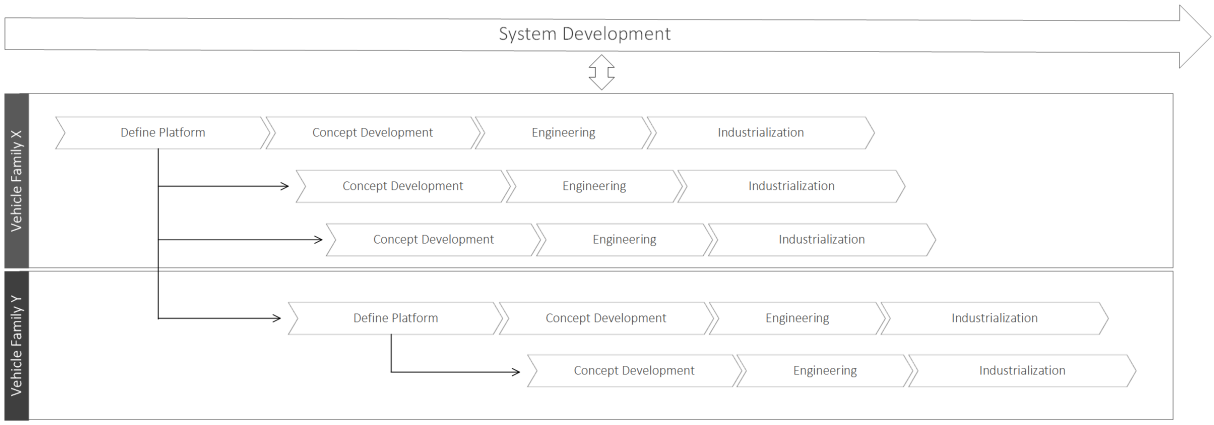


Figure 1.5: Conceptual schematic over the pd processes at the company.

The changes in the automotive industry challenge established practices for many firms. Expectations on the studied company — decreased lead times, increased profitability and customer satisfaction — combined with changes on micro level, require the company to become more agile and responsive. The changes include truly becoming a large, premium car manufacturer and embracing the new pd paradigm. Hence, there is a need for new tools and methodologies managing product variety and complexity, ultimately strengthening the competitiveness of the company's pd process.

1.2 Mission of the Thesis

The overall mission of this thesis is to increase the understanding of variety-induced complexity and its challenges in vehicle development and how novel tools and practices can be used to mitigate the costs of this complexity. This mission is further specified and developed below.

1.2.1 Problem Statement

The phenomenon studied in this thesis is that of *product variety and the complexity induced by internal variety*. The possible set of product variants is constrained to those obtainable by combining a set of pre-defined components into those technically feasible (Salvador & Forza, 2004). With the increasing number of components and functions in modern vehicles, this set becomes immensely too large to handle. With only 1.6 of all vehicles built at the company today being identical, the organisational efforts required to maintain control over each product configuration are substantial. Achieving an optimal level of manageable product variants is the key strategic challenge considered in this thesis. This is a process typically based on experience, repetition and heuristics. As is shown in Figure 1.6 this is a trade-off between the revenue provided by commercial variety and the complexity costs in the industrial system.

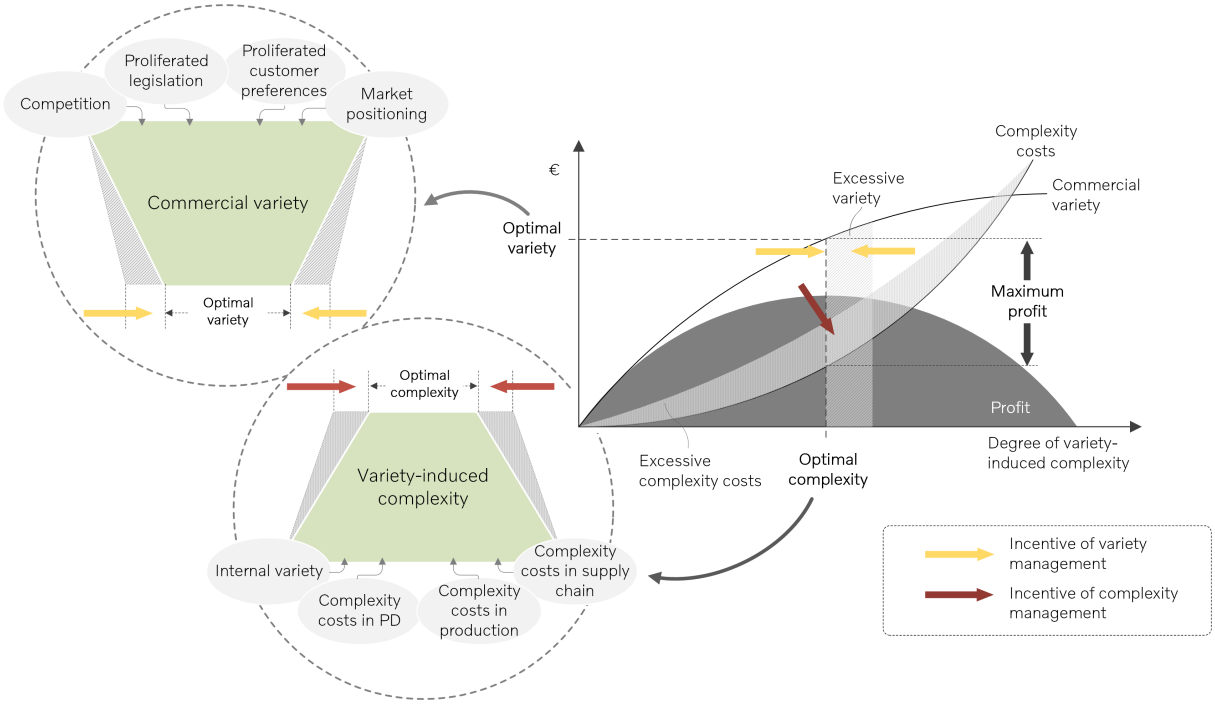


Figure 1.6: Illustration of the economic challenge with developing configurable products.

There are two themes in organisations and literature targeting this challenge; variety management and complexity management. The former with the incentive of maximising revenue through an optimal level of commercial variety. The latter by reducing complexity costs, typically including strategies such as modularization, platform-based development and product family design. The research gap of this thesis is shown in Figure 1.7.

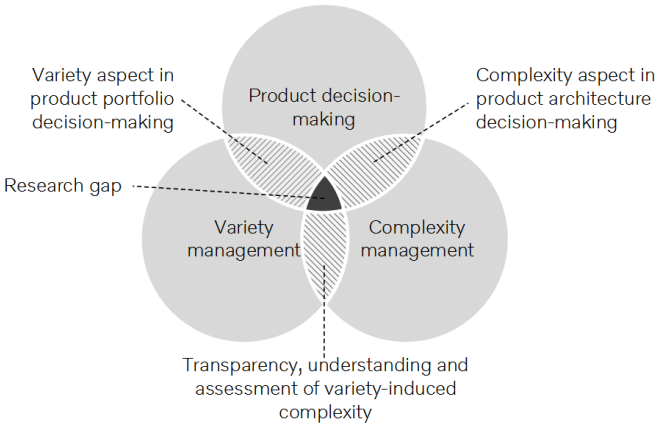


Figure 1.7: The research gap studied in this thesis is located in the intersection of variety management, complexity management and product decision making, adapted from Götzfried (2013).

The studied problem is the inability to effectively integrate variety-induced complexity in product portfolio and architecture decision-making. In particular, efficiently designing solutions minimising variety and its induced complexity in an overall complex product. That is, making the right decisions early, when mature information is scarce.

1.2.2 Aim and Goal

The overall aim of this thesis is two-fold:

(1) To investigate and describe challenges of automotive pd resulting from complexity induced by internal variety.

(2) To investigate and demonstrate support for integrating an understanding and assessment of such complexity in product decision-making.

1.2.3 Research Questions

Three research questions were formulated, based on the mission of the thesis. They are defined in order to frame and focus the research:

> ***RQ 1:*** *What are the main challenges that can be traced back to dealing with variety-induced complexity in pd?*

The aim of this question is to zoom out of the main phenomenon in a manufacturing firm. The answer to the question is mainly based on a descriptive study with internal stakeholders, supported by a literature review.

> ***RQ 2:*** *Given a subset of challenges (from RQ 1); what are the root-causes and results on product performance and/or organisational efficiency?*

The aim of this question is ti limit the work by zooming in on specifics of the answers to *RQ 2*. The answer to *RQ 2* is based on a more focused descriptive study.

> ***RQ 3:*** *How can design engineers and related functions be supported in better managing these root-causes?*

This third research question aims at identifying supports applicable to the challenges identified by *RQ 2*. The answer is mainly based on a comprehensive literature review.

1.2.4 Research Hypothesis

The way forward, based on the answer to RQ 2, was driven by a secondary phenomenon, namely *early availability of information describing variety, complexity and design pre-conditions.* In order to answer RQ 3, a hypothesis was stated and tested through a case study:

> *Analysing change propagation in the functional domain enables design engineers to assess variety-induced complexity and to design flexible and robust products at reduced complexity costs.*

1.3 Scope and Delimitations

The scope of this study is to provide a comprehensive description of challenges resulting from variety-induced complexity in vehicle development based on a qualitative study at a premium car manufacturer, along with a brief description of suitable support for better managing such complexity in pd. In particular, this thesis shall deliver:

1. An increased understanding about variety-induced complexity at an automotive firm, including identified root-causes and negative company effects, based on a comprehensive qualitative study.
2. New insights and state-of-the-art methodologies enabling organisations to be more resilient against the negative effects of variety-induced complexity, based on a theoretical framework applied to a simplified use case.

The subject of configuration management composes the thesis' theoretical framework. It considers the activity of establishing initial product specifications and maintaining consistency between the product, requirements and associated product configuration information (Stark, 2015; Usman & Kidd, 2015). As this scope is quite broad the study has been significantly delimited.

This study is delimited to those activities taking place in the conceptual, engineering and industrialisation phases of pd and their supporting functions (including product planning, manufacturing engineering and supply chain planning) at the studied company. Any benchmarks with other companies are not included in this thesis. Further delimitations include:

- The investigation of supporting methodologies is delimited to methods and processes in research literature targeting the creation and management of information describing variety, complexity and design pre-conditions.
- The work is delimited to the applicability of existing theories and methods, not to the development or improvement of such theories or their related software.
- The usefulness of the proposed support framework is delimited to the functional domain and its applicability in the physical domain is only discussed briefly.
- The use case is delimited to the design area of external RVMs at the studied company. Improvements to their design are not suggested, rather the case study is a way to exemplify the suggested framework.
- The scope does not contain evaluating economic benefits and drawbacks with the proposed framework.

1.4 Thesis Outline

The remainder of the thesis is organised as to provide the reader with a clear overview of the findings underpinning the answer to each of the three research questions as well as providing a concise overview of the study's context and of the course of action taken.

Chapter 2 provides the reader with the research framework together with a presentation and justification of the research methodology used to collect, analyse and validate data.

Chapter 3 provides a concise compilation of published theories relevant for the thesis. The content aims at providing a context for the studied phenomenon as well as complementing the empirical result and the proposed support.

Chapter 4 provides findings from the empirical study. The empirical data is structured as to describe the challenges and organisational symptoms caused by variety-induced complexity.

Chapter 5 discusses how change prediction, EF-M modelling and the CC framework can be used as a support for enhanced complexity management. The discussion evolves around published theories and is exemplified using a case study on RVMs.

Chapter 7 provides a summary of the answers to the research questions and a discussion about the validity and the generalisation of the empirical data.

Chapter 8 concludes the work and presents outlooks for future work.

2

Research Methodology

This chapter presents how the research that underpin this thesis was approached. The applied research framework and methodology is introduced to the reader. How data was collected, analysed and validated in order to answer the research questions stated is presented. Finally, the choice of methods is described and justified.

2.1 Research Environment

Blessing and Chakrabarti (2009) define design in relation to research as *those activities that actually generate and develop a product from a need, product idea or technology to the full documentation needed to realise the product and to fulfil the perceived needs of the user and other stakeholders*. Design is a complex process as it involves, among other, people, an often complicated-product, a process involving a variety of activities, procedures, tools and methods, and an organisation (Blessing & Chakrabarti, 2009). When conducting research on engineering design, the formal as well as the informal design processes undertaken by individuals can be seen as key sources of data. This study was mainly carried out on site at the company, with direct access to the design activities composing the research environment.

2.2 Research Strategies

Bryman and Bell (2011) makes a distinction between the two strategies: *Qualitative research* and *Quantitative research*. The former is an approach to understand the reality based on the participants' own interpretation. The latter is an approach that stresses data collection and testing theories. The research conducted in this thesis is mainly of an exploratory character, thus taking the path of qualitative research. This is mainly due to the difficulties quantifying the studied phenomenon. It is assumed that the phenomenon, embedded in a complex process, is best understood through the participants own experiences. The problem is thus approached holistically as the scope includes identifying real industrial needs not necessarily obvious to the non-initiated.

Qualitative research is characterised by an *inductive view* — theory is a result of generalised conclusions based on observations — on the relation between theory and practice (Bryman & Bell, 2011). Moreover, its exposition is less prescriptive than that of its quantitative counterpart. It is characterised by deduction — hypotheses are tested based on theoretical premises (Creswell, 2009). Based on the hypothesis that the findings represent an actual organisational need, quantitative methods have thus been included to validate the results.

2.3 Research Framework

Blessing and Chakrabarti (2009) introduced a Design Research Methodology (DRM) as a framework for design research. This is also the basis in this thesis. The DRM consists of four stages,

illustrated in Figure 2.1. Blessing and Chakrabarti present several types of research in their framework, applicable in different scenarios. This study has, to some extent, been through all stages. Most of the time and effort have however been spent on Research Clarification and the Descriptive Study I. This study uses a research type being *review-based* in the *Research Clarification (RC)* stage, meaning that an initial understanding about the existing situation is obtained through a literature review.

The *Descriptive Study I (DS-I)* stage employ a *comprehensive* study (partly implemented in parallel with the RC stage in order to gain a company specific understanding about the current situation and to clarify the research goal), meaning that empirical studies are the main sources of data. The *Prescriptive Study (PS)* stage allowed for a *review-based* study, in this case meaning that the findings from a literature review, showing promise to improve the as-is situation were outlined. The *Descriptive Study II (DS-II)* stage again employ a *comprehensive* study as the support identified in the PS stage were evaluated (if only at high level) through a case study.

Describing the existing and the desired situation has a central role in the framework. This is represented by a *Reference Model* representing the existing situation (the baseline against which intended improvements are benchmarked) and an *Impact Model* representing the intended situation (indicates the impact of the support to be developed). An initial reference and impact model was outlined during the RC stage and completed during the DS-I stage. The DS-I stage was in this study itself divided into two sub-phases: *Explore current situation*, And *define and detail problem*. The purpose of the former being to gain knowledge about the current situation and making adequate delimitations. Reviewing empirical data during the DS-I stage was intended to be sufficient to complete the reference model, to refine the impact model and to identify a *Key Factor* — addressed in the PS stage (Blessing & Chakrabarti, 2009).

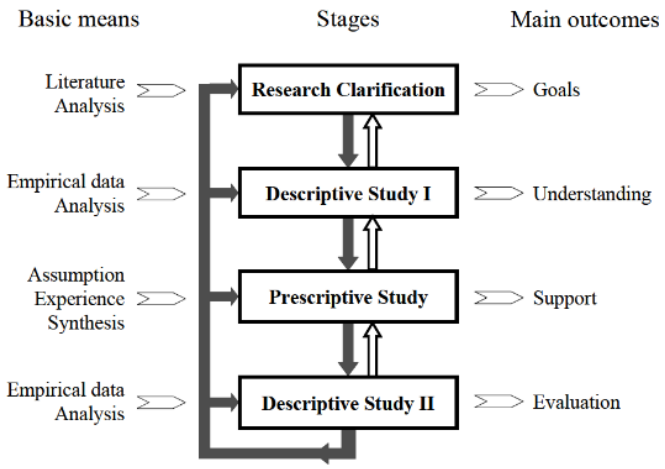


Figure 2.1: DRM framework including means in each stage and their respective outcomes. Bold arrows illustrate the main process flow and the light arrows illustrate the non-linear and iterative nature of the framework (Blessing & Chakrabarti, 2009)

The broad aim of this study is understanding a demarcated area of the existing situation (outcome of stage DS-I) and identifying appropriate support elevating this to the intended situation (outcome of the PS stage). The aim is that both outcomes should be of benefit to industry as well as academia. To ensure clear, realistic and purposeful research, several documentations have been applied: the main research problem has been clarified (Section 1.2.1); research goal and aim has been defined (Section 1.2.2); research questions has been developed (Section 1.2.3) and a hypothesis stated (Section 1.2.4).

2.4 Applied Data Collection Methods

As the research in this thesis is qualitative in nature, data collection methods are chosen to support this approach. Typical data collection methods in qualitative research include interviews, observations and documents (Creswell, 2009; McQuarrie, 2016).
Real industrial needs are best elicited from the practitioners with experience from the setting where they emerge. The large Swedish manufacturing firm studied in this thesis allowed for exploring needs and challenges at the source. With access to vast amounts of knowledge owned by the people in different parts of the organisation, interviews and observations were carried out to gather the information required. With the Department of Product Lifecycle Management (PLM) as base, access to senior experts were available as well as internal documents and databases. These informal contacts were used to gather sufficient knowledge about the phenomenon and to ensure validity of the study.

2.4.1 Interviews and Observations

Empirical data was collected mainly through interviews with employees within R&D but also from product strategy, manufacturing engineering and logistics. These were included to ensure a full coverage of the phenomenon and to avoid a too one-sided result. As the problem could be narrowed with increased knowledge, interview focus was mainly shifted towards specific design areas and related roles within R&D. Observations were carried out as a complement, being part of formal meetings and observing daily work.

Interview Sample
As the study aims at identifying real industrial needs and challenges, both senior and less experienced personnel were interviewed. This was mainly due to the fact that the occurrence of well-known problems may become normalised with experience. As knowledge and experience regarding this phenomenon diverge significantly between different roles within R&D ,a relatively large interview sample was targeted and then reduced when the result began to saturate. In total, 14 interviews were carried out with employees with 3.5 to 36 (average 16) years of experience from vehicle development.

The sample included product architecture specialists, complexity managers, business developers within PLM, product specification specialists and design engineers (system responsible and group design leaders) working in different phases of pd. Design engineers responsible for two different high variety sub-systems were selected; external RVMs and fuel lines. These two areas provided anecdotal evidence and where chosen based on their complexity and variety (along with availability of interviewees). RVMs were interesting as they are a mechatronic sub-system, highly integral, have significant commercial variety and are styling sensitive as they are visible parts. Fuel lines were interesting from another perspective, having almost no commercial variety but with extremely sensitive dependencies with its surroundings.

Interview Approach
The challenging subject in focus influenced the interview approach. To emphasise spontaneity and flexibility (McQuarrie, 2016) when conducting the interviews and to allow them to take a direction influenced by the interviewees, they were in the form of a guided conversation, that is semi-structured interviews. Due to the complexity of the subject, emphasis was on preparation. An interview protocol Creswell, 2009, p. 183 was prepared for each interview. Each protocol containing a set of pre-defined questions, with emphasis on open-ended questions (McQuarrie, 2016). Each interviewee was informed in advance, prepared with the aim, purpose and definitions of key concepts. Each interview was conducted during 45-120 minutes. To facilitate the discussion, to be able to ask probing questions (Creswell, 2009; McQuarrie, 2016) and to steer it

in a relevant direction, each interview was recorded and a mediating tool in the form of pictures were used to spark discussion. Several additional steps were followed to ensure high quality of the empirical result, as suggested by Creswell (2009): *(1)* interviews were carried out during one face-to-face session in the interviewees natural setting; *(2)* the interview data was complemented with data collected through observations and documents; and *(3)* the research and interview design were allowed to evolve as more knowledge about the phenomenon was gained.

Empirical Data Analysis
Due to the complexity of the studied phenomenon, each interview was transcribed and all relevant data was summarised and organised visually by clustering different key topics according to the *KJ method* (Scupin, 1997). This allowed further analysis and identifying relationships and correlations. Due to the divergence of the content, little statistics could be derived. The presentation of empirical data in Chapter 4 follow a narrative approach, supported by actual citations extracted from the interviews to provide validity. Within the exposition of data, conclusions and statements derived from analysis are integrated.

2.4.2 Literature Review

According to Creswell (2009), a literature review serves the purpose of substantiating the problem and identifying results from similar studies. Blessing and Chakrabarti (2009) state that it should be continuously applied throughout the project, as it extends the knowledge gained thus far. Literature can be seen as a part of the broader term *Secondary Research*. According to McQuarrie (2016) it encompasses any data collected for some other purpose, but useful for your own purposes. In addition, the author states that secondary data is not only useful when exploring the current situation, but also in generating, as well as selecting options. Apart from e.g. books and journal articles, Blessing and Chakrabarti (2009) emphasise the use of internal documents, describing the organisation, its products and processes.

Published literature and internal documents were reviewed with the aim of framing the problem and broadening the theoretical base. This included a variety of relevant subjects such as platform-based development, complexity management and product variety. The review of secondary data was continuously conducted during all phases of the research. As the research was concertised, reviews were focused on the subjects relevant for the problem areas of interest, using external documents in order to support the generation of improvement options. Internal data such as design and requirement documents were used to support the case study. Relevant research topics were used to search for literature (books and journal articles prioritised) in databases such as the *Chalmers Publication Library (CPL), Chalmers Library* and *ProQuest*.

2.4.3 Case Study

Blessing and Chakrabarti (2009) denote a case-study as a study involving data from a real setting primarily used for exploratory research or for pre-testing some research hypotheses. In this thesis a case study was performed to evaluate the proposed support framework, which was applied on the actual design of RVMs. The case study is used to test the validity of the hypothesis in Section 1.2.4. A design engineer responsible for RVMs was consulted along with existing documents (requirement specifications, 3D models, product data, etc.). RVMs were selected based on their simple essential functionality but complexity in terms of variety, mechatronic content and design dependencies. In addition, the availability of domain experts influenced the selection.

3

Theoretical Framework

This chapter presents an assortment of published theories relevant for the thesis. It aims at framing the studied phenomenon and underpinning the findings. The reader will find pertinent research findings within product variety, -configuration, -architecture, and platform-based development, as well as on variety-induced complexity, system-based product structures and change propagation.

3.1 Introduction to Product Development

Product development (pd) is a complex, interdisciplinary process that reaches from ideation of a product to its production, sales and delivery (Ulrich & Eppinger, 2012). Rather simplified, it constitutes the phases depicted in Figure 3.1. The concept phase consists of defining the technological content, specifying functions, features and performance targets in order to meet stakeholder needs. System-level design includes development of the product architecture and component embodiment. These components are further developed and tested in the detailed design phase.

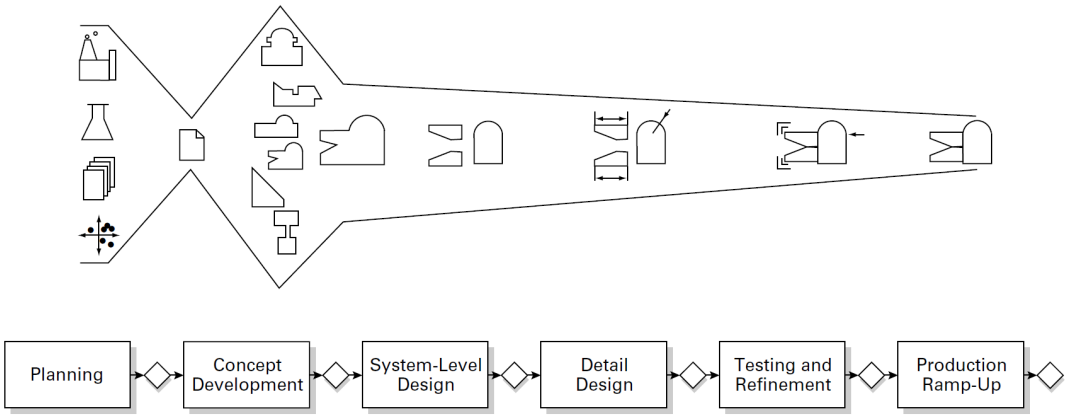


Figure 3.1: Generalised pd process (Ulrich, 1995; Ulrich & Eppinger, 2012).

Modern pd processes have become increasingly complex, iterative and non-linear as they encompass multi-domain engineering of highly configurable and complex products (Martin & Ishii, 2002). Complex products are those that consist of a significant number of interrelated systems and components characterised by a large flora of requirements (Bhise, 2013). Moreover, re-design of existing products is a pre-dominant characteristic of today's pd processes. Re-design ranges from slight modifications to face-lifts, substantial improvements and technology shifts (Gustafsson et al., 2016). In this context, Gustafsson et al. state that understanding how the product works, how its components interact and what properties the customers value the most are of significant importance.

3.1.1 Systems Engineering

Systems engineering is an interdisciplinary approach to pd that focuses on an integrated ecosystem of means that enables efficient realisation of complex systems. A system is here defined as a *'construct or collection of different elements that together produce results not obtainable by the elements alone'* (INCOSE, 2017). Systems engineering aims at supporting complex, multi-domain pd as it combines various disciplines such as mechanical development, software development, computer science and electrical development (Bhise, 2013). The advocated approach starts with a top-level understanding about the functionality that justifies the product's existence. This step is followed by requirements management, extensive design space exploration, product modelling and continuous integration (INCOSE, 2017). Modelling the product architecture is thus key, as the models are logical constructs that define pre-conditions required for efficient execution of the aforementioned tasks.

3.1.2 Product Lifecycle Management (PLM)

PLM is a business approach aiming at the enablement of efficient and effective pd (Stark, 2015). This approach includes not only product data and IT systems, but also the necessary engineering processes, methods, tools, organisations and people (Stark, 2015). If systems engineering is the strategic approach to development of complex products, then PLM provides the necessary ecosystem of tools, processes, methods and people. PLM thus focusses on integrating the heterogeneous parts of the ecosystem to ensure a consistent and seamless information flow throughout the product's lifecycle.

Product Data Management (PDM) is typically seen as an IT system for efficient management of product data (Stark, 2015). It is an integral part of the overall PLM ecosystem. The term *product data* is essentially equivalent to *engineering data*. A subset of the product data managed by a manufacturing firm is used to configure the product. This data is referred to as *product configuration data* (Tidstam, 2014).

3.2 Product Variety and Mass Customisation

Pine (1999) argued that the view of one, homogeneous market was amiss. A market is composed by individuals whose unique needs can be assessed and fulfilled. Mass Customisation (MC) is a strategy that aims at providing variety, optimally aligned with the capabilities of the company, to fulfil these heterogeneous needs. The overall aim is introducing scale benefits by reducing lead times and development costs (Landahl, 2016). Variety provides customer value if the product's functionality varies in some way (Ulrich, 1995), as the product offering then becomes more attractive (Wortmann et al., 1996). Variety will however, by nature, make even simple design problems more complex (Blecker et al., 2004). Variety may be triggered by external factors — proliferated customer preferences, competition, technology, regulation — and by internal factors — organisational or technical deficiencies leading to unnecessary variety (ElMaraghy et al., 2013).

3.2.1 Product Variety

Product variety can according to Ulrich (1995) be defined as *'the diversity of products that a production system provides to the market place'*. Anderson (1997) makes a distinction between external and internal variety. External variety is perceived by the customers while internal variety is experienced as complexity in the activities of the firm (Anderson, 1997; Du et al., 2001). The successful implementation of MC thus depends on achieving an optimal balance between low internal variety and high external variety (henceforth referred to as *commercial variety*).

Internal variety particularly refers to the variety of components, modules and products (Blecker & Abdelkafi, 2006). The balance can be illustrated as an hour glass (McKay et al., 1996), exemplified in Figure 3.2. The internal variety is represented at the so called 'neck-of-the-glass', where components in average have the highest commonality (van Veen, 1991). The number of secondary variants at the top grows exponentially with an increasing number of primary variants at the neck.

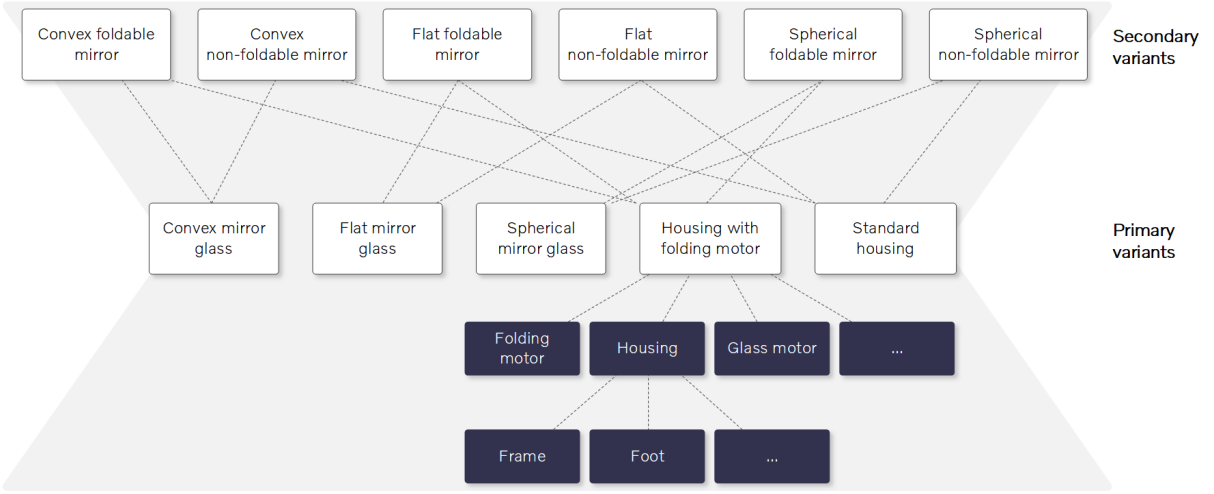


Figure 3.2: Simplified example of variety of exterior rear-view mirrors (adapted from McKay et al. (1996)) used to illustrate the balance between internal and commercial variety.

MC constitutes an interdisciplinary design problem as it encompasses optimising commercial variety both with respect to customer needs and to the industrial system. Blecker et al. (2004) identify five sub-processes that influence, and are impacted by complexity induced by variety; development, customer interaction, purchasing, manufacturing and logistics. An optimal solution space is dependent on successfully identifying the product attributes for which customer preferences diverge (Salvador et al., 2009). The *variety steering* strategy (Blecker & Abdelkafi, 2006) aims at delimiting the commercial variety to be offered within architectural boundaries. A variety steering decision can be to not offer 4-cylinder diesel engines for a new product family. *Variety management* is on the other hand a collection of strategies that promote process and component commonality. Product platforms and modularization are commonly applied variety management strategies (Blecker & Abdelkafi, 2006).

3.2.2 Product Configuration

Wortmann et al. (1996) present three different approaches to customer-driven manufacturing; *engineer-to-order*, *make-to-order* and *assemble-to-order*. What separates them is the *customer order decoupling point* which is the point where material and information flows shift from being planned (pushed) to order-driven (pulled). For variety-rich products, the all approaches have one thing in common; they need to be configured.

As originally stated by Mittal and Frayman (1989, p. 1395) configuration can be defined as a *'a special type of design activity, with the key feature that the artefact being designed is assembled from a set of pre-defined components that can only be connected together in certain ways'*. Soininen et al. (1998) expands this reasoning, letting the term *product configuration* refer to the routine, often automated, engineering activity in the sales-to-delivery process. *Configuration design* on the other hand, comprises the more design-oriented activities required to develop a configurable product. A configurable product has the key characteristic that *'product variants*

are generated by combining sets or pre-defined components, i.e. no new components are designed to address the customer's needs' (Salvador & Forza, 2004, p. 275). The ways in which components and attributes are allowed to be combined are defined by *configuration rules* (Tidstam et al., 2016). In the automotive industry, products are commonly configured and assembled in accordance to a placed customer order, adopting an *assemble-to-order* approach (Wortmann et al., 1996).

3.3 Configuration Design

Economies of scale is achieved when the same set of building blocks can be re-used in a large number of different product variants (Jiao et al., 2007; Simpson, 2004). Capitalising on similarity and commonality is thus a pre-requisite for managing variety (ElMaraghy & ElMaraghy, 2013) as *'products that use many common parts inheritably have less variety costs than products with unique parts'* (Anderson, 1997, p. 78). Systematic re-use of components is pursued through *configuration design* — a typical knowledge-based process (Brière-Côté et al., 2010). Re-usable components may be of three classes, standard components, standard component variants and parametric components (Tiihonen et al., 1996).

Standard components decrease complexity, costs and lead times. They also decrease uncertainty as they are known and thus eliminate new learning curves. However, Ulrich (1995) state that they may add to a firm's inertia due to the difficulties to integrate innovative solutions into a standardised product. Jiao and Tseng (2000) argue that, to significantly increase commonality, focus must be on maximising the utilisation of common components in high-volume product variants. This is due to the fact that commonality makes products more similar, making low-end products more desirable and high-end products less desirable (Kim & Chhajed, 2001).

Berglund and Claesson (2005) defines a system's *design bandwidth* as its flexibility to be used in different product variants. Wahl and Claesson (2010) extended this reasoning, arguing that all alternative solutions together constitute a combined system bandwidth. The internal variety of a system is thus highly solution-dependent. That is, the *functional bandwidth* of a system may be realised by fewer or more variants depending on how the system is realised. This often manifests itself in a trade-off between variety and performance or between variety and cost.

3.3.1 The Role of Product Architectures to Achieve Mass Customisation

Re-use can be leveraged through a multitude of approaches, including *modularization*, *standardisation* and *product family design and platform-based development*. All approaches are based on a well-planned product architecture which is key for capturing and utilising commonality (Du et al., 2001). Architectural decisions thus have the potential to significantly impact the cost and scope of product variety and the succeeding pd activities (Ulrich, 1995; Ulrich & Eppinger, 2012). The architecture is however defined during early pd phases, when system knowledge is scant (Raudberget et al., 2015). As the commitment to a specific architecture has significant effects on all succeeding design activities, the correct decisions at this stage can profoundly help product success.

Ulrich (1995, p. 420) defines product architectures as *'the scheme by which the function of the product is allocated to physical components'*. More precisely it encompasses how functional elements are organised and mapped to physical embodiments and the specification of the interfaces along which the physical embodiments interact (Ulrich, 1995; Ulrich & Eppinger, 2012). In essence, it corresponds to the layout, configuration or topology of the functional elements and their embodiments (van Wie et al., 2003). The notion of functional elements (or FRs (Suh,

1990)) refers to what the product does, as opposed to its physical characteristics. The latter is instead described by physical components (or *design parameters* (Suh, 1990)). Adopting the notion used by Ulrich (1995), a *component* is extended to be synonymous to a distinct region of the product which allows including software in the definition.

3.3.1.1 Variety Management through Modularization

The mapping of components to FRs is done early in development. This mapping can be of three types; *one-to-one*, *many-to-one*, or *one-to-many* (Ulrich, 1995). One-to-one mappings constitute a *modular product architecture*. This is according to Suh (1990) and his theory of complexity an axiom of a 'good' design as the interfaces between components (or *design parameters*) are uncoupled. An *integral product architecture*, which contains no or few one-to-one mappings, is more complex as the interfaces are coupled. Two components are said to be coupled if one cannot be changed without introducing a change to the other (Ulrich, 1995). Thus, the product architecture defines the functional elements which will be affected by a change to a particular component as well as the components which need to be modified in order to fulfil a functional change (Ulrich, 1995).

Product modularity enables one to enhance commonality so that economies of scale, scope and substitution can be achieved (Blecker & Abdelkafi, 2006). A module is synonymous to a physical or conceptual grouping of components that share some characteristics (Jiao et al., 2007; Ulrich, 1995). The main objective with modularization is defining such grouping so that each module may be treated as a logical unit (Newcomb et al., 1996). This includes de-coupling interfaces between components (Ulrich, 1995). Modularization may thus maximise the set of product variants that can be obtained with little internal variety.

Modularity may not only be defined at the product level but also on system or component level (Sosa et al., 2007). Sosa et al. (2003, p. 240) define a modular system (or sub-system) as one whose *'design interfaces with other systems are clustered among a few physically adjacent systems'*. A modular system is characterised by minimal between-systems interactions whereas interactions within a system may be complex (Jiao et al., 2007; Ulrich, 1995). On component level, Sosa et al. (2007, p. 1120) define modularity as the *'level of independence of a component from the other components within a product'*. The more disconnected a component is, i.e. the more *degrees of freedom* it has, the more modular it is. Modularization supports both configuration design — by extending the set of alternative building blocks — and design re-use over time, by allowing component carry-over from existing products.

3.3.1.2 Variety Management Through Architectural Flexibility

Modularization allows building complex products from smaller sub-systems that can be designed independently, yet function together as a whole (Baldwin & Clark, 1997). This requires interface standardisation (Crawley et al., 2004). Though interface standardisation, modular product architectures are given a higher flexibility than their integral counterpart. Modular architectures are driven by variety, product evolvability and service requirements while integral architectures are often driven by product performance, design or cost (Chen & Wang, 2008; Cutherell, 1988).

Architectural flexibility is required to meet future requirements and increased product complexity (Landahl, 2016). As Figure 3.3 illustrates, a flexible architecture enables re-use over time (Muffatto & Roveda, 2000). Flexibility is particularly important for complex products subjected to a highly dynamic set of requirements and customer preferences. Crawley et al. (2004) highlight that flexibility may require substantial planning and even over-designing systems.

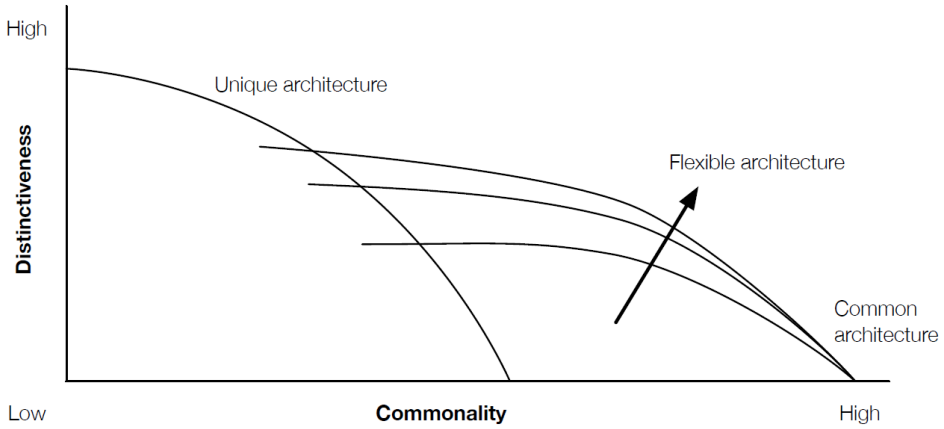


Figure 3.3: Architectural flexibility and the effects on product commonality and distinctiveness (as drawn in Landahl (2016), redrawn from Muffatto and Roveda (2000)).

3.3.1.3 Product Families and Platform-based Development

Product family design is a well-documented strategy for providing sufficient commercial variety while maintaining economies of scale and scope (Robertson & Ulrich, 1998). The commonality principle is utilised in the sense that differentiated product individuals are derived from a set of common core features, called a *product platform* (Jiao et al., 2007; Meyer & Lehnerd, 1997). While a product family is a set of similar products, a product model (e.g. a Volvo XC 90), possess specific features or functions that aim at fulfilling the needs of a specific customer segment. Product models can in turn diverge along a series of features aiming at addressing specific customer needs, resulting in a number of product variants (Du et al., 2001).

Literature definitions of a product platform vary: Meyer and Lehnerd (1997) define it as *'a set of sub-systems and interfaces developed to form a common structure from which a stream of derivative products can be efficiently developed and produced'*, McGrath (1995) sees it as the set of common elements, in particular the core technology, across a range of products; Robertson and Ulrich (1998) extend the definition letting the elements include components, processes, knowledge and even people — so called assets. The focus on platforms as a mean to re-use knowledge has increased recently (Jiao et al., 2007). Baldwin and Clark (2000) argue that a product platform has three distinct characteristics: *(1)* It has a modular architecture; *(2)* It has distinct interfaces; And *(3)* there are standards and design rules to which the modules must conform. In general, platforms have the overall aim of systematic re-use (Tidstam, 2014). A product platform aims at re-using components, a technology platform aims at re-using technology and other types of platforms aims at re-using knowledge.

There are essentially two ways to incorporate variety into a platform (Jiao et al., 2007); making them scalable (i.e. parametric) or modular. Scalable platforms can be stretched or shrunken to fit a range of customer requirements (Lewandowski et al., 2015; Simpson et al., 2001). Scaling is achieved by varying design parameters along specific dimensions. Modular platforms allow deriving product variants by adding, removing and/or substituting specific modules (Du et al., 2001; Ulrich, 1995). Varying performance may thus be achieved by scaling design parameters and/or by changing a module to another (Lewandowski et al., 2015).

The benefits of product platforms have been exhaustively examined in research (Jiao et al., 2007). Benefits include enhanced ability to upgrade products and reduced system complexity, development time, development risk and efforts spent on testing and certification (Sawhney, 1998; Simpson, 2004). The strategy however also affect the scope and structure of product

architectures (Robertson & Ulrich, 1998). Systematic planning across the product's functional, technical and structural views is required (Jiao & Tseng, 1999).

Lewandowski et al. (2015) highlight three phases of platform development: *(1)* In the *scoping phase* targeted markets and functionality are decided; *(2)* In the *design phase* the platform evolves and variety and commonality are incorporated; And *(3)* in the *execution phase* derivative products are created. A relatively large effort should be spent on the former two, fulfilling a wide range of customer needs with as little internal variety as possible. This is however complex due to uncertainty, long lead-times and the interdisciplinary nature of platform development.

New models almost always carry some functional changes relative to their predecessors (Ulrich, 1995). Pre-defined building blocks are developed with the aim of satisfying known customer needs. Thus, in business environments where technology and functional requirements are dynamic and hard to predict, platforms are sub-optimal (Landahl et al., 2014; Landahl et al., 2016; Lewandowski et al., 2015). Platforms typically live for several years, as development is costly and time consuming. Requirements might change multiple times during this period (Lewandowski et al., 2015). In order to remain competitive, the platform needs to be *maintained* to allow for new requirements and technologies (Ben-Arieh et al., 2009; Lewandowski et al., 2015). As it is difficult to plan for changed customer preferences, major changes may be difficult to avoid. Architectural stability extends the lifetime of platforms (ElMaraghy et al., 2013). But when the platform needs to evolve, stability may inhibit innovation. Flexibility is thus required to achieve efficiency over time.

3.4 Complexity

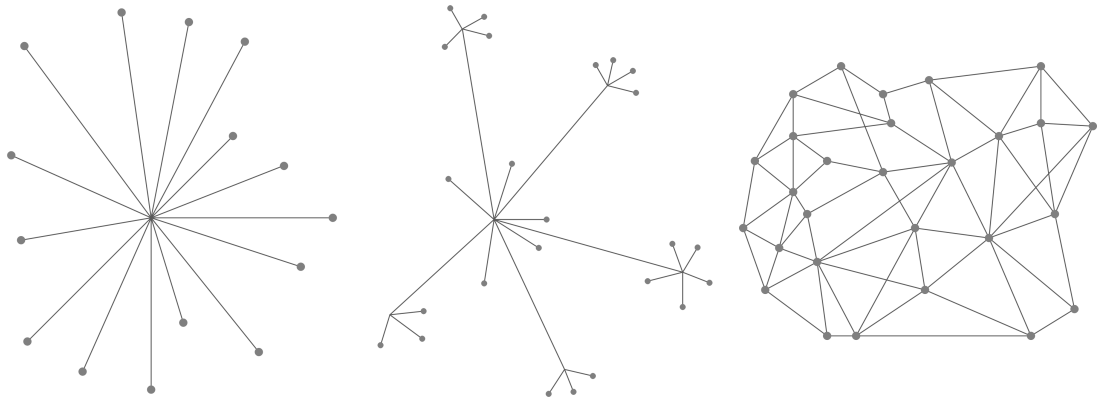
Today's engineered products are becoming increasingly complex (Landahl et al., 2014; Sinha & de Weck, 2016). This strains the efficiency of established pd practices. A major challenge is stated by Sinha and de Weck (2016, p. 10); *'the perception of complexity is observer-dependent and it varies across human subjects for identical level of underlying complexity'*. Moreover, complex systems have historically been designed preeminently based on trial-and-error and empiricism, lacking means to understand and assess complexity (Suh, 2005).

Complexity lacks one universal definition. It has been ascribed to novelty, uncertainty, ambiguity or difficulty in past research (Jacobs & Swink, 2011). For example Campbell (1988) relate complexity of a specific task to the diversity of information and to the rate by which information changes. Related to product design, the complexity theory based on axiomatic design defines complexity as a *'measure of uncertainty in achieving the specified FRs'* (Suh, 1991, 2005). Baldwin and Clark (2000) see complexity as proportional to the number of design decisions. Griffin (1997) relates complexity to the number of functions in the product. Sinha and de Weck (2016) calculate complexity of technical systems as a function of their physical elements and dependency structures. A *'system or object (tangible or intangible) can therefore be deemed complex if it is made up of a multiple of diverse, interrelated elements'* (Jacobs & Swink, 2011, p. 679).

Complexity is an inherent property of a system (Tang & Salminen, 2001) and it often correlates to increased performance. It may not itself be a negative or undesirable property. Ben-Yehudaa et al. (2015) rather see complicatedness as the negative resulting by-product of complexity. Complicatedness is thus something that should be avoided all together. The negative effects of complexity take the form of *complexity costs*, making it impossible to achieve costs close to those of mass production (Blecker & Abdelkafi, 2006). A desired characteristic of a product architecture is thus mitigation of complexity costs Sinha and de Weck (2016).

3.4.1 Structural Complexity

Sinha and de Weck (2016) define *structural complexity* as a function of a system's physical elements and their dependency structure. If for instance two systems perform the same function, the one with a higher number of components is more complex. A complex system is *'one whose properties are not fully explained by an understanding of its component parts'* (Suh, 1991). Complexity is thus mitigated if products are designed in such a way that strong dependencies are kept within modules and only weak dependencies may connect modules to each other (Ulrich & Eppinger, 2012). Structural complexity calculated based on the complexity of each individual component or module, *and* on the characteristics of the dependency structure. The former correlates to the required development effort. The latter is calculated as the product of the sum of the complexity of all pairwise interactions on the one hand and of the topological complexity on the other. Topological complexity reflects the effort required to perform system integration (Sinha & de Weck, 2013). It increases from a *centralised* to a more *distributed* product architecture, as illustrated in Figure 3.4.



(a) Centralised architecture. **(b)** Hierarchical architecture. **(c)** distributed architecture.

Figure 3.4: A system's topological complexity increase from a centralised to a more distributed architecture (redrawn from Sinha and de Weck (2016)).

3.4.2 Variety-induced Complexity

Blecker and Abdelkafi (2006) introduced *variety-induced complexity* to better distinguish complexity induced by product variety from structural complexity. It can be defined as the *'systemic effect that numerous products, customers, markets, processes, parts and organisational entities have on activities, overhead structures and information flows'* (Saeed & Young, 1998). Such complexity arises due to an increased number of product variants, increased product complexity and insufficient tools to capitalise on similarity and commonality (ElMaraghy & ElMaraghy, 2013). Blecker and Abdelkafi highlight that variety can add additional complexity to an already complex product. For instance, as the number of unique components becomes difficult to keep track of, it might be easier to develop a new component than finding an existing, suitable one. From the perspective of complexity, this additional variety is unnecessary.

Product variety induces complexity in a multitude of operations across a manufacturing firm (Blecker & Abdelkafi, 2006). Complexity costs are overhead costs that result from residual complexity (Sinha & de Weck, 2016). Residual complexity does not add additional value to the product. The value added by product variety shall always exceed the increased complexity costs. Sinha and de Weck (2016) introduced the *complexity budget* to keep track of the optimal level of complexity from a value perspective. Although related to structural complexity, it may be extended to variety-induced complexity as well. Sinha and de Weck (2013) stated that it

is imperative with an *'objective and quantifiable measure of structural complexity'* in order to explore a system's design space and optimise its architecture. It is however difficult to accurately compute complexity costs and ascribe the costs to individual product variants (Blecker & Abdelkafi, 2006). An example of indicators of variety-induced complexity used at an automotive company are disclosed in Figure 3.5.

	Development	Purchasing	Manufacturing	Sales
Focus	Technical product creation	Logistics/supplier	Plant processes	Product portfolio
Indicators	Effort per variant [Additional effort compared to product line, %]	Sequence streams [# of required logistics sequence streams]	Launch complexity [% of changed production lines in existing production]	Volume per variant [# per variant]
	Multi-use rate [% of overall parts taken from product line]	Additional article codes for required materials [# of article codes]	Plant article codes [# of new plant article codes / existing plant article codes	Standard engines [% of non-standard engines]
	Article codes per variant [# of article codes for variant]		Divergence in production sequence [% of stable takts in place and used]	Standard option packages [% of non-standard packages]

Figure 3.5: Indicators for the evaluation of value chain process complexity, (Götzfried, 2013).

A quantifiable measure of complexity costs combined with systematic design space exploration can support identification of favourable or non-favourable architectural variants. Blecker and Abdelkafi (2006) suggest using key metrics for detecting variety-induced complexity in the pd process, for instance: The *multiple use metric* (Ericsson & Erixon, 2000) measures the number of product variants requested by customers compared to the total number of modules; The *part commonality metric* (Jiao & Tseng, 2000) measures the extent to which products are built around common parts; The *average platform cycle time efficiency* Meyer and Lehnerd (1997) measures the average time required to develop a derivative product divided by the time required to develop the platform.

Variety steering aims at reducing the number of variants while *variety management* allows, yet controls, variety (Landahl et al., 2014). Landahl et al. conclude that limiting variety only has short-term benefits. Moreover, established variety management practices seldom consider the evolution of customer needs and technology. As they evolve, stable product architectures may be outdated (Landahl et al., 2014). When products based on a platform are forced outside the platform's design boundary, complexity management practices fail. This originates from a too narrowly defined platform. Platform flexibility and adaptability are thus important factors to consider when quantifying and managing variety-induced complexity.

3.5 Complexity and Engineering Changes

Most product platform research assume that systems may easily be changed (Ferguson et al., 2014). Changes are however rarely isolated; other components are usually influenced (Clarkson et al., 2004). Such changes may have severe effects if not managed properly (Fricke et al., 2000; Maier & Langer, 2011). With increasingly complex products, management of the propagation and impact of *Engineering Changes (ECs)* is likely to become increasingly important (Wynn, Caldwell, et al., 2010). Late ECs are known to be very costly (Landahl et al., 2016; Terwiesch & Loch, 1999). The alternative however — sticking with the current design — is seldom viable as it risks impacting sales negatively (Schuh & Company, 2015).

ECs arise throughout a product's lifecycle and are a natural part of most pd processes (Terwiesch & Loch, 1999). Hamraz, Caldwell, and Clarkson (2013b) define them as:

> *'Changes and/or modifications to released structures (fits, forms and dimensions, surfaces, materials, etc.), behaviour (stability, strength, corrosion, etc.), function (speed, performance, efficiency, etc.), or the relations between functions and behaviour (design principles), or behaviour and structure (physical laws) of a technical artefact.'*

Structure defines what an artefact is, behaviour describes what it does and function defines what it is for.

ECs can be triggered by external factors such as new technology, regulations or customer needs (Hamraz, Caldwell, & Clarkson, 2013a; Terwiesch & Loch, 1999; Ulrich, 1995). They can also be triggered by internal factors such as cost savings or performance improvements (Terwiesch & Loch, 1999). It is useful to distinguish initiated changes — those arising from changing customer requirements — from emergent changes — those emerging due to a weakness in the architecture (Clarkson et al., 2004; Ferguson et al., 2014). As change is inevitable, their management is key as companies that develop complex products strive to avoid their negative effects on development time and cost (Huang & Mak, 1999; Terwiesch & Loch, 1999). According to Gustafsson et al. (2016), pd is in practice not always characterised as much on fact-based decisions as on rules of thumb and experience. The result is a lack of sufficient knowledge about how changes may affect the product's behaviour. A risk multiplied in the face of variety.

3.5.1 Design Dependencies

Many studies have been devoted to modularization in practice (Jiao et al., 2007). Many of which study *design dependencies* — a specific kind of connection between two components. Suh (2001) identifies design dependencies between functional elements of a product in terms of flows of energy, material and information. Other work, e.g. Sosa et al. (2003), focusses on dependencies between physical components based on how the functionality of one component depends on spatial, structural, material, energy and information constraints of others.

Interfaces are formed by aggregating design dependencies of various types (Sosa et al., 2007). An interface is a connection between two entities that exist only to serve certain functions (Bhise, 2013). Interaction along interfaces may be of different types; physical, energy, material or information (Pimmler & Eppinger, 1994). Meyer and Lehnerd (1997) established the importance of interface standardisation. The design of robust interfaces was highlighted by Steward (1981) as a mean to standardise interfaces across product variants. Internal variety is inevitable if interfaces are not specified and designed properly (Blecker & Abdelkafi, 2006).

The dependency structure of any technical system can be represented by a *DSM* (Sosa et al., 2003). For example Yu et al. (2003) applied DSMs to identify highly interactive groups of product elements and to cluster them as modules. The DSM is a matrix-based, graphical method introduced by Steward (1981) to model the structure of product architectures in terms of design dependencies between components (Sosa et al., 2007). Sosa et al. used DSMs to calculate modularity based on dependency patterns between components. This approach takes into consideration, not only components directly connected, but also those indirectly connected. This is relevant as dependencies have the potential to propagate through intermediate components and reach more distant ones.

Sosa et al. (2007) studied the effects of modularity on re-design. This relationship is highly dependent on the interfaces and design dependencies. Sosa et al. (2007, p. 1126) define *com-*

ponent re-design as *'the percentage of actual novel design content relative to the design of the component included in the previous version of the product'*. According to Sosa et al., *unplanned re-design* (a result of emergent changes) should be avoided as it is a wasteful activity, especially for highly coupled components:

> *'Because components are connected through various types of design dependencies, design changes in one component are likely to propagate to other components in the product. As a result, a component that depends (directly or indirectly) on many other components or one in the midst of many other components is more likely to be in need of re-design to accommodate unforeseen design changes'.*

Modularity fosters innovation as planned re-design is easier if systems are less coupled (Baldwin & Clark, 1997, 2000). Even so, in environments where decisions are unlikely to be based on facts, re-design can have unpredictable outcomes (Gustafsson et al., 2016). When tasked with the development of complex products, design engineers need access to powerful tools to provide them with a clear view of the road ahead.

3.5.2 Change Prediction

When systems are developed in parallel, changes in one may affect the others (Lewandowski et al., 2015). This have implications when the systems involved have varying levels of maturity. This is the case in products with a high degree of re-use. *Change propagation* is the phenomenon when changes to one component lead to knock-on effects on multiple other components, close or distant, which ultimately result in new, emergent changes (Clarkson et al., 2004; Fricke et al., 2000; Hamraz et al., 2015; Terwiesch & Loch, 1999). Change propagation is enabled by high interconnectivity in products. It isa key challenges managing ECs (Hamraz, Caldwell, & Clarkson, 2013a; Hamraz, Hisarciklilar, et al., 2013).

Change propagation will inevitably lead to unplanned re-design. Even if the original change is known beforehand, the knock-on effects might lead to unpredicted changes. Changes are more likely to propagate to and through components with a large number of dependencies. Such components possess higher risks of being re-designed and may require significantly larger re-design efforts than anticipated (Sosa et al., 2007). Components with few dependencies are more suitable candidates for re-design.

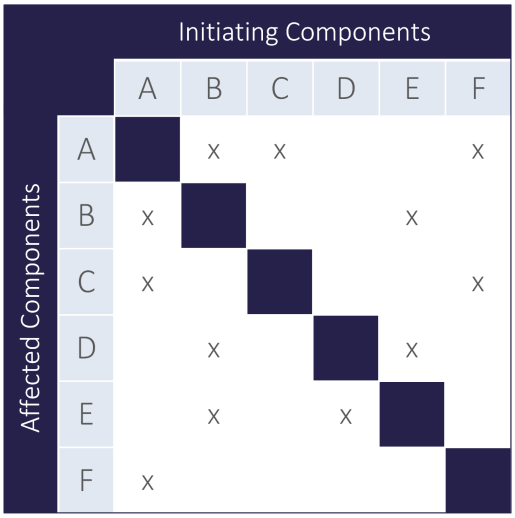


Figure 3.6: Component-based DSM expressing the components initiating a change as columns and the components affected by a change as rows. Design dependencies between component pairs are noted with an 'x'.

Many methods and tools have been proposed to manage ECs (Hamraz, Hisarciklilar, et al., 2013). For a comprehensive review, see Jarratt et al. (2011). The *Change Propagation Methodology (CPM)* uses a model of component pairs and their respective design dependencies to compute the overall risks that changes to one component will propagate to the other components in the system (Clarkson et al., 2004).

CPM have the potential to reveal parts of a system not directly affected by a change, but significantly influenced (Raudberget et al., 2015). The direct inbound and outbound dependen-

cies between components are modelled in component-based DSMs, as illustrated in Figure 3.6. CPM is built on the fundamental assumption that a change might only propagate from one component to another if they are directly linked to each other (direct change propagation) or if several intermediate components create a path between them (indirect change propagation) (Hamraz et al., 2015). The CPM is available in the software tool Cambridge Advanced Modeller (CAM) (Wynn, 2007; Wynn et al., 2009; Wynn, Wyatt, et al., 2010).

CPM requires quantified design dependencies in terms of how likely a change is to propagate from one component to the other, and the impact if it does. Change impact quantifies how severe the effects would be if the change propagates to the linked component, i.e. the effort of re-design. Both figures are estimations between 0 and 1 (0 means no likelihood or impact and 1 means maximum likelihood or impact). Such estimates typically require vast expertise or experience of the involved design engineer.

The first step of change propagation analysis is decomposing the system into sub-systems. Decomposition is a mean to break complex systems down into more manageable parts. The second step is defining and quantifying direct design dependencies between the lowest level components (inbound, outbound or two-way) using a binary DSM (Hamraz, Hisarciklilar, et al., 2013). These components can be physical or abstract representations. The direct risk of change propagation is computed as the product of direct likelihood and direct impact (Clarkson et al., 2004). CPM then calculates the combined risk (the sum of direct risk and indirect risk) of change propagation using a numerical search-based algorithm. The resulting *combined risk matrix* provides a quantified view of the overall risks of change propagation in the system, Figure 3.7.

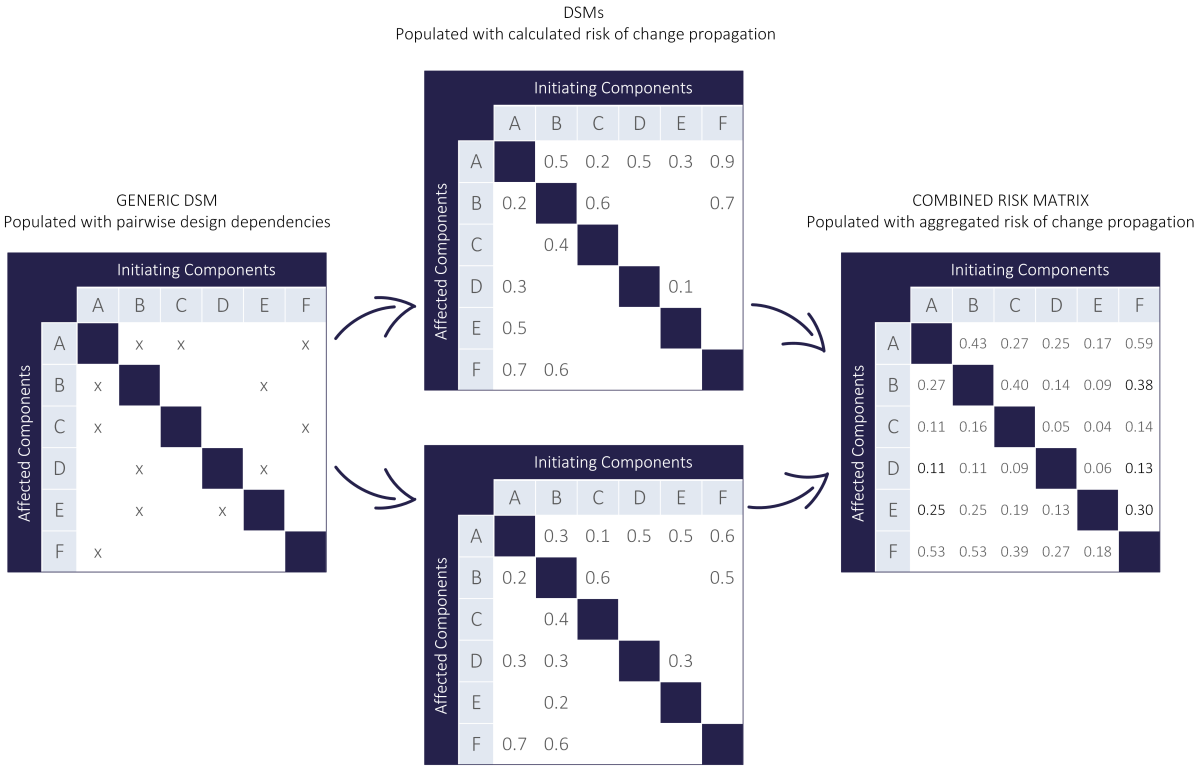


Figure 3.7: The generic steps of CPM (applied from Belt et al. (2015)).

By creating a DSM of each system variant, CPM may assess change patterns across the system's configuration space. A set of system variants or a change to a delimited part of the system can be evaluated against three criteria (Raudberget et al., 2015): *(1) Ability to integrate*; *(2) Design*

process efficiency; And *(3) development process efficiency*. These criteria are important when assessing the cost and complexity of re-design.

Ability to integrate expresses how difficult it is to change the system or to integrate it in a new context. It is calculated based on the system's level of coupling. Less coupled systems are good candidates for re-use as they are easier to integrate in new contexts. Design process efficiency expresses the complexity of an architecture with respect to design iterations. The calculation is based on the number of interactions that are affected by a change. Development process efficiency expresses the inherited risk of the architecture, represented by the effort required to re-design the system in order to mitigate the risks of change propagation.

With CPM, the design space of a product architecture can be explored based on topological complexity, overall risk, robustness and on identified change absorbers and change multipliers. Not only when solutions are embodied as components, but also during pre-embodiment (Raudberget et al., 2015) phases. In particular, the methodology may be used to optimise the product architecture as well as to manage planned and emergent ECs (Hamraz, Caldwell, & Clarkson, 2013a; Hamraz et al., 2015). Hamraz et al. (2015) proposes extending the methodology, using a multilayer network composed of functional, behavioural and structural attributes in a multi-domain matrix as input to CPM. This allows predicting how changes propagate not only between components but also between components and behavioural and functional attributes such as the weight or aerodynamic drag of a complete system.

3.6 Modelling Configurable Products

Product models are representations of the various information that describe a product. A product model may consider a specific aspect, dimension or viewpoint of a product, depending on the requirements of different business functions (Tidstam, 2014). Collier (1999) introduced twelve different product models in his framework *the twelve-fold way*. Andreasen (1992) introduced another framework, called *the chromosome product model*, which is a system-oriented approach to product modelling. It contains four distinct models; *product specification*, *function structure*, *organ structure* and *component structure*. The *Theory of Technical Systems* presented by Hubka and Eder (1988) provides the foundation for this framework. The theory states that a technical system only exists in order to realise a transformation from input to output. Moreover, it establishes that the four aforementioned models are required to fully describe a technical system and its development process, as functions and solutions are developed in parallel. A key characteristic with the chromosome model is the causal relationships between the different models, which allows one to trace the origins of design characteristics. Casual relationships are present as requirements define a function, as a function is solved by an organ and as an organ is embodied as a component (Hubka & Eder, 1988).

A similar theory is the F-MT (Svendsen & Hansen, 1993; Tjalve, 1979), a well-known methodology for functional decomposition of systems as well as for concept generation (Raudberget et al., 2015). The F-MT is a decomposition of the product's functions (also called FRs) and the corresponding means (also called *organs* or *DSs*) including their casual relationships. It is a graphical representation of Hubka's law as it models an architecture in the function-means domain, Figure 3.8.

3.6.1 Product Modelling for Knowledge Creation

Re-use of design information promotes efficient and effective pd (Gedell & Johannesson, 2013). Rich information flows are needed to enable re-use of knowledge and developed assets (parts,

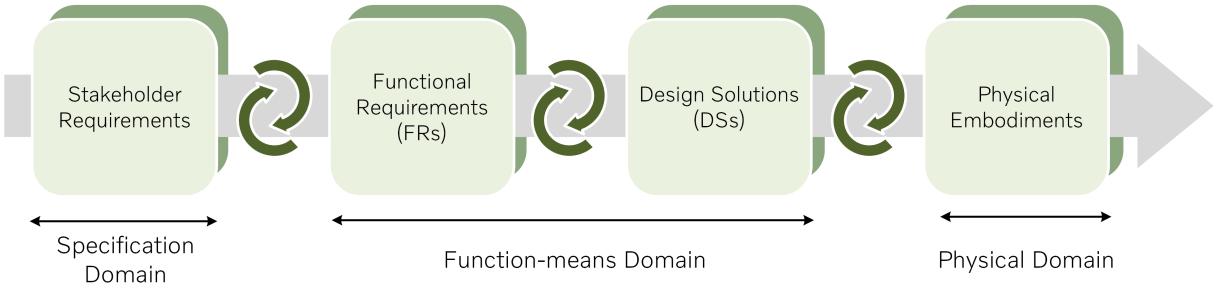


Figure 3.8: Illustration of the chromosome product model introduced by Andreasen (1992) as used by the F-MT.

components, models, documents, etc.). Landahl (2016) argues that flexible design re-use in early pd phases requires knowledge about the functionality of systems, their interaction patterns, their limitations and how they came into existence. That implies information about, not only a system's physical design, but also *why* the design exists in the form it does and the functions it realises (Gedell & Johannesson, 2013). Such information enables re-use as it contributes to an understanding about how a design may be changed without losing desired behaviour.

The arrangement of FRs and their interrelations is referred to as a *function structure* (Ulrich, 1995). This structure allows decomposition of complex systems without losing context. New product platforms are traditionally based on physical representations of historic platforms (Raudberget et al., 2015). By modelling product architectures in the function-means domain, new platforms may instead be derived from this information-rich representation. Assessment of product platforms in the function-means domain rather than in the physical domain open up for new ways of developing and evaluating the architecture earlier than in traditional pd (Raudberget et al., 2015).

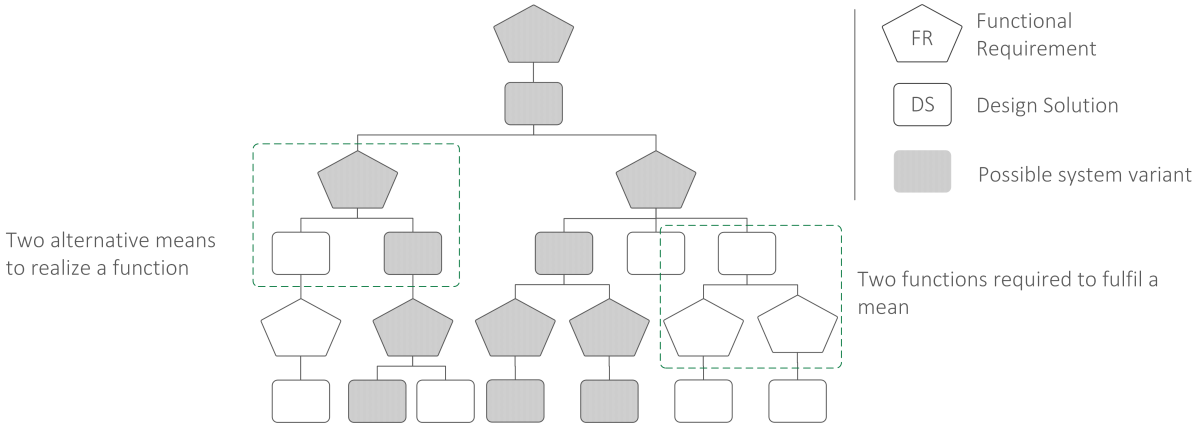


Figure 3.9: Example of a generic F-MT with alternative DSs to specific FRs allowing multiple system variants to be derived (redrawn from Raudberget et al. (2015) originally applied from Svendsen and Hansen (1993)).

Visualisation of a phenomenon is useful for human beings in multiple ways (Gustafsson et al., 2016); providing overviews, identifying details and patterns. The F-MT allows visualising a system's function and organ structures. The technique consists of a systematic, hierarchal mapping of the DSs fulfilling a FR, and the subsequent FRs implied by the selection of a DS. One *main function* is defined at the top. This FR can be fulfiled by one *mean* (DS). Each DS is further decomposed into two or more FRs that need to be fulfiled, see Figure 3.9. The technique is consistent with the principles of axiomatic design. The mapping of DSs to FRs is strictly 1:1. A FR may however be fulfiled by alternative, mutually exclusive DSs. In that case, each

DS represents a system variant — including or excluding a complete branch of the tree. The appearance of the tree is thus highly dependent on the means included in a variant, as different DSs will render different lower-level FRs. Each variant renders a unique dependency structure, which can be further analysed.

3.6.2 Product Structures

Product structures model the technical representation of a product. A product structure can be defined as a composition of elements and their interrelationships, together describing a structural aspect of the product (Brière-Côté et al., 2010; Svensson & Malmqvist, 2001). They are traditionally hierarchal classifications of the parts comprising the product (Brière-Côté et al., 2010). For configurable products, product structures typically employ a *two-level configuration* (Haag, 1998). The top level is feature-oriented and relates to the choices made by the customer. The lower level is item-oriented and relates to the implied assets, e.g. parts and components (Tidstam & Malmqvist, 2011).

The feature level — the product specification — describes each possible product variant as a list of allowed *feature variants* (Mather, 1982). Individual feature variants describe varying functional or non-functional attributes from a technical viewpoint (Tidstam, 2014). *Feature families* are used to group feature variants that describe similar attributes together. For instance, the feature family *exterior colour* may include the feature variants *black* and *white* (Tidstam, 2014). *Configuration rules* are logical constructs that define how feature variants may be combined into buildable product variants. A configuration rule can define if the selection of a specific feature variant requires the inclusion or exclusion of another (a 20-inch wheel can for instance exclude a 19-inch rim). Configuration rules typically use propositional logic, relating feature variants with the logical connectives; NOT, AND, OR, IF-THEN and IF-AND-ONLY-IF (Tidstam, 2014).

3.6.2.1 Product Configuration Data

The formalisation of product configuration data (feature families, feature variants and configuration rules) is a natural part of configuration design. With product configuration being an automated activity following a customer order, the set of rules must be completed and validated before any products can be released. Many industrial companies have dedicated roles responsible for the development of configuration rules (Mesihovic, 2004), henceforth referred to as *product specification specialists*. It is however the design engineers who are responsible for detecting if the rule set needs to be modified (Tidstam et al., 2016).

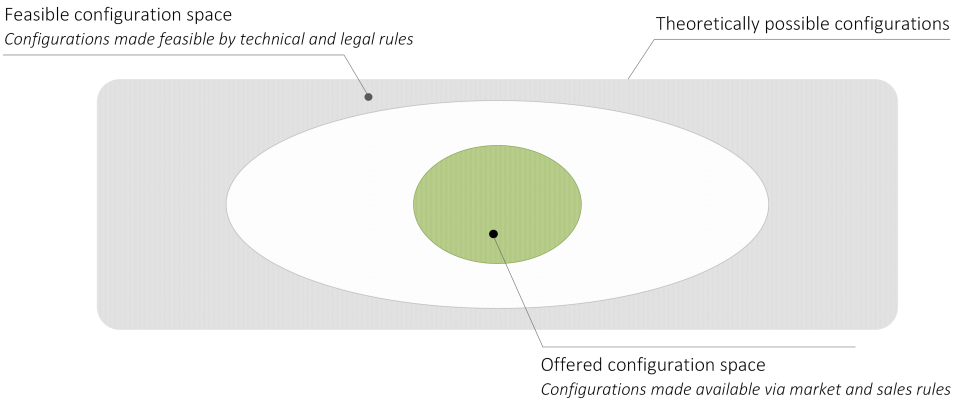


Figure 3.10: Configuration rules are authored in order to constrain the theoretically possible configuration space into a configuration space offered to customers.

The process of developing configuration rules consists of capturing informal knowledge from *domain experts* and formalising it, a specific type of a *knowledge acquisition process* (Ferguson et al., 2014). Domain experts have expert knowledge about a specific domain such as engines or seats (Mesihovic, 2004). With knowledge about technically or legally infeasible combinations of components, rules are authored to limit the configuration space, Figure 3.10. Rules are however also requested by other departments such as marketing, finance and product planning. The result is the configuration space finally offered to the market place.

3.6.2.2 Product Structures for Platform-based Development

In platform-based development, product structures need to support individual product variants derived from a common platform. Traditional product structures (e.g. the *generic Bill of Materials (BOM)* introduced by van Veen (1991)) however lack in their ability to represent complex structures (Lewandowski et al., 2015). The *CC* approach (Figure 3.11), originally developed as a meta-model for platform modelling by Claesson (2006), is a *system-oriented product structure* based on design theory (Andreasen, 1992; Hubka & Eder, 1988). It can model modular as well as scalable product platforms. It models platforms based on sub-systems and concepts rather than physical parts. Each sub-system can be configured to fit in a number of system environments. Each sub-system is described in terms of its *bandwidth* and a motivation to why it exists and looks the way it does — its *Design Rationale (DR)*.

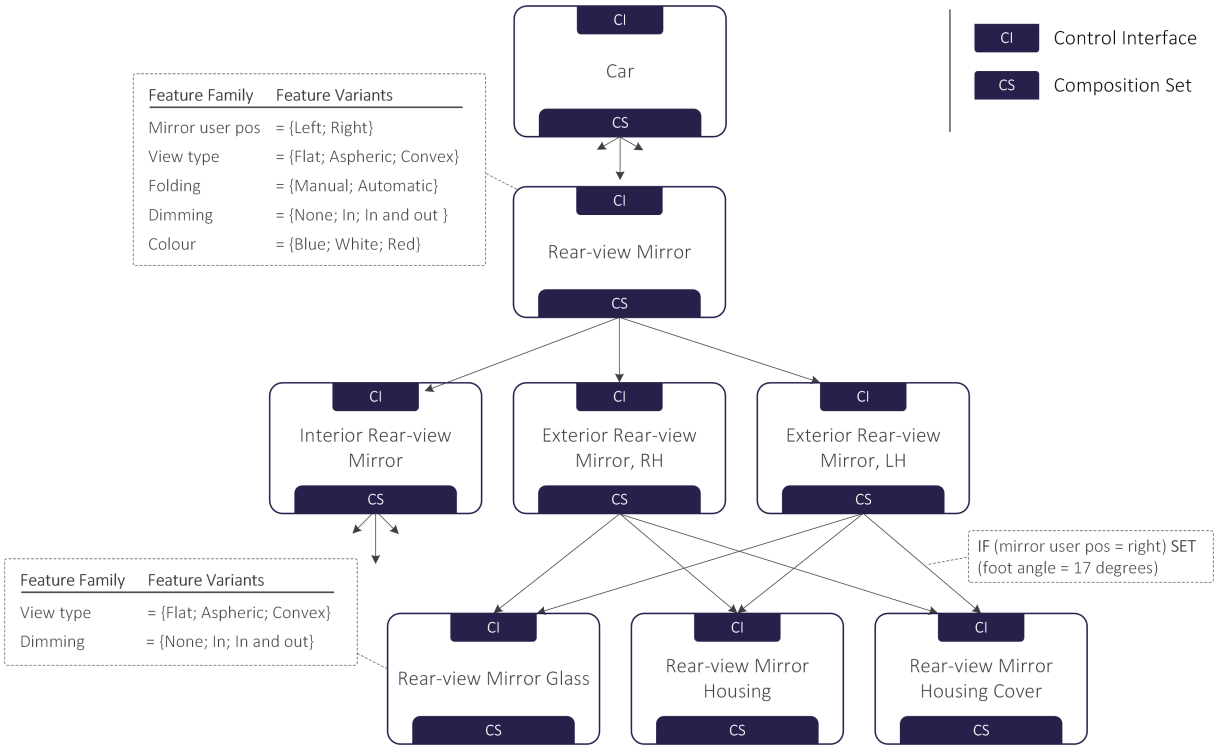


Figure 3.11: The CC approach (adapted from Tidstam (2014) originally adapted from Claesson (2006)).

The CC approach models a product architecture as a network of sub-systems; CCs. Each CC may be decomposed into several other CCs. Each one is generic in the sense that it represents every possible variant of the sub-system. The *Variant Parameters (VPs)* (or feature families) define in which dimensions the CC can vary and the *Variant Parameter Values (VPVs)* (or feature variants) define its bandwidth. Configuration rules reside in the CC which means that each CC can be configured autonomously. The CCs is instantiated by assigning one value to each of its VPs. Each CC instance thus represents one variant.

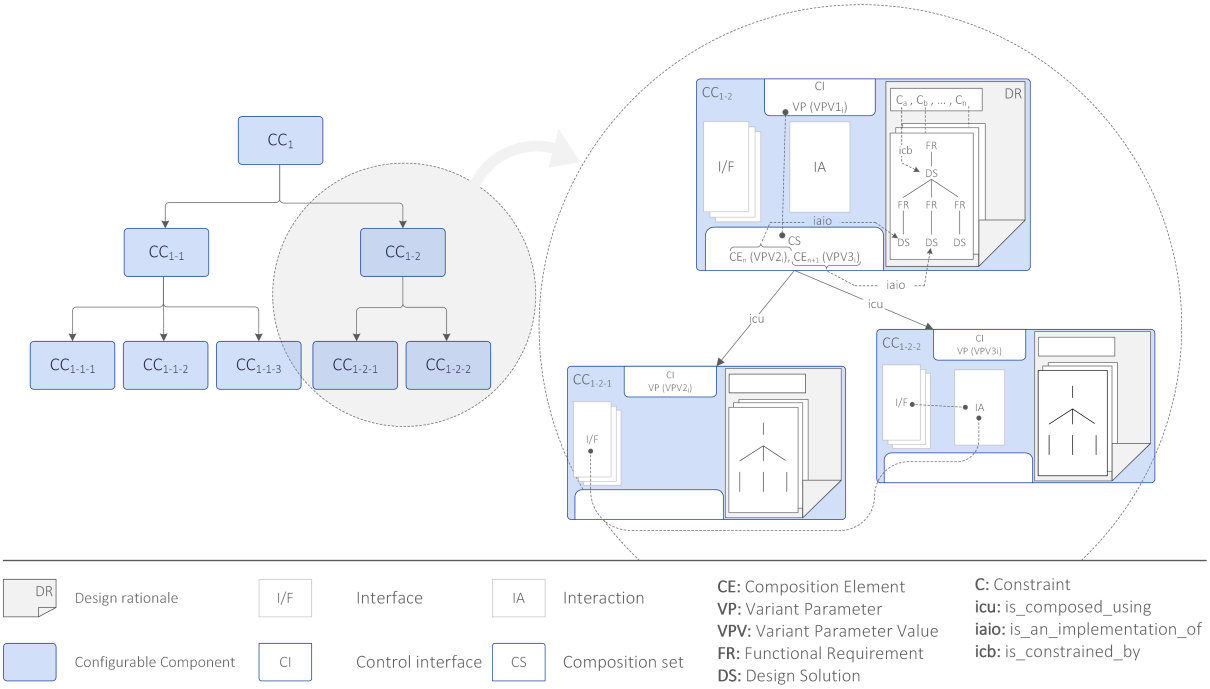


Figure 3.12: The building blocks of a CC (redrawn from Lewandowski et al. (2015), adapted from Claesson (2006)).

The building blocks of a CC are depicted in Figure 3.12. The *Control Interface (CI)* makes sets of VPs visible to the overall system. The CIs allow interchanging VPs hierarchically between CCs. Each *Composition Element (CE)* has a pre-defined *parameter set* that defines the bandwidth of the element itself. This rationale thus defines the actual number of CC instances required to fulfil the bandwidth of the CC. How VPVs may be combined is defined by *parameter maps*. They are thus equivalent to configuration rules.

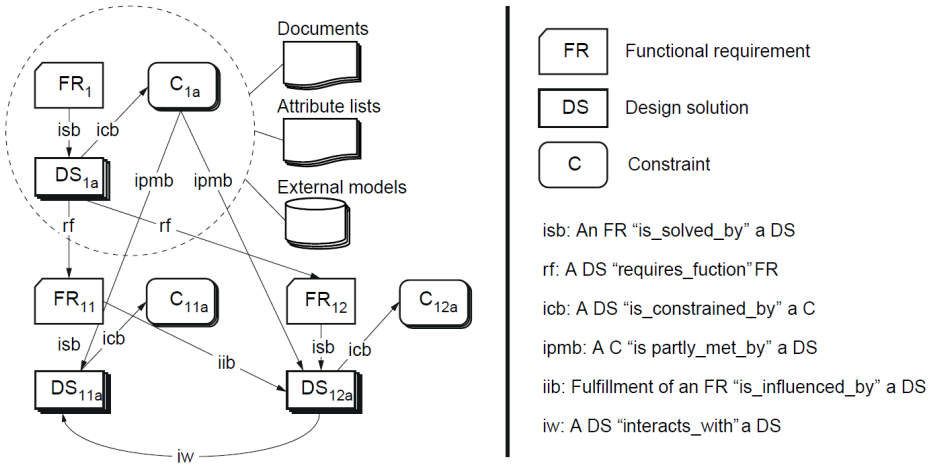


Figure 3.13: The building blocks of an EF-M tree with the possibility to attach external models and documents describing the DSs, FRs and constraints (Lewandowski et al., 2015).

The *DR* describes how the sub-system is designed. This is represented by an EF-M tree (Schachinger & Johannesson, 2000). The methodology is an extension of the F-MT as additional

objects, such as Constraints (Cs) (capture requirements that constrain the solution space) have been introduced. Interactions between DSs have also been introduced to model design dependencies. The building blocks of an EF-M tree are shown in more detail in Figure 3.13. As with F-MT, a system is decomposed into FRs and each FR is realised by a DS. As a FR may be fulfilled by alternative, mutually exclusive DSs (Figure 3.14) the DR is itself configurable. Each DS is set to *true* in a configuration if and only if a specified set of VPVs are included in the configuration, Figure 3.9. Moreover, VPVs may be used to control the existence of specific FRs. The CC approach thus supports configuring the entire system-of-systems as well as individual sub-systems.

EF-M includes relationships that express axiomatic couplings and design dependencies. A semi-lateral relationship, *is influenced by (iib)*, expresses that a DS is influenced by the FR of another DS and thus indicate a coupled design. Moreover, a lateral relationship, *interacts with (iw)*, expresses design dependencies between DSs on the model's lowest hierarchal level. These interactions allow modelling design dependencies that may have either positive or negative effects on the performance of the system. The configurable *interface* object of the CC is a specialisation of the DS object committed to representing interactions that spread from one CC to another (Lewandowski et al., 2015). Interfaces may be configured using VPVs. The *interaction* objects serve as the communicator between different CCs and thus bridge interfaces from different CCs (Lewandowski et al., 2015).

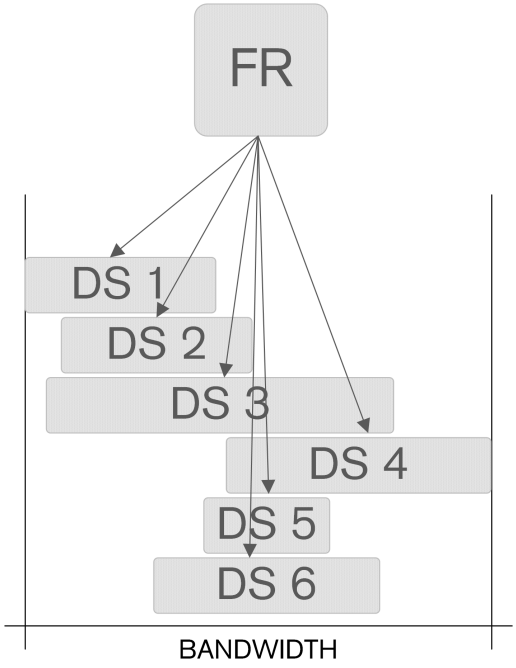


Figure 3.14: Each FR may be realised by a number of alternative, mutually exclusive DSs. Each DS fulfils a specific part of the complete functional bandwidth. The number of system variants required to satisfy a functional bandwidth is thus highly solution-dependent (redrawn from Lewandowski et al. (2015)).

Through the design rationale, the CC approach support three out of the four models requested by the chromosome product model — the product specification, the function structure and the organ structure. Moreover, it allows relating physical parts to each CC, making it possible to extend it to incorporate the physical structure as well. An EF-M tree can serve as either an abstract or a concrete representation of a system, depending on the amount of knowledge at hand. As knowledge about the system is obtained, additional hierarchical levels and interactions can be added.

4

Empirical Data Presentation and Analysis

*This chapter presents a summary of the results derived from analysing the gathered empirical data. The findings outline the context of the studied phenomenon and the identified organisational symptoms, clarified as to answer **RQ 1**. This is completed with the interviewees' experiences of complexity, clarified as to answer **RQ 2**.*

4.1 Business Environment and Complexity Drivers

'We're still at a point where we try to identify complexity (. . .) there's no panacea for all cases, every system has its own story'

- Product Architecture Strategist, R&D

This section describes the research environment, in particular the extent of variety and the strategies implemented to manage the resulting complexity. It aims at providing the reader with an overall context for the identified challenges of variety-induced complexity.

Variety is, in a general sense, increasing as new car models are introduced at a steady pace. Of the interviewees directly experiencing the effects of variety, 73 per cent (8 out of 11) perceive an increase. This is supported by the recent growth of offered configurations, Figure 4.1.

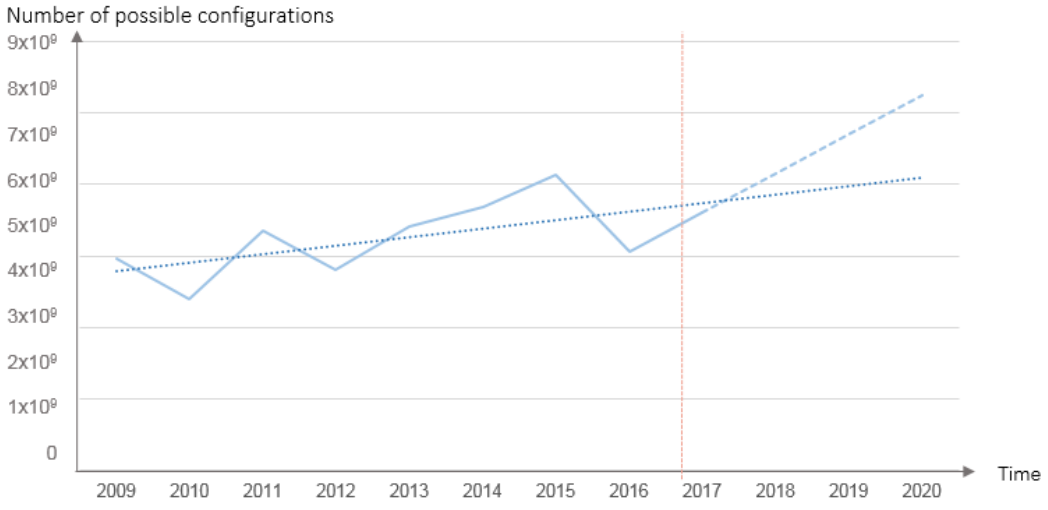


Figure 4.1: Approximation of the number of vehicle configurations offered since 2009 (based on data in the company's PDM system), with interpolated increase.

All interviewees indicate that the importance of managing complexity and optimising product

variety have gained recognition during recent years. Most efforts are however still carried out with local practices, dispersed within different organisational functions. At the same time, the organisation is changing significantly. A new pd paradigm has found its place due to recent industry and company transformations.

As a small company, development was traditionally cyclic. A large project organisation took development from start to finish, essentially starting over with *a blank page* before a problematic complexity debt had been built up. Today the situation is very different; systems live longer, architectural decisions need to be made with more foresight and architectures will evolve continuously, module by module, rather than being completely re-designed at the start of each cycle. This new paradigm imposes new challenges, as expressed by a product architecture strategist:

> *'A domain specialist must prepare a new system for production at the same time as he develops the next generation of the same system, he hasn't really done that before (. . .) he thus needs to understand his system and how it interacts with different sub-systems at a whole other level than before'.*

The company's pd process is thus characterised by challenges not only directly imposed by variety-induced complexity. The very same strategies in place to manage complexity impose challenges as well. Effective complexity management is thus of paramount importance, more so than before. The following sections describe identified drivers of variety and the strategies in place to manage complexity.

4.1.1 Drivers of Variety

The current situation can be described by a network of influencing factors, Figure 4.2. These internal as well as external factors triggering variety are described in the following paragraphs. Four factors are targeted in this study; 'quality pre-conditions', 'waste and re-design', 'level of innovation' and 'percentage of time spent adding value'. Value adding activities are those that provide value to the customer and do not include e.g. testing, verification and certification.

4.1.1.1 Commercial Variety

> *'We think we know what our customers want, but we don't'*
>
> - Complexity Manager, R&D

The main drivers of commercial variety are proliferated customer needs and legislations, with contribution from emerging technology and mimicking competitors. This drives variety-induced complexity, but also allows the company to take market shares. It is however a balance between what the markets 'want' and what the production system is capable of suppling which defines the configuration space the organisation needs to develop.

An experience shared by the interviewees is a general lack of market understanding. The interviews are unanimous in that the company is lacking in its ability to predict the real demand when planning for new product families and models. The result is *excessive commercial variety*, i.e. variants developed with a demand not justifying the costs. This experience is supported by Figure 4.3 which indicates a substantial percentage of low-volume variants. One interviewee highlights cases of specific configurations appearing only once a week or once a month (approximately one per 4,100 and one per 18,000 produced vehicles, respectively) on the assembly line. Two root-causes to this symptom are identified:

- ***Imprecise predictions:*** Forecasting demand is made complicated by dynamic customer preferences, long time-to-market, deficient involvement by the sales organisation in product planning and new competitor offerings.

- ***Cost and value discrepancy:*** Low-utility-low-cost variants are developed with little or no intentions of ever being sold. These are offered to increase the value perception of the high-profit configurations and to provide competitively priced product variants.

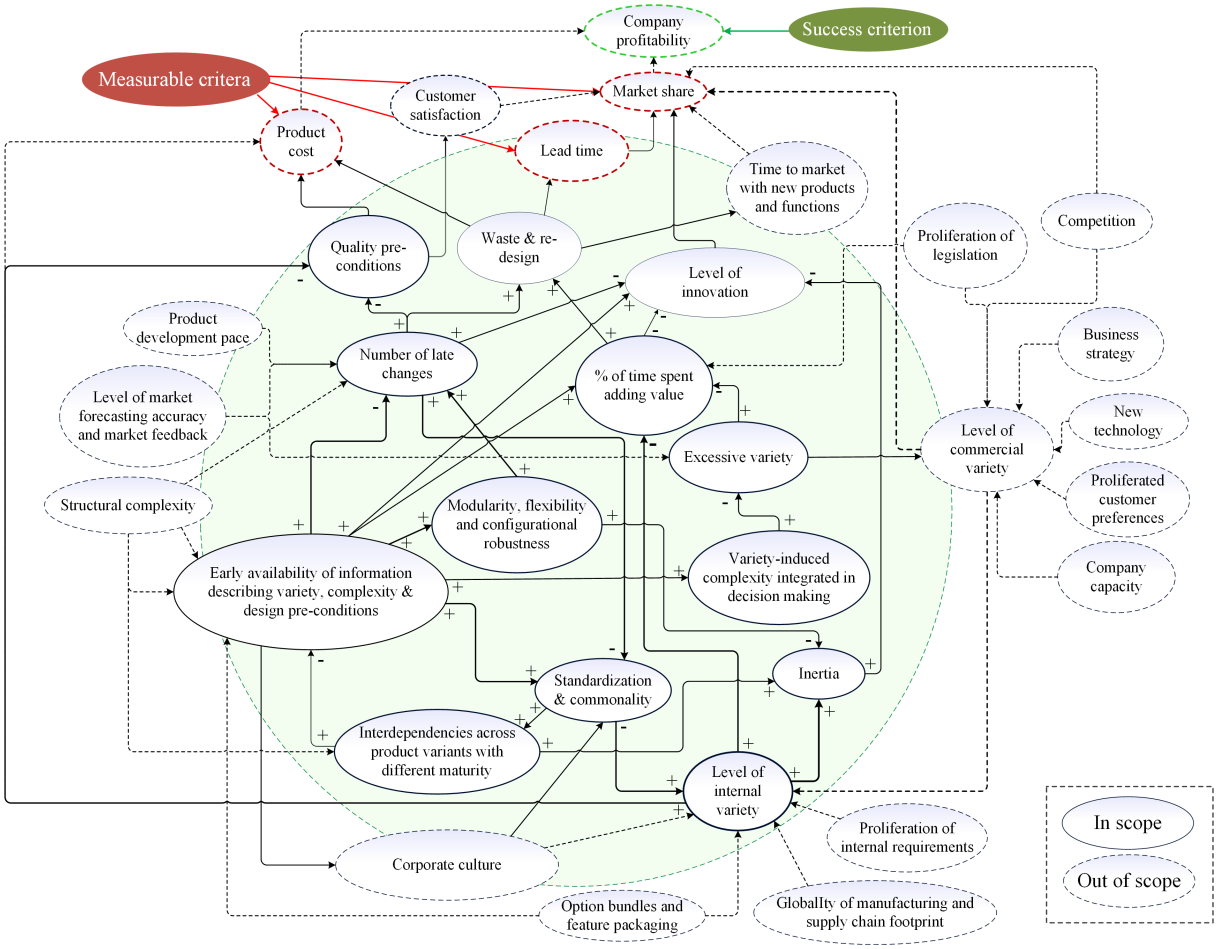


Figure 4.2: Reference model, describing the research environment as a network of influencing factors (applied from Blessing and Chakrabarti (2009)). The plus and minus signs indicate the direction of the relationship.

A majority of the interviewees identifies excessive commercial variety as one of the key issues as it contributes to waste (resources spent on developing, testing, certifying, preparing production and planning supply chains for variants not adding revenue) and late ECs:

'These variant reductions or additions often arrive very late since we realise that; "oh we need to offer this as well", or; "well we can remove this offer" '.

4.1.1.2 Internal Variety

'I find that we talk about variant reductions but in reality we're heading in the other direction'

- Manufacturing Engineer

The key to successful MC is internal variety with minimal complexity, fulfilling its commercial counterpart. In practice there are however multiple factors that counteract this goal.

Sub-optimal Decision-making

Designing sub-systems with only their short-term context in mind, making decisions decoupled from overall strategies and other projects, implementing 'panic' fixes to solve quality issues.

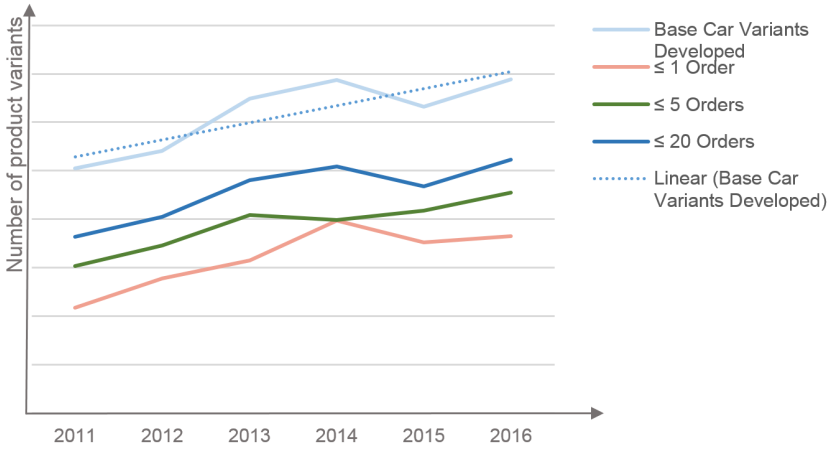


Figure 4.3: Number of base car (completely configured vehicle excluding interior/exterior colour variants and options) variants developed versus order volume (based on internal studies).

These are examples of decisions that trigger excessive internal variety. Specific examples include; isolated cost- or weight rationalisations; and taking advantage of less stringent legislations in individual markets to develop a cheaper variant. Several interviewees indicate that when a clear picture of project costs starts to emerge, management often starts to look for ways to reduce costs; *'we very much base optimisations on purchasing cost, unit cost, instead of looking at the cost of the entire process'*. Similar reasoning is common for weight rationalisation, as expressed by a manufacturing engineer:

> *'Every so often we detect; "our cars weigh way too much, we need to reduce the weight", and then we start looking; "are there any screws here that we can shorten?" (. . .) and then suddenly we have different screws on model 1 and model 2'.*

Time pressure, cost targets and uncertainty easily lead focus towards the aspects that design engineers are measured by, missing the bigger picture. Performance is for instance measured model by model, not on the complete product portfolio. The inability to strategically design long-term is mentioned as a driver of re-design and excessive variety:

> *'(. . .) we don't have time to think strategically and long-term today, instead we receive a sub-optimal solution for that first application which the engineer is measured by (. . .) they must understand that the thing you develop in this project shall exist 15 years and fit in 30 different vehicles (. . .) long-term strategic development plans per sub-system foster re-use (. . .) and when you visualise it, you suddenly understand; "all right this system should be in these vehicles and exist during this time and be able to manage these technology shifts on surrounding sub-systems" (. . .) to be able to measure the project's performance in this aspect, we don't have those tools today'.*

Distributing knowledge horizontally is important. This is often lacking; *'(. . .) often you don't bring knowledge along between vehicle projects (. . .) it's a too small of a critical mass of core competencies involved'*. Retaining knowledge from previous and ongoing projects is paramount to not *'re-invent the wheel'*. Knowledge about requirements, analyses and design intent is also difficult to obtain. Several interviewees testify about re-used designs that no one dares to modify, as it might contribute to unknown functionality. Each design area's own ways of structuring knowledge through documents or meetings influence their abilities to find a component suitable for re-use. A weakness partly highlighted by a PLM specialist:

> *'A significant barrier is our way of dividing our products in different structures where we basically need to keep a manual sync between the common material'.*

Late ECs often increase internal variety, as neither time nor safety margins exist to create an optimal solution. Flexibility decreases and risks increase as start of production approaches; *'you have to utilise absolutely certain solutions, as you are running out of time. And then you easily create new variants of something'.*

Deficient Coordination
All interviewees posed with the question express that other departments, especially sales, design and purchasing lack an understanding about complexity in R&D[1]. This correlates to information bias in decision-making, highlighted by a complexity manager:

> *'It has to do with the internal power balance. I would like to see that R&D had the mandate to, and were better at, saying; "no, we have a well-functioning gearstick, it works (. . .) we will use that". Design, they love to re-do everything, they don't care if it needs to be re-done underneath'.*

Proliferation of Requirements
Internal requirements emerge from manufacturing, logistics and purchasing[2] as well as from R&D. Coordinating the definition of requirements across product families and models is mentioned as a key enabler of commonality:

> *'We should be very clear with what's important to our core values. But with what's not important, it is enough being as good as, or as bad as our competitors (. . .) but sometimes we, due to some specific requirement, send something back to the supplier and say; "this is not good enough" '.*

Corporate Culture
A culture that allows changing standard interfaces, too casually adjusted to the problem of variety, too easily allows internal variety; *'within R&D we love to re-design, it's what we're good at, what we like to do'.* At the company, there seems to be large cultural differences between different design areas where some are working much more actively with complexity reduction.

4.1.2 Complexity and Variety Management

The strategies pursued to reduce variety and manage complexity are outlined in this section.

4.1.2.1 Product Platforms, Modularization and Commonality

> *'You develop a platform in order to reduce it [variety]. But as time goes by, someone wants this and that (. . .) What could have been small becomes ruined eventually'*
>
> - Group Leader, R&D

A strategic department is responsible for paving the way for development of modules with a performance bandwidth and standardised interfaces. This aims at further establishing modularization strategies into the corporate culture. An interviewee states that although some design areas already are *'thinking modular'* they still suffer from a culture that *'allows changing standard interfaces'.* There is a discrepancy between sub-systems. *Top-hat* components are for instance developed more ad hoc. The long-term development plans introduced for many sub-systems aim at early identification and mitigation of design dependencies and risks.

Commonality is a well implemented strategy. All design areas participating in this study admit to actively trying to reduce internal variety through commonality, in particular for high-cost,

[1] In addition supply chain management and manufacturing engineering express that R&D often lack an understanding about the complexity emerging in the production system as a result of variety.

[2] Using different suppliers for the same component increase variety. Common reasons for using several suppliers include risk reduction, cost reduction, different manufacturing sites or isolated project decisions.

long-lead-time components:

'(. . .) we try to find carry over solutions when entering the next project. Can we re-use this rear-view mirror? Is there something we can influence on the door for instance, where the mirror is attached? (. . .) But normally we fail because it's a new design and we have to do modifications'.

The future likelihood of carry over is however seldom considered during development of a new component. The cultural difference is also highlighted by the fact that this commitment varies across the organisation:

'If I take bumpers as an example, we asked (. . .) "is there any basic concept we should try to implement for all our bumpers so that we can have one process or simply our way of thinking?" But no (. . .) for every product family we develop a new concept because it's not a part of the platform. We don't standardise when it's about visible components'.

A key challenge is transforming the organisation to adopt a culture that prevents pd to digress from strategic boundaries:

'(. . .) to gain an understanding within management as well as with the engineers how important it is to abide to standard interfaces and frameworks in order to reduce complexity (. . .) because it permeates everything and everyone can affect it'.

4.1.2.2 Variety Management

'If you ask the ones selling our cars they say; "we don't want you to package the features too much" (. . .) sure, but if we leave everything open, we can't afford the development'

- Complexity Manager, R&D

The variety developed in a project originates from intended functions, characteristics and target markets. The commercial variety which is theoretically possible is limited in different ways. *Option bundles*[3] are increasingly used in order to limit the commercial freedom of choice (typically referred to as *minimising complexity*). They also simplify demand forecasting and keeping less volatile volumes of specific parts over time.

A cross-functional team is actively working with variety management (internally referred to as *complexity management*), targeting the issue with excessive commercial variety. This includes identifying the optimal balance between commercial and internal variety. Another key task is implementing complexity management as a mindset in the daily work of design engineers. The team partly works as a control organ. Design areas try to reduce their variants in new development projects, much based on experience and in some cases on historic demand, extrapolated from similar models:

'(. . .) together with the sales organisation we make sure that we don't have too many variants and actively review, are there (. . .) any offers we might change? (. . .) we know that we typically don't need those, there we don't need that, we'll remove those'.

Take-rates[4] per component variant, based on historic demand, are used to eliminate low volume components. Although this information could be used to remove historically under-performing variants in development, it may be outdated quickly. It is also often neglected as design engineers

[3]Option bundles are feature variants bundled together in packages of different sizes, meaning that a large portion of functionality is a result of indirect customer choices. E.g. instead of a camera being optional regardless of the other features embedded in an RVM, option bundles dictate that the camera is only valid in combination with automatic dimming and blind spot indication — reducing the number of possible system variants.

[4]Take-rates is a metric that indicates the percentage of all vehicle variants sold a component or system variant is used in.

move on to the next project. Removing low-volume variants already in production however has limited economic benefits for pd.

It can be concluded that product variety is triggered by external as well as internal factors. The effects of variety-induced complexity span strategic, organisational and cultural capabilities as well as processes and tools. Many challenges are in fact recognised and targeted at different levels of the organisation. However, the commitment seems to vary significantly. Design areas with a self-image of being complex, having a tradition of managing variety, are significantly more aware of the challenges and committed to avoiding them.

It is clear that variety has implications for a manufacturing firm on several levels. The most evident are the measurable overhead costs in development, purchasing, production and after-market. Complexity management efforts are often biased towards areas where complexity costs are evident. For instance in the production system where variety triggers unique stations, tooling and fixture investments, changeovers, fool proofing the assembly line[5], and factory space dedicated to pre-processing[6]. Variety-induced complexity in manufacturing is well-documented, e.g. (Asadia et al., 2016; Hu et al., 2008; Wang et al., 2010; Wang et al., 2011; Zhu et al., 2008). Negative impacts on automotive manufacturing, specifically manual assembly processes and parts supply (e.g. degrading quality and productivity) are well studied phenomena (Fisher & Ittner, 1999; Fisher et al., 1995; MacDuffie et al., 1996). It is however assumed that a significant part of the costs of complexity arise in pd. These issue have been less debated and are the focus for the remainder of this chapter.

4.2 Variety-induced Complexity in Product Development

'Time is spent on managing complexity and studying which four different solutions are required for this, let's say wire, depending on the overall vehicle configuration. Instead of spending it on designing a smart solution that fits in all vehicles, with the right material, with low weight'.

- Product Architecture Specialist, R&D

This section further unveils variety-induced complexity in pd. The organisational symptoms detailed in this section are listed in Table 4.1. Although known, many are notoriously difficult to quantify.

A design area is ultimately responsible for releasing a set of sub-system variants. Each variant shall fulfil each own's set of requirements, functional and non-functional, while functioning in all indented configurations. Of the design engineers posed with the question, 80 per cent express that their sub-system is complex and 67 per cent state that they have significant problems with dealing with this complexity.

[5]Different costly *Poka Yoka* and *Pick-to-Light* solutions are often needed to ensure that the correct component variant is assembled when human operators are involved.

[6]Pre-processing includes sequencing (the flow is moved from the assembly line to a sequencing area), component kitting (kits with the components used to save space and time on the assembly line) and pre-assembly. Each unique part number is stencilled to claim in average, 7 m^2 heated factory space for such operations and sequencing itself claim 30,000 - 35,000 m^2. Factory space is fixed short-term, why this constrains possible variety. In the long-term it is a significant cost driver.

Table 4.1: Organisational symptoms and identified root causes.

Symptom	Root Causes	Comment
Poor quality pre-conditions	Complicated and less repetitive assembly operations	Increase the risk of mounting the wrong variant or downloading the wrong software
	Complicated supply chains	The more a parts is handled, the higher the risk of quality issues
	Complex product configuration	Complicated to validate that a vehicle is configured with the correct parts
	Incomplete testing and verification	Increase the risk of offering configurations with faults or poor quality
	Information overload	Increases the risk of making the wrong decisions without access to the right knowledge
Development waste	Large number of wasteful activities	Time and resources are spent on wasteful activities such as administration and gathering understanding design pre-conditions
	Testing and verification	Time and efforts spent verifying that each configuration satisfy requirements
	Excessive variety	Resources spent on developing and preparing production for variants not generating revenue
	Knowledge/information shortfall	Time spent on collecting and managing information
	Complex troubleshooting	Time spent on identifying errors in physical products and in product structures
Insufficient design reuse	Sub-optimal solutions	Rationalisations, late changes, isolated quality fixes, short-sighted decisions, insufficient planning and deficient coordination trigger excessive variety
	Late variety growth	Late changes, time pressure and rigid, complex design conditions prevents generic solutions
	Design differentiation	Differentiated design that entail unjustified variety
Development inertia	Architectural rigidness	Difficult and time consuming to understand design pre-conditions, find a solution and predict the effects of a change
Development bottlenecks	Availability of resources	Creating value is limited by the availability of time, money and competence
	Knowledge/information shortfall	Difficult to procure, maintain and oversee necessary information
	Project delays	Unforeseen changes and quality issues delay projects
	Late engineering changes	Require a lot of resources and risk introducing new problems
Development uncertainty	Knowledge/information shortfall	Inability to procure, maintain and retain knowledge make design pre-conditions uncertain
	Low information completeness	Complexity and information overload increase the risk of releasing incomplete documentation, etc.

4.2.1 Defining Complexity and Making it Transparent

When we talk about complexity, we often think about how it is in production (. . .) but the difficult complexity, the one that costs us a lot of money, it is created in early phases, where we don't have the tools to work with complexity'

- Product Architecture Specialist, R&D

There is no pervading definition of variety-induced complexity within the company. Nor is it distinguished from structural complexity. Everyone has their own interpretation influenced by experience; *'you need to have experienced it during a number of projects before you fully appreciate how complicated it is'*. Complexity costs are almost exclusively related to the number of variants (part numbers) of a component. The baseline is almost exclusively historic numbers. The growth of RVM variants from 30 to 36 for one vehicle family to the next was for instance considered a large step backwards although the implications could not be quantified further.

Development costs (per project) are closely monitored, although no measures relate costs to variety. Stencilled costs are instead used to assess the costs per added part number, above all based on tangible costs in production processes. The result in an incentive to only reduce the number of unique part numbers when targeting complexity costs. There are tools quantifying part commonality. They do not however fully depict complexity as it is not solely about maximising the number of common parts. It also only provides a snapshot of the situation. Using such figures to contrast commonality over time proves difficult. Making the right decisions turns into a difficult task; *'if we add one of these dark threads in our seats, we do not know what that means, how much more work we need to spend to make that happen'*.

A majority of the interviewees relates the ability to make complexity costs transparent to better cooperation and decision-making. Ultimately choosing the *right* components to differentiate. In particular as the people with arguments backed up by figures win:

> *'(. . .) the marketing side has no understanding about industrial complexity (. . .) there can be a sales request saying, "if we could only get R&D to develop this variant of something, we will be able to sell 2 000 additional cars and the profit from 2 000 additional cars is this much", and very quickly that request is approved'.*

4.2.2 Variety as Part of the Development Task

'If you have problems in one area, how do you make sure that they don't affect others? Can you keep the solution locally to only that configuration or will it affect all other configurations? I guess that's the biggest challenge, keeping track of everything'.

- Design Engineer, R&D

Documentation, information and knowledge gathering, testing and verifications, and re-design can, in contrast to actual development work, be seen as *non-value adding activities*. Without appropriate support, resources spent on these activities seem to grow exponentially with increasing variety. The design engineers deny that the main issues relate to *developing* an additional variant[7]. The issue is rather threefold: *(1)* Understanding the pre-conditions when subject to variety; *(2)* Performing non-value adding activities; And *(3)* assuring product performance and conformance to requirements across the configuration space.

[7]The generalisation of this statement vary. For systems such as RVMs it is much about combining different components. Excluding one RVM variant might not result in a reduced number of unique components. A tremendous amount of resources is on the other hand required to develop left-hand drive and right-hand drive body structures.

Most design engineers identify management of design pre-conditions and sub-system variants over time as the most pressing issues. The amount of information that defines the context of complex systems becomes an issue to procure, manage, oversee and transfer. Increasing internal variety exacerbates this, as the technical field of responsibility typically shrinks. This means that each team gets responsibility over smaller and smaller sub-systems. The interviews indicate that design pre-conditions are becoming increasingly complicated — significant collaboration and extensive information flows are required in order to fully understand them.

4.2.2.1 Managing Knowledge and Information

'To be able to design correctly, you need access to a tremendous amount of information (. . .) many of these decisions [about design pre-conditions] are not made when the design engineer starts to develop the solution and that makes it difficult. Consequently, the design engineer needs to actuate a number of conditions himself'.

- Product Architecture Specialist, R&D

The amount of information increases with internal variety and with proliferation of requirements. Many decisions are above all based on experience[8]. Navigating the trade-off between performance, cost and variety is significantly more complicated than the trade-off between only cost and performance. The more complex the system, the more difficult it is to evaluate the solution space. It may for example be particularly difficult to find a good solution when the performance of different configurations diverges along a certain requirement, e.g. a load case.

Following the identification of a cost-efficient solution, keeping variety to a minimum is prioritised. The three-dimensional trade-off makes this process difficult and highly iterative. One-third of the interviewees express that sufficient knowledge about one's system and its interfaces is fundamental for managing variety-induced complexity. Half of the interviewees instead highlight holistic knowledge. That is, how everything is interrelated and the local as well as global effects of decisions and changes:

> *'I think that the most important thing is knowledge about your own system as well as how it interacts with surrounding sub-systems and the complete vehicle. You might have a good understanding about the things defined as requirements for your system today (. . .), but all the things that are unwritten, all functionalities the system contributes with to the bigger picture, which we haven't written down completely and which we haven't steered, it simply turned out that way, that's like tacit knowledge to some extent. That's the knowledge we need to find and document somehow so it becomes manageable'.*

A lack of knowledge about the factors that make sub-systems robust against changes is also highlighted. This knowledge is important to keep variety to a minimum over time. The PLM ecosystem places a high responsibility on the individual design engineers to manage necessary information. Except master documents, information and knowledge is shared both vertically and horizontally through meetings and word-to-mouth. Knowledge is somewhat retained through design journals, *lessons learned* and documentation of old decisions and quotations: *'I think that those in the top have difficulties seeing exactly how each decision strikes down here (. . .) it's only a few people who really understand the bigger picture'.* '

[8]The design area of RVMs has a strategy where they hedge themselves against possible late and costly feature changes by designing with every possible known functional requirement (the feature variants historically requested) in mind, even if it is not included in the project scope. For instance, by designing a variant with camera, they hedge themselves against incorporating a camera later on in case a particular requirement on field-of-view is not met.

4.2.2.2 Visualising and Modelling Variety

'A model with an option that has a camera might also represent one lacking camera, and that makes things complicated because we are working more and more virtually, and are supposed to verify more virtually'.

- Group Leader, R&D

The company's PDM setup is composed by one tool that contains a virtual 3D representation of the vehicle and one tool that contains the physical structure. The two product models are synchronised manually. In addition, variant-rich sub-systems typically only have a fraction of their variants represented as 3D models due to highly administrative release processes. The intended configuration space is often documented in Excel during concept phases due to poor tool support. The product structure of a previous model is then used as a starting point to define the new model as knowledge about its architecture starts to mature. The product structures starts to mature during the industrialisation phase. That is, the *product specification* and the generic BOM is defined. Before this point, information is scarce and distributed in different tools, without access to configuration support.

The existing tools (in particular the legacy system with low user-friendliness) do not visualise variety in a form required by different roles within and outside R&D[9]. Nor are they structured to leverage modularization and commonality. The support is particularly poor in the pd phases preceding industrialisation. Many areas thus use informal, Excel-based documents, structured in order to visualise variety according to their individual needs. Figure 4.4 provides an example of such a variant matrix[10].

The variant matrix is typically used by design engineers and the sales organisation to visualise variety and to relate components to feature variants and markets. It also aids variety management and communication[11]. A matrix usually contains, per variant, its part number, effectivity, take-rate and validity in terms of features variant (or option bundles). The interviews testify that the variant matrices have replaced much of the functionality in the PDM system even if they too are becoming increasingly difficult to maintain.

Formal configuration data mastered in the PDM tool is difficult to maintain and understand. Few fully understand how vehicles are configured as very few master the art of reading the product specification; *'it's basically only those who create the product specification who can read it to a hundred per cent'*. This is a key reason why data moves out to the informal world. In particular as the relationships between feature variants, feature bundles, markets and the actual customer choices are notoriously complicated. There is an intricate net of dependencies between feature variants making it complicated to understand and communicate how specific product variants are be configured, what is standard features, what is optional, and so on.

[9] A design engineer admits that they could use the product specification or other formal documents to understand and communicate variety. The user friendliness is however so poor, and the information is so complicated that other means of documentation are used instead, for instance when communication with suppliers.

[10] A limitation in the legacy PDM system restricts the usage of a part to be defined by a combination of more than five feature variants. To accommodate for this limitation, *extra variants* are created, containing a specific set of feature variants. For RVMs these packages are referred to as *MIR packages*, i.e. each of the packages *MIR 1 ... MIR n* specifies a particular combination of feature variants. It is not always unambiguous which feature variants a certain configuration contain as the content of the extra variant may be inconsistent across vehicle models — a *MIR 5* might have different functionality from one model to the other.

[11] Design engineers are responsible for developing a sub-system satisfying the functional requirements and characteristics required by the sales organisation and the vehicle project. Product specification specialists are responsible for authoring configuration rules, specifying which specific feature variants that make up each product variant. Change order specialists are responsible for release of parts and defining for which combinations of feature variants a part is valid.

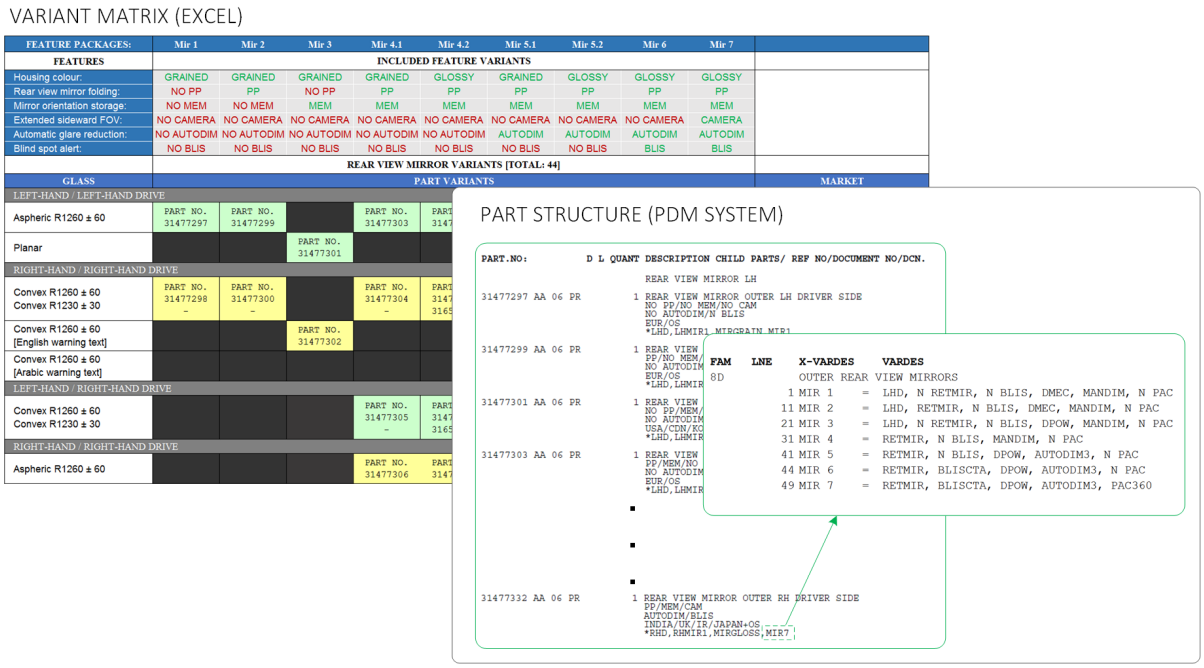


Figure 4.4: A variant matrix for external RVMs for a product family and a snapshot of related information in the PDM system.

Information is often transferred from informal tools to a series of formal tools. As different business processes have their needs, informations is copied, adapted and built on in different tools (e.g. tools for sales configuration or production planning). Each tool developed for a specific purpose. At each point of transfer, a slightly different language is used. Making it virtually impossible to understand the configuration data during the complete product lifecycle. There is a lack of consistency between organisational domains and life cycle phases as information may be distorted, unreliable and mean different things to different people. Figure 4.4 provides an example; it is not evident that the function *Power Park (PP)* on sub-system level (variant matrix) has its counterpart in the function *RETMIR* on vehicle level (PDM system).

It is also difficult to accurately ascribe take-rates on vehicle levels to individual sub-systems; *'we try to break it down to part number level on the rear-view mirrors, but we don't always succeed'*. The set of configuration rules is in some cases so complicated that it becomes very difficult to know the configurations consuming a given part, how well this corresponds to the initial purpose of the component and if the component actually contributes to any value:

> *'If you do not choose precisely this exterior and that interior together with the rest of the combinations, then you never get this part. That probably happens a couple of times a year'.*

An increasingly difficult task is the maintenance of design dependencies from a configuration perspective. That is, the constraints making specific combinations of feature variants or components invalid. With distributed functionality and option bundles, the configuration rules describing how different component variants are allowed to be combined might be extremely complicated to identify and communicate. Recurring issues related to acquiring the knowledge required to create the product specification were reported; *'sometimes it's enough with a phone call (. . .), but in some cases it can actually take months (. . .) we don't have any good tools really (. . .) it's Excel and it's Power Point'.*

The interviews testify that a significant amount of the time spent on administrative tasks is dedicated to troubleshooting, verification and validation. The most common issue is a product configuration lacking a component or lacking the correct component.

The ambiguousness described in this section stands in direct contrast to a holistic view of product configuration data. Communication and knowledge transfer across domain boundaries is limited. Moreover, transfer of information to the informal world implies risks of lost or altered data. Information also becomes isolated in silos. One-sixth of the interviewees request better support systems in order to visualise and understand the configuration spaces.

4.2.2.2.1 Design Dependencies and Engineering Changes

'How do you ensure that your problems do not affect other areas? Can you try to keep the solution locally or will it affect all variants? I guess that's the biggest challenge'.

- Group Leader, R&D

Figure 4.5 illustrates how frequent ECs are by depicting the number of changes per year made to the most common types of configuration rules. Knowledge about the context of one's sub-system, especially how the physical and functional layout differs between product families, models and variants, is required to manage changes. Such knowledge often originates from experience, meetings, documents and from creating digital mock-ups and performing packaging analyses.

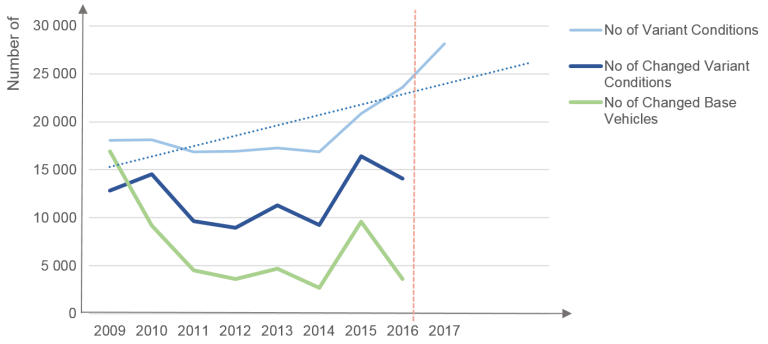


Figure 4.5: Number of feature variants and the number of changes made to the functional content of vehicles (company statistics).

The variant matrix is used for this purpose as well; *'for instance if you change a clip, then you might need to know in which cars the clip is used (. . .) and when it was introduced'.* With such information one may assess which variants are subject to a change. Many interviews however indicate that the information is extremely difficult to acquire:

'I would say that there is a very good understanding about how different components interact within a sub-system. In between sub-systems however, (. . .) there are very few with the competence and knowledge about how different sub-systems interact (. . .) it's not just mechanically, that's the easy part, functionality is a lot harder'.

The task of designing sub-systems that are robust against feature changes and the evolution of interfacing sub-systems is a part of the complex nature of vehicle development:

'It cannot simply fall through the cracks (. . .) if we change a component, the cabling may for example all of a sudden be too short as a consequence (. . .) as soon as you need to add a larger component you almost automatically push something else (. . .) everything from zero effects to complete re-design of the engine compartment'.

Functional dependencies are difficult to identify and maintain in particular for mechatronic systems; *'I believe the active safety system is at the height of distributed functionality (. . .) it affects 15-20 different control units'.* Physical dependencies are easier to identify with existing

tools. As experience is more scarce in the mechatronic domain, conditions are often unknown:

> *'when the car becomes completely silent, without motor noise masking other things, you suddenly hear noises you've never heard before (. . .) fan motors or antifreeze bubbling'.*

> *'I guess we didn't have in mind how the door node works. Our component is not tested with (. . .) the real software, it's tested in a rig where you activate and deactivate the power fold function. The behaviour might be different in reality. In reality there is an electronic interface we didn't think about designing the frame. We could have the function active much longer than we thought, and it ultimately broke'.*

Design dependencies influence wanted as well as unwanted behaviour (e.g. noise)[12]. The key, according to a product architecture specialist is a knowledge-based way of working to intercept such dependencies early in development. Two-thirds of the inquired interviewees expressed that they are experiencing issues with changes causing unforeseen effects on neighbouring subsystems. This places high demands on communication. If a change is allowed to propagate in several steps, the magnitude of the initial change might be underestimated. An engineer responsible for fuel lines highlights an example; *'we often need to implement a change in the end (. . .) that's why we're always late with everything'.*

Dependencies that multiple across variants must be assessed as well; *'it can be a bit stressful to like, have I really checked against all available variants now'?* The information required to evaluate an intended change can be significant as affected components may already be in production or at the hands of customers. With the speed and relative simplicity to scale functionality in software, such configurational dependencies increase. Compatibility between versions of software and hardware are often neglected and not appropriately documented.

More than half of the interviewees posed with the question testify about recurring issues with unforeseen, negative effects of changes propagating between variants. These trigger costly re-design and the risks increase the later the changes occur. All interviewed design engineers express that they often suffer from late ECs. This is to a large extent caused by the high development tempo and an inability to do things right from the beginning. Variety also trigger ECs as there seems to be a paragon of developing short-term designs, optimised for the context closest in time. This is highlighted by a real case; the door Electrical Control Unit (ECU) and window lift integration in the largest model of a vehicle family:

> The software binary deployed in the door ECU operates, among other things, the window lift. The window's elevating moment is provided by an electric motor located in the door's dry zone. The physical space is limited and the topology is highly optimised. Although most components (including the rear door ECU) were common in two vehicle models, the topology in the rear doors were modified in order to fit in the smaller model. When deciding the new topology, the interface between the ECU and the electric motor was neglected. In the new layout the motor was flipped 180 degrees whilst the ECU remained unchanged. The result being, as the cable connector remained in the same position, a counter-clockwise rotation lowering the window when the upwards button was pressed and vice versa. This error was detected in complete vehicle prototyping close to production. This ultimately resulted in two ECU variants as it was too late to update the topology.

As the above example shows, short sighted decision making might increase complexity even with

[12] A sub-system might use another sub-system's component as structural support, e.g. a console or a set of holes. For commonality reasons, the hole or console might be included in all vehicle configurations. Real cases from the vehicle body structure highlight the need to seal holes in the variants where they are not used in order to prevent leakage of air, liquids or noise generation — sometimes a neglected dependency.

the best intentions. Less mistakes occur when sub-systems are developed in parallel, with cross-functional cooperation. With platform-based development, component re-use and continuous evolution however, sub-systems will differ in maturity and cooperation requires more communication; *'as we use as many carry-over components as possible, it affects the projects already in production (. . .) the work load is at least constant, if not higher when we have carry-over as we have to consider old cars'*. High commonality result in more rigid design pre-conditions. Finding a 'good' solution becomes more complicated. Tightly coupled dependencies in this context increase the rigidness in platforms and vehicles, making them incapable of efficiently incorporating new requirements or technology artefacts. A design engineer working with fuel lines highlights the importance of decoupling such dependencies:

> *'As our fuel lines are so long they are very sensitive. If a change is implemented, we often have to change as well (. . .) if it wasn't so sensitive you might have less variants'.*

Variety-induced complexity can be managed by designing more robust systems. To do this, such vulnerabilities need to be identified early. This is difficult due to tight safety margins; *'increasing the height of the car with, say 20 mm — we are so close to not fulfilling legal requirements today — can result in a need to change the size of the mirror rather late in a project'*. This is highlighted by a real case:

> The effective Field-of-View (FoV) provided by an exterior RVM is (partly) a function of the distance between the mirror and the driver. Until the industrialisation phase of one particular vehicle model, *one* RVM design was re-used across all models in a vehicle family. Approximately halfway into industrialisation of the new model, with tools already ordered, a 10 mm increase of the height of the chassis was decided as it would enhance the aesthetics of the vehicle. The change affected the seating position, which affected the distance to the mirror which no longer provided an effective FoV satisfying legal requirements. The decision was made without apprehending this effect. The ultimate result was a new, unique mirror glass variant and a new variant of the housing (with new dimensions) per exterior colour.

A similar case highlights neglected dependencies when introducing a new feature for a set of specific variants. The software application used to control the window lift is calibrated to avoid pinching obstacles (e.g. a finger) between the window and the door structure. A sun curtain operating in parallel with the window was however introduced for a specific set of trim levels. The curtain required a hook in the door structure who's effect on the pinch calibration was neglected. The result; failure to meet the requirements on pinch force for the vehicle variants employing the sun curtain. The issue was detected very late and resulted in a re-release of the complete set of software deployed on the door ECU.

4.2.2.3 Implications for Quality and Product Performance

'I don't think it significantly affects the overall product quality (. . .) but for those expensive things, where we want particularly good quality, we risk ending up with poor quality'.

- Manufacturing Engineer

Variety-induced complexity has no direct correlation to poor product quality. It may however impair the pre-conditions achieving high quality. One factor is the inability to test and verify each possible configuration; *'we verify three or four vehicles thoroughly, or semi-thoroughly, we at least try'*. Physical prototypes unveil many issues. They are however extremely costly. The cost of virtual verifications also increases with variety as the required time increases exponen-

tially[13]. Shortcuts are thus pervading; *'you choose the worst configurations (. . .) so it is important to choose smart. If it works on them, it probably works on the rest as well'.* The interviews indicate that critical issues are rather rare due to good knowledge about which configurations to verify. It is however a high-risk approach. Quality issues may pass unnoticed, in particular for low-volume variants.

As less time is spent adding value, less innovative products which are lagging in both performance, quality and cost efficiency might be developed. This risk increases with time pressure and insufficient knowledge. A rigid environment also becomes an obstacle, propelled by a large complexity debt, i.e. a history of sub-optimal designs.

4.2.2.4 Variety-induced Complexity in the Mechatronic Paradigm

'People think software can be very flexible but often it is not'.

- Software Developer

Mechatronics introduces a whole new dimension of design dependencies. Software binaries are designed to handle variety by employing a *superset strategy.* A software package deployed on an ECU consist of three parts: *(1)* A load module which contains the executable software application; *(2)* A configuration file which contains information about the parameters used for inter-software communication; And *(3)* a parameter file with parameters used to configure the load module for any given vehicle individual, e.g. language settings. The parameter file is the only file unique for each configuration. The parameters are flexible, risk-free and quick to change. A change to the load file might however propagate to other systems or variants as a new version might have to be released towards all variants in product range.

Software and electronic systems are typically verified in delimited contexts, with varying completeness in their interactions with other systems. When dependencies are not contained it becomes and issue:

> *'I need to test my control units in a complete environment why all other control units work. But they might work with another supplier, with other time plans not planning to freeze their conditions until later, why their control units lack the correct software (. . .) then I can't verify my component until way, way too late. You then test the entire system and discover things you never had a chance discovering earlier'.*

Propagating changes are in particular common for electronic systems. Experience and tool support (packaging analysis and digital models) mitigate the risks in the mechanical domain. In the mechatronic domain however both experience and tool support are scarce in comparison.

[13]Time spent on verification correlates to the number of variants and requirements. E.g. a body is verified against a load case consisting of three collision types. 5 configurations thus require 15 simulations

5

Framework for Enhanced Complexity Management

*This chapter presents a framework for enhanced complexity management. First, the key challenges identified in the study are summarised, placing the framework in its context. Finally, the framework is outlined, clarified as to answer **RQ 3**. By intertwining product architecture modelling, a system-oriented product structure and the CPM, it focuses on knowledge creation, assessing variety-induced complexity and on quantifying risks of change propagation in architecture variants.*

5.1 Summary of Challenges in Product Development

In many industries, companies are required to develop increasingly complex products with shorter lead times and at greater efficiency (Wynn, Caldwell, et al., 2010). The situation described in Chapter 4 is not unique and many of the challenges have been previously described. This section summarises the key findings in order to condense the context framing the proposed framework.

The first and foremost conclusion is that variety-induced complexity is an organisation-wide concern. Thus, a myriad of capabilities in the firm's PLM ecosystem can propel its ability to limit variety and manage the resulting complexity. To summarise the key findings in a systematic manner, a number of capabilities have been formulated. These are derived partly from the empirical result and partly from published literature.

A proper positioning of the variety offering can result in less complexity. It may also result in lost market shares if it makes the offering less attractive. Thus, the following capability is formulated:

> ***Capability 1: Optimally positioned variety offering.*** *The organisation's ability to continuously supply a product offering with an optimal balance between variety demanded by the customers aligned with the company's strategic profile and the costs of variety-induced complexity.*

This is a topical challenge; Jiao et al. (2007) named it the *product family positioning problem*. It is a key challenge at the studied company. Several interviewees express the inability to predict the optimal level of variety and its relative volumes as the most pressing issue. As a result, vehicle variants are developed with none or little contribution to profitability. The product family positioning problem is a well-known challenge and will not be explored in any more detail in this study.

A particular challenge related to the product family positioning problem is the ability to predict customer preferences in a dynamic market with an advocacy supporting the long lead times

in platform-based development. Time is in this context a very important factor. Uncertainty increases the further into the future one looks. A key strategic capability is assessing the costs of complexity in time — planning for complexity. This is the very aim of product family oriented design. If these strategies are not based on the right decisions, late ECs are inevitable and add complexity instead of reducing it. Five out of nine interviewees who deal with internal variety on a daily basis express that internal variety increased over the platform's lifetime. Achieving low internal variety over time is a significant challenge in practice.

Given the dynamic nature of the automotive industry, one may dispute if it is at all possible to estimate demanded variety precisely enough. Many systems have a lifetime spanning ten years or more. While others continuously evolve around the pre-conditions that the former define. The integral nature of vehicles places extreme demands on defining robust interfaces between sub-systems with different cycle times in order to achieve flexibility. The limited physical space however forces engineers to optimise topologies for individual variants. Ulrich (1995) introduced *geometric nesting*; a design strategy for efficient use of space and material. It involves *'the interleaving and arrangement of components so that they occupy the minimum volume possible, or, in some cases such that they occupy a volume with a particular desired shape'*. The inevitable drawback is however more coupled interfaces.

Being flexible enough to rapidly adjust to new circumstances in the market is a key challenge. In particular, this challenge relates to maintaining a product architecture flexible enough to accommodate for uncertainty, avoiding costly re-design loops. This allows formulating two additional capabilities:

> ***Capability 2: Flexible variety offer.*** *The organisation's ability to monitor and feed demanded variety back to pd while flexibly adjusting its offered variety. Along with its ability to understand the complete solution space and in particular the key contributors to customer utility and to complexity costs.*
>
> ***Capability 3: Architectural flexibility and robustness.*** *The organisation's ability to design sub-systems with an architectural flexibility that allows introduction of changes and of new technology artefacts. This require modular solutions with robust interfaces capable of encapsulating functional and geometric dependencies.*

These capabilities are different, yet interrelated. If not met, the firm will ultimately supply the wrong commercial variety, at a price too high or with profit margins too low. The result is likely an increased complexity over time as the firm tries to salvage the situation with ECs, cost reductions and quality improvements. The costs of sub-optimal design decisions could be long-lasting as a complexity debt is built up. This debt makes it extremely difficult to efficiently meet new customer needs or to incorporate new technology, again leading to sub-optimal decisions. Thus the firm fails to achieve economies of scale and scope over time. Moreover, a holistic modularization and variety strategy is crucial in early stages of development, even if design pre-conditions are fuzzy. An uncontrolled growth of variety might otherwise occur during the industrialisation phase which was in fact the case at the studied company.

The ever-increasing number of software and mechatronic systems in vehicles adds another dimension to this challenge. The current focus on modular architectures is largely related to the mechanical regime. In the mechatronic regime, modularization is a challenge as functionality is pre-dominantly distributed.

There is a clash when processes, the organisation and the culture are biased towards single products rather than an architecture. This is manifested by organisational structures that fail to take a holistic responsibility for variety and commonality across the entire product range.

This includes taking a holistic responsibility for variety-induced complexity — positioning the commercial variety offered while balancing it against the costs of complexity. A fourth capability can thus be formulated:

> ***Capability 4: Holistic variety management*** *The ability to foster complexity reduction and design re-use across all levels of the organisations. Achieving a consensus of what variety-induced complexity is, its drivers and how to make the costs of complexity transparent, ultimately integrating variety-induced complexity in product portfolio and product architecture decision-making.*

There are several dimensions in which variety-induced complexity can be managed or limited to a certain degree. In this study, four such dimensions are identified, depicted as surfaces in a pyramid in Figure 5.1. The innovation dimension can be thought of as how smart designs and commercial decisions can be used to limit complexity. For instance, not developing combustion power trains or achieving personalisation with software. In this dimension, the product architecture interplay with the commercial offer. When the latter interplay with the organisation, the strategy dimension is created. This relates to how strategies are used to manage complexity, e.g. brand positioning, modularization and platform-based development.

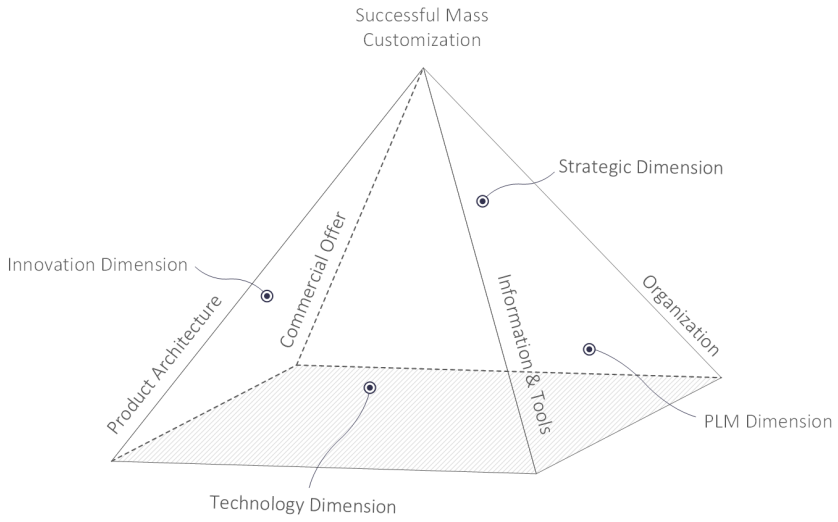


Figure 5.1: Four dimensions and influencing components for successful mass customisation.

Many of the challenges discussed thus far are somewhat generic and belong to either the strategic or the innovation dimension. In order to cope with the challenges however, richer information flows as well as more efficient tools are required to provide design engineers with a clear view of the road ahead (Zipkin, 2001). When in lack, design engineers often fail to understand the full picture when dealing with complex products (Gustafsson et al., 2016). Blecker and Abdelkafi (2006) argues that the lack of insufficient tools and adaptable models to capitalise on similarity and commonality is a trigger of variety-induced complexity. Early design re-use for example requires knowledge about the functionality of systems and sub-systems, their interaction patterns, their limitations and how they came into existence. The company's inability to generate, maintain and distribute the appropriate information is among the biggest issues identified in this study. A lack of the appropriate information flows and tools introduces waste (delays, re-design and costly verification loops) and ECs. It also inhibits innovation and trigger new internal variety (late changes under time pressure in a rigid technical environment force one to use simple and well-known solutions).

During the remainder of this thesis we will focus on the PLM and technology dimensions. The

former can be thought of as organising around information order to improve the efficiency of pd related activities. The latter can be thought of as leveraging tools in order to better manage the complexity introduced into the product architecture, by choice or accidentally. Both dimensions assume that a specific level of commercial variety is targeted and instead focus allowing variety, yet controlling it in the industrial system.

Adopting the definition of a platform as *a set of common assets from which a stream of product variants may be derived*, a platform can be seen as a mean to systematically generate, share and re-use knowledge. The empirical result shows that the development of a product architecture that capitalises on flexibility and commonality is more challenging than developing individual products. This is partly due to long lifecycles and integration of systems with varying maturity. A final capability can thus be formulated:

> ***Capability 5: Enhanced product modelling support.*** *Provide tool and method support to visualise and understand design-preconditions from a variety-centric point of view. Procure and transfer information to support effective design decisions in all pd phases.*

A number of shortcomings related to this capability were identified at the company:

- Inability to plan for variety and complexity on sub-system level and on platform level.
- Inability to integrate variety in early product architecture modelling and decision-making.
- Inability to model variety for the product platform and inability to model the complete solution space, on sub-system- and on vehicle level for the entire product range.
- Inability to formally integrate variety and variety-induced complexity in the design task.
- Insufficient transparency and information about designs, their intent and their context.
- Insufficient traceability between structures, e.g. between components and customer utility.
- Inability to unveil and quantify the costs of variety-induced complexity.

The framework introduced in this chapter aims at providing the firm with an *enhanced product modelling support*. It targets the key factor introduced in the *reference model* in Section 4.1.1. The aim and goal of this support is presented in Figure 5.2. Moreover the framework aims at proving the research hypothesis stated in Section 1.2.4.

The framework rests upon four postulates defined in this study:

1. Designing 'good' solutions when subjected to variety-induced complexity require more information about systems and their contexts; the 'goodness' of a design is not only a function of its cost, quality and customer utility but also its contribution to overall complexity.
2. Architectural flexibility and robustness are necessary to leverage commonality over time, as evolvability is a strategic advantages; when prediction fails, engineering changes will continue to be pre-dominant.
3. There exists a trade-off between performance and commonality and commonality implies a smaller and more rigid solution space; integration of new features and technology artefacts may be counteracted by commonality.
4. The ideal state of axiomatic design is virtually unattainable in automotive since functional dependencies will exist between sub-systems.

5.2 Managing Complexity using Change Prediction

This section presents a framework for modelling and assessing product architectures. It aims at supporting enhanced management of variety-induced complexity. It is based on state-of-the-art methodologies presented in Chapter 3.

Focus lies on *managing* variety-induced complexity through modularization and commonality,

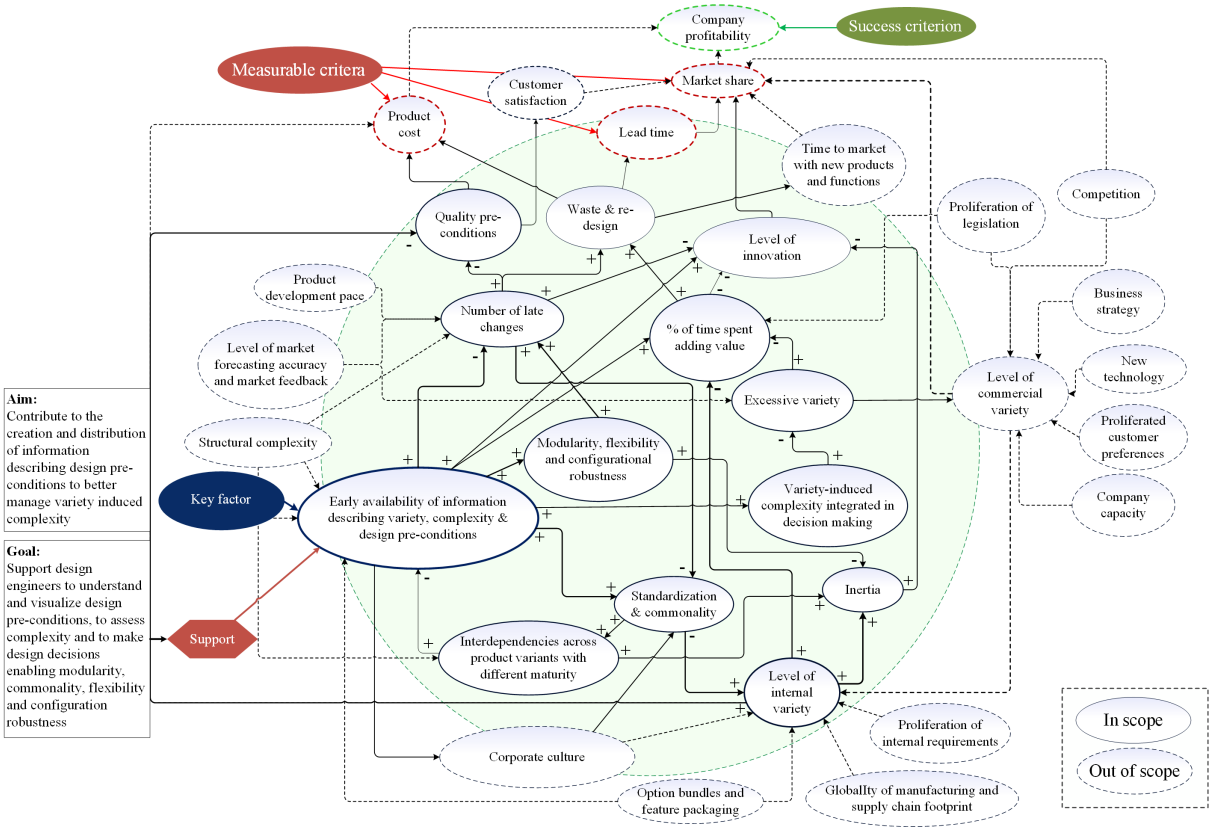


Figure 5.2: Impact model, with aim and goal of the support framework proposed in this chapter (applied from Blessing and Chakrabarti (2009)).

rather than on limiting variety. A correlation is identified between low architectural flexibility and robustness on the one hand, and increased internal variety on the other. Thus, the framework also focuses on change prediction. CPM shows good promise to generate information supporting the development of modular product architectures leveraging complexity assessment, design re-use and architectural flexibility.

The information required to make well informed decisions in early pd phases is often immature or pre-dominantly based on experience. In the mechanical domain, a lot of knowledge may be extracted from geometric data. A key concept of systems engineering is that specification, functional decomposition and solution abstraction precede physical embodiment. Three out of the four product models required to fully describe a technical system (according to Hubka and Eder (1988)), namely the *product specification*, *function structure* and *organ structure* (Andreasen, 1992), may thus be readily available in early stages of pd. The proposed framework uses the CC approach which integrates these models. Regardless the level of abstraction, it allows formalisation of knowledge about a system's architecture before the solutions are embodied. Raudberget et al. (2015) call this the *pre-embodiment phase*. The combined model of the product architecture may be assessed using CPM. This allows analysing e.g. the 'goodness' of the solution much earlier. Figure 5.3 provides a schematic overview of the framework. It rests upon the state of the art on EF-M modelling, CCs, design dependencies and change propagation. The following sections describe each step in more detail.

5.2.1 Specifying the System Bandwidth

A system's *functional bandwidth* is defined by the commercial variety it aims at satisfying. It is derived from the market segments and customer needs a system shall fulfil. The bandwidth

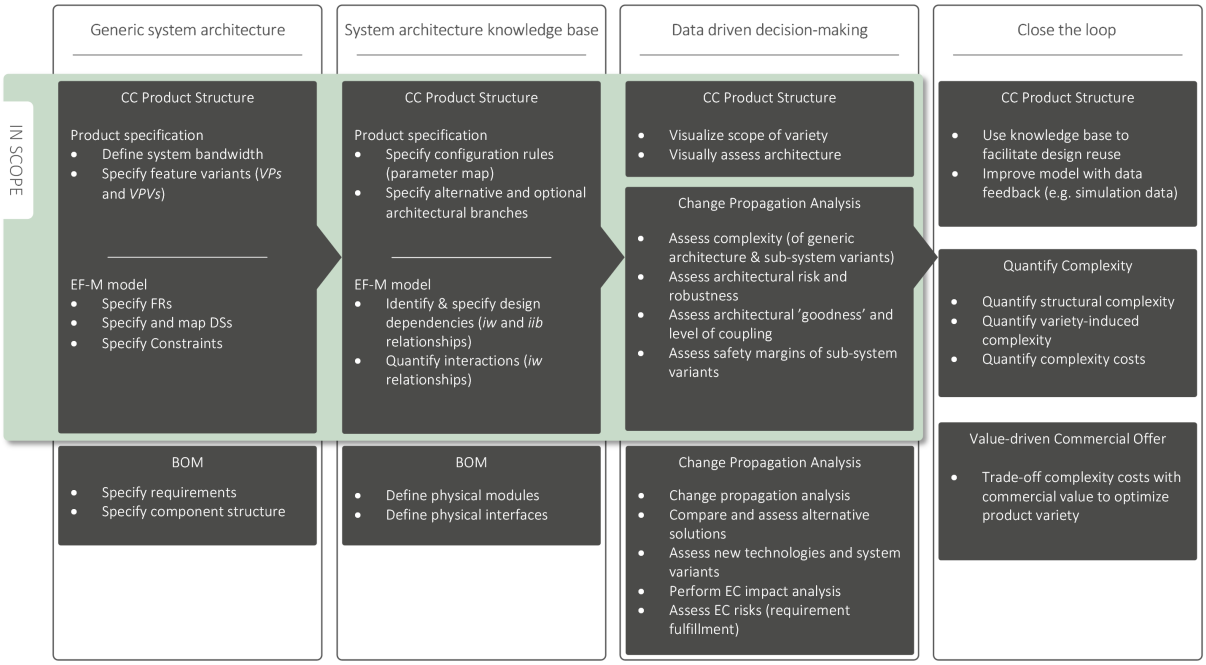


Figure 5.3: The proposed framework and the different steps composing it. It includes steps that lie outside the scope of this thesis.

may be expressed in terms of *feature variants*. These may be 'true' or 'false' values to a specific attribute or manifested as values distributed along a continuum ranging from e.g. a low target value to a high target value of a requirement.

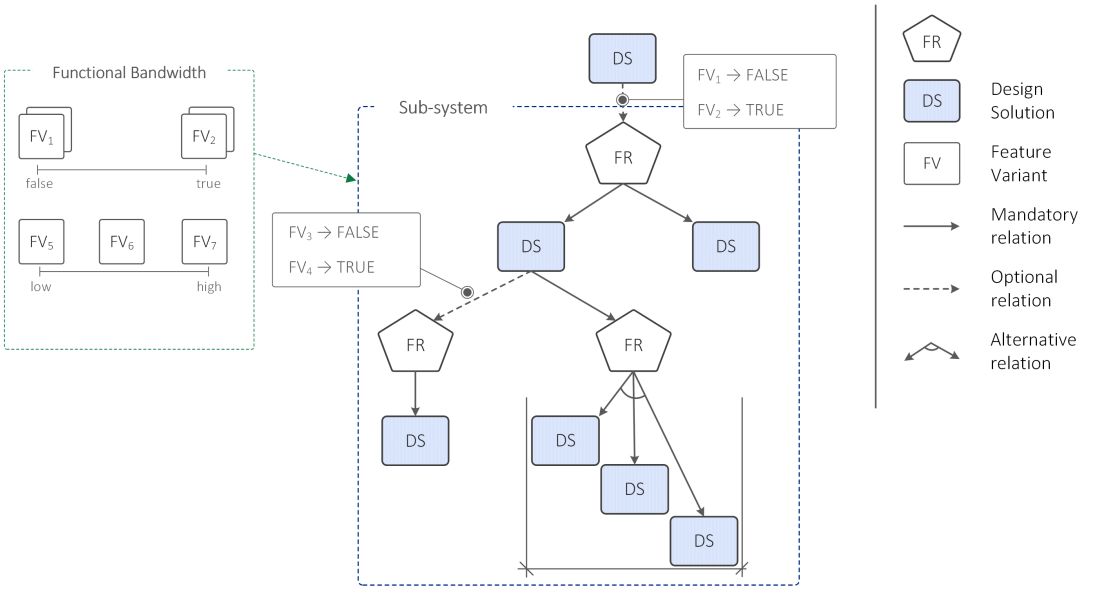


Figure 5.4: Feature variants in the functional bandwidth used to configure branches of a sub-system's EF-M tree.

The functional bandwidth for a complete system can be cascaded down to individual sub-systems. In an EF-M tree, this may be represented by alternative or optional FRs at the different hierarchal levels. The FRs of a system may thus have different *states* depending on the configuration. The states may be true or false or have more explicit target values, e.g. *high, mid, low*.

The *complete bandwidth* of a system is a solution-dependent extension of the functional bandwidth. Each FR may have a *design bandwidth*; it may have alternative DSs fulfilling it. That is, there might be different solutions to the same FR. Alternative DSs may arise due to varying requirements, varying performance targets, cost targets or generations of technology. Figure 5.4 conceptually depicts how the complete bandwidth of a sub-system can be modelled in an EF-M tree given a functional bandwidth.

5.2.2 Create Generic Functional Architecture

In the physical domain, data can be extracted from analysing geometrical models and simulations. In the functional and organ domains however, other methods are necessary. Modelling a product's *context* rather than *content* allows analysing the causal relationships between the functional domain and the organ domain as well as abstract design dependencies in the organ domain. The EF-M tree is a well-suited tool for modelling functional architectures. In addition, by a visual representation of the functional architecture, a deeper understanding about a system's behaviour and potential can be obtained (Raudberget et al., 2015).

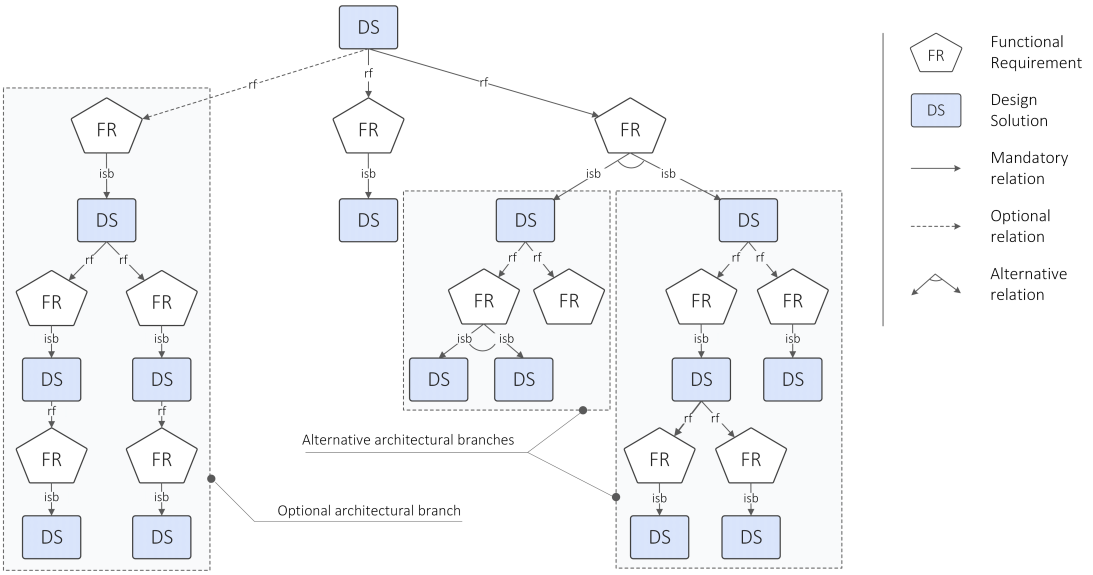


Figure 5.5: Generic (non-configured) EF-M tree. The dashed boxes represent branches of the tree that may, or may not be present in a specific configuration

A *generic* EF-M tree constitutes a representation of variety as its FRs and DSs cover the complete system bandwidth. At each hierarchal level, EF-M modelling comprises one-to-one mappings between FRs and DSs. The causal, *is solved by (isb)* relationship between FRs and DSs implies that the DS will exist in a configuration containing the FR. A FR may however be fulfilled by multiple, mutually exclusive DSs. The isb relationship is thus configurable. Moreover, a FR may or may not exist in a specific configuration. The casual *requires function (rf)* relationship between DSs and FRs is also configurable. Configuration assigns true or false values to each optional and alternative relationship. Figure 5.5 conceptually depicts a generic EF-M tree.

The starting point is defining a single, top-level FR. For concept generation, the EF-M tree is typically generated top-down and abstract solutions are chosen on each subsequent hierarchal level. It may however also be used for formalising knowledge about an existing architecture, i.e. a bottom-up generation of DSs and FRs.

An EF-M tree can be analysed regardless how many levels it contains. As the architecture evolves, new hierarchal levels can be added making the assessment more complete. Lewandowski et al. (2014) define three hierarchal EF-M levels based on the level of detail. The *static level* consists of the top-level functions that provide the overall structure. The *conceptual level* consists of the intermediate functions that provide additional level of detail. The *concrete level* is the last functional description before embodiment of physical components (Raudberget et al., 2015). In the physical domain, one physical component may embody several DSs and one DS may be embodied by several physical components. DSs at the conceptual level are useful for describing alternative or optional architectural solutions while models containing the concrete level are more useful to formalise knowledge about the architecture.

5.2.3 Configuration of the Functional Architecture

The CC approach allows decomposition of a system into an arbitrary number of sub-systems. The EF-M models of each sub-system is allocated to the DR of specific CCs. The CC is used to configure the EF-M tree. One approach to configure a system decomposed into sub-systems is to allocate each configurable sub-system to its own CC. A non-configurable sub-system (a sub-system without variants) on the other hand is allocated to the CC of its parent system. Figure 5.6 depicts this approach schematically.

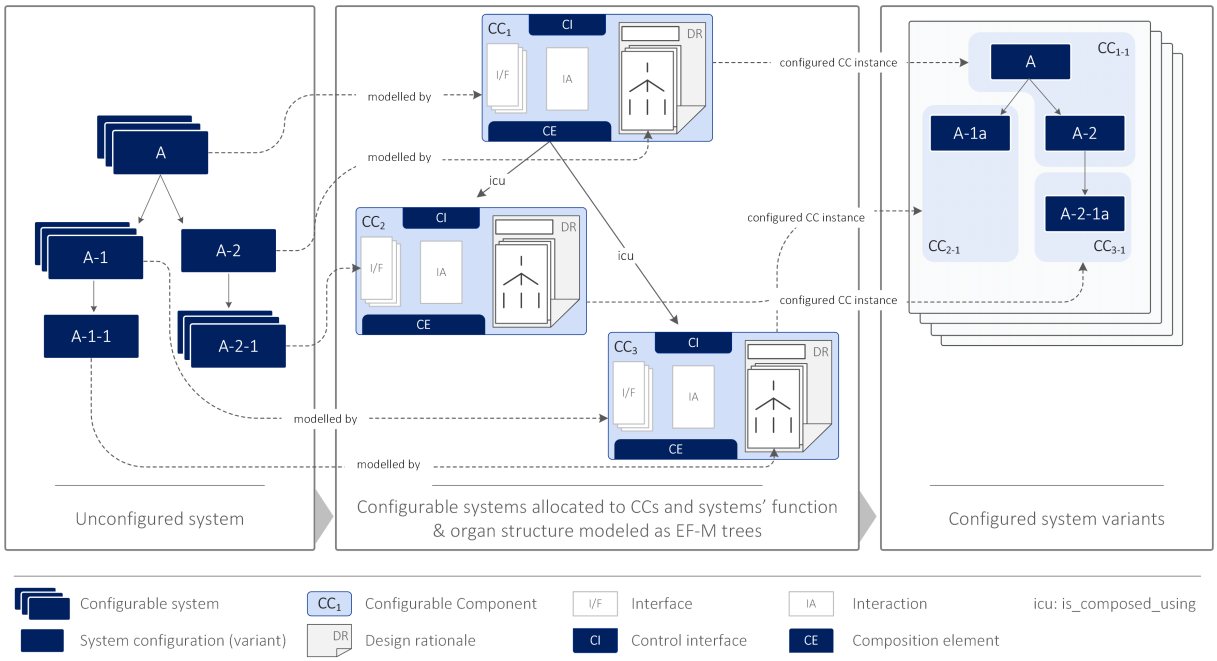


Figure 5.6: Configurable sub-systems allocated to CC objects used to configure a set of system variants.

Each sub-system's EF-M tree is allocated to the DR of the CC in the same way, Figure 5.7. VPs and VPVs within each CC are used to control the occurrences of optional and alternative relationships in the EF-M tree.

The complete system bandwidth defines the system's scope in terms of variety. In reality, all permutations (all theoretically possible system configurations) are seldom realised for mass customised products. Some combinations of VPVs are restricted due to technical, legal, economic or commercial reasons. Modelling a system with its complete system bandwidth as an EF-M tree captures knowledge about mutually exclusive DSs in a formal manner. There are however additional rules needed in order to constrain the configuration space. The *parameter maps* within

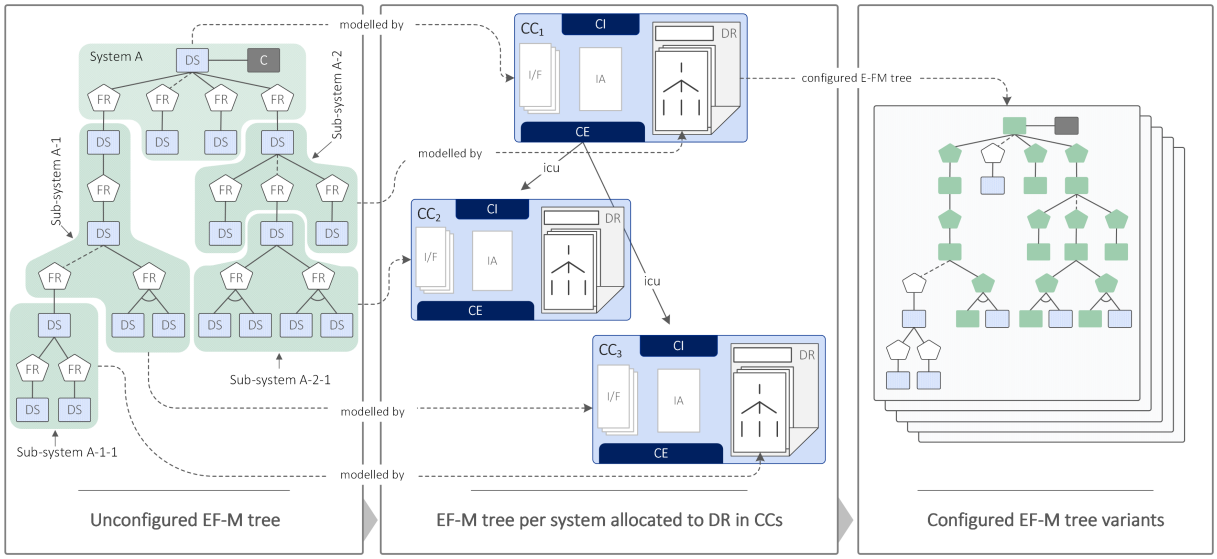


Figure 5.7: Configurable EF-M trees allocated to CC objects used to obtain a set of configured functional system architectures.

the CCs must be defined in a way that invalid combinations of DSs and FRs are restricted. The number of feasible system variants may be expressed as:

$$V_s = ((F_1^{bw} D_1^{bw})(F_2^{bw} D_2^{bw})...(F_i^{bw} D_i^{bw})) - C^{restricted} \tag{5.1}$$

Where:

V_s	= the number of system variants;
F_i^{bw}	= the functional bandwidth of a FR;
D_i^{bw}	= the design bandwidth of DSs fulfilling the FR;
i	= the amount of FRs in the system; and
$C^{restricted}$	= the number of restricted configurations.

For many systems, the technically and legally feasible number of variants is vast and results in great complexity. Specific configurations are typically excluded in order to balance the offer against complexity costs. Figure 5.8 illustrates how the functional- and design bandwidths span a configuration space consisting of infeasible configurations, excluded configurations and allowed configurations. Through early assessments of the product architecture it is possible to asses each feasible configuration based on for instance structural complexity. This can then be used to narrow down the configuration space based on data rather than on experience and heuristics.

5.2.4 Change Prediction Based on Design Dependencies

Adding flexibility to product platforms is one way of reducing costs of re-design (Suh, 2007). The CPM supports the identification of areas where flexibility is needed (Ferguson et al., 2014). And even if a pure platform approach is not pursued, the analysis can be used for early evaluation of concepts, identifying areas where the risk of re-design is high. Areas where this risk is low are more suitable for customisation. One should always strive towards implementation of initiating changes without triggering emergent changes. Assessment of where the risks of emergent changes are high can support early decision-making and the quantification and mitigation of risks associated with an intended change.

The probability of emergent changes may also serve as a knowledge base; re-design efforts

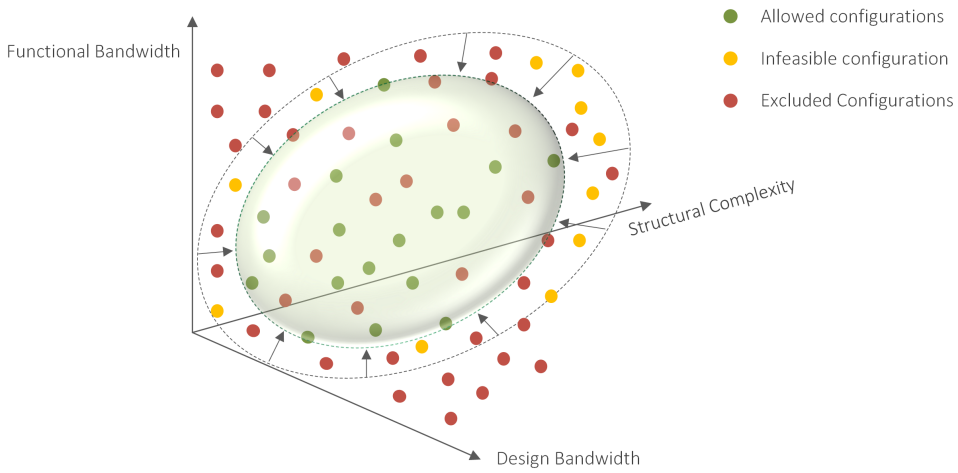


Figure 5.8: Illustration of theoretically possible configurations (variants) of a system based on the combined functional- and design bandwidths. Each configuration is illustrated as a dot colour-coded based on if it is infeasible, excluded or allowed. The structural complexity dimension plots configurations based on complexity.

may be pursued to make the product architectures more robust and high-risk interactions and complex couplings can be avoided in early development phases. This is beneficial as the task of finding a modular solution becomes increasingly complicated as variety, uncertainty and complexity increase.

The framework incorporates change propagation analysis based on design dependencies modelled in the function-organ domains. It makes use of input data from a CC product structure in order to support design engineers to understand and visualise design pre-conditions, to assess complexity and to make design decisions enabling modularity, commonality, flexibility and robustness. The design dependencies are modelled as *iw* relationships between DSs in the EF-M tree. The risk that a change to one DS propagates to other DSs in the system is then calculated. As the DSs are abstractions of physical embodiments, the dependencies are approximations of dependencies in the physical domain. The more levels the EF-M tree has, the more this data resemble reality. The interactions in the EF-M tree may be of types *spatial interaction*, *energy exchange*, *material exchange* or *information exchange*.

5.2.4.1 Quantification of Design Dependencies

As configuration data is added to the CC product structure, the branches of the EF-M tree that may coexist are defined. Design dependencies are then identified between the lowest level DSs of each branch (Figure 5.9) and modelled as iw relationships.

Dependencies with positive as well as negative effects on system performance are modelled. An interaction may be required (e.g. contact pressure between surfaces in a joint) or should be avoided (e.g. contact that risks chafe cables). The iw relationship between DSs captures the direction of the interaction (one-way or two-way), the probability that a change will propagate along the interaction and the impact of such a propagation. Likelihood and impact are both modelled with a value between 0 and 1 or between -1 and 0. A minus sign represents an interaction with negative effects on the system and vice versa.

The identification and quantification of design dependencies require expertise and is not necessarily straight forward. It may be based on experience, tests, trade-off curves or by other means. The quantifications will however typically be an approximation, in particular for novel designs.

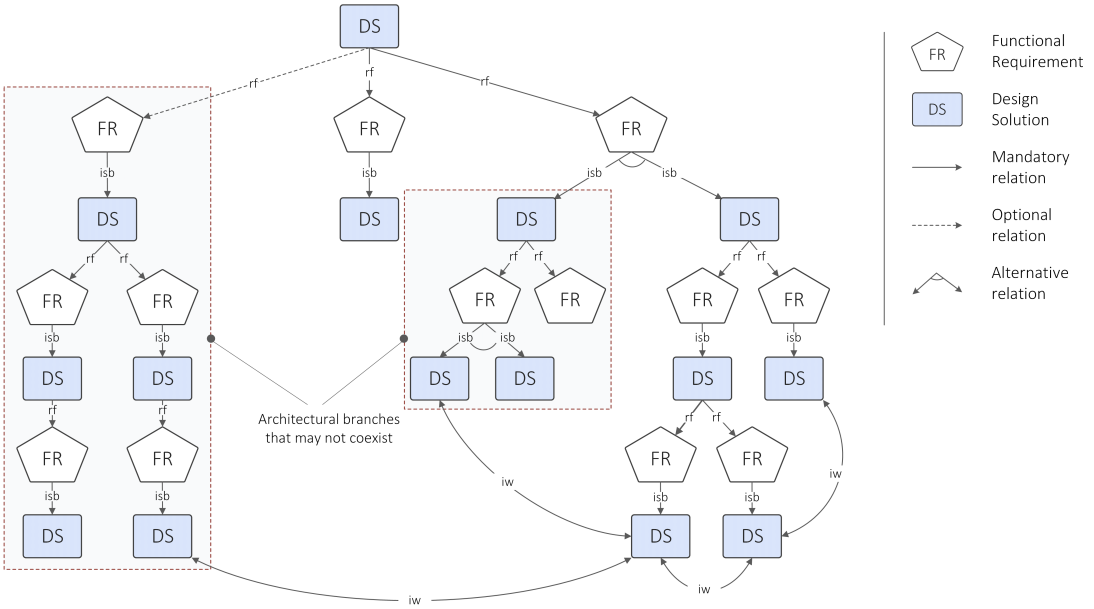


Figure 5.9: Generic EF-M tree with design dependencies modelled as interactions (*iw* relationships between DSs). Dependencies between architectural branches not coexisting are not modelled.

A configured CC product structure results in a definite set of networks with CC instances. Equal to the number of unique networks are the number of unique EF-M tree configurations. Each configuration represents the design rationale of one system variant. As architectural branches will be *true* or *false* in different configuration, each one possesses its own unique dependency structure which can be represented by a DSM. The framework makes use of the Configurable Component Modeller (CCM) software to model CCs, EF-M trees and to generate DSMs for each system configuration.

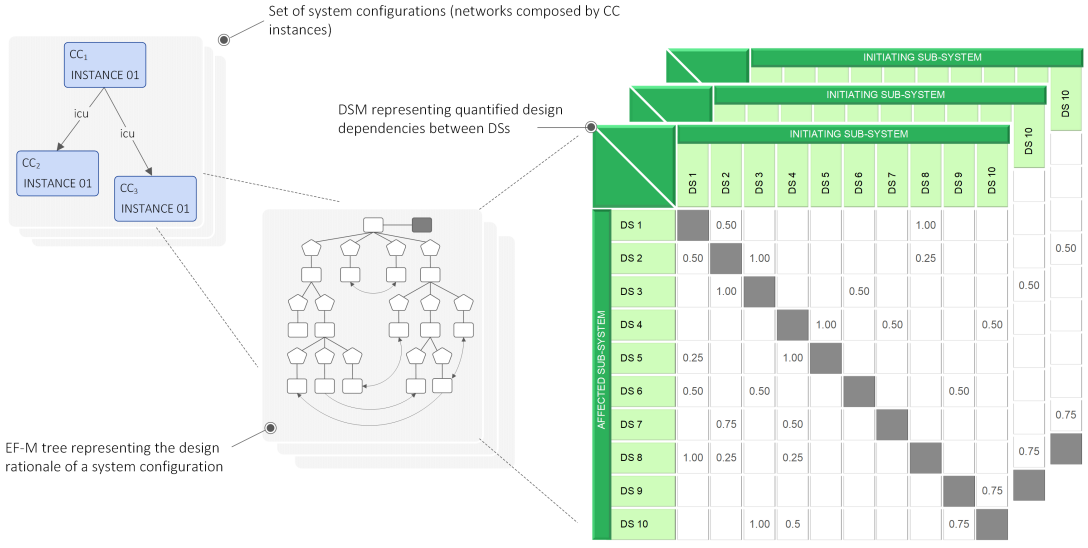


Figure 5.10: Illustration of how DSMs with change likelihood and impact between interacting lowest level DSs of a system are generated based on a configured CC product structure.

A design dependency required to achieve a desired behaviour of the system creates an *is influenced by (iib)* relationship between the DS and the parent FR of the DS it interacts with. The iw as well as the iib interactions can be captured in a DSM. The former may be used to

assess *physical modularity* (design couplings). The latter represent how coupled a FR is and may be used to assess *functional modularity*. That is, the number of DSs that needs to cooperate in order to fulfil the FR. Figure 5.10 illustrates schematically how DSMs with iw relationships between DSs are derived for each system variant. All lowest level DSs are represented and the cells are populated with the product of change likelihood and impact.

5.2.4.2 Change Propagation Analysis

The DSMs are used as input to calculate the combined risk of change propagation. The CPM not only allows assessment of direct but also indirect interactions. It can unveil parts of the system highly influenced, by not directly affected by a change. The methodology mathematically calculates how changes are likely to propagate through a system. In this framework, CPM is used to analyse how the DSs of a system affect each other and the system's ability to fulfil functional requirements. Moreover, it unveils whether DSs absorb or multiply changes. This allows identifying parts of a system sensitive to changes, possessing greater risk, and specific variants with greater structural complexity. The result can be used to improve the design. It can also be used to assess individual configurations and identify parts of a system suitable for differentiation and re-use.

6

Illustrating Case

This chapter illustrates the proposed framework by applying it on a simplified sub-system from the studied company, namely exterior Rear-view Mirrors (RVMs). They are a typical example of a sub-system afflicted by many of the factors driving variety and variety-induced complexity.

Exterior rear-view mirrors are very complex. We talk about 30 to 35 system variants per vehicle model only for rear-view mirrors, and that does not include colours.

- System Responsible, R&D

By incorporating variability in the definition of the product architecture it is possible to let the logic behind how the product is designed control how it is configured. Modelling the architecture as a system-of-systems enables encapsulation of variety and thus concurrent development of the architecture.

This case study focuses on the RVM architecture for a vehicle family of one sedan and two wagons with different chassis height. As physical dimensions vary across vehicle families, component re-use is often futile. Design solutions however mostly remain the same, why re-use of design knowledge might serve as a better approach.

Figure 6.1: Rendering of an exterior RVM.

The RVM design has different embodiments depending on if the mirror is mounted to the left or to the right on the vehicle. Most of the variety is embodied in the *incomplete RVM*, which is the shaded part in Figure 6.1[1]. There are 44 unique variants of the incomplete RVM for this particular vehicle family. Throughout most of this case study colour options are neglected as they contribute little to variety-induced complexity from a design perspective.

The system is modelled as a network of CCs. Each CC is autonomously configurable and contains a design intent — how the design and variability is realised — modelled as an EF-M tree. A configuration rationale within each CC defines how they may be configured. Control in-

[1]Managing the incomplete RVM separate from the housing cover is a strategy for simplifying assembly as well as administration in planning and development. With this approach, 44 plus 30 variants (expressed as unique parts in the PDM system) need to be managed instead of 660. Thus variety is created closer to final assembly.

terfaces and composition elements allow interchanging configuration data between the CC such that the complete system may be configured.

Exterior RVMs are an example of a complex mechatronic system characterised by a large configuration space. The physical interface that mounts it to the front door is quite trivial but there are several functional dependencies to other sub-systems as well. Variety has increased during recent years, whilst the dimensions of the mirror have shrunk. As the architecture is mature, it is analysed bottom up. This means that allocation of design solutions to CCs is based on breakdown of a known design. Only the variety provided to the market place is modelled.

The framework allows a more systematic allocation of DSs to CCs based on an assessment of design dependencies. In addition, it allows assessing the complexity of individual variants, the impact of adding new variants, and if the system bandwidth makes the architecture robust against changes. Three criteria are used to evaluate system variants (Raudberget et al., 2015):

- Ability to integrate
- Design process efficiency
- Development process efficiency

Ability to integrate quantifies how easy it is to re-use the system in a new product context. It is calculated based on system couplings, as less coupled solutions are easier to integrate and are more suitable candidates for change. Design process efficiency expresses the complexity of an architecture in terms of design iterations. The further a change is likely to propagate through the design, the more design iterations are needed to accommodate it. Development process efficiency indicates the amount of resources necessary to re-design an architecture in order to reduce the inherited risks imposed by wanted *risk absorbers* and unwanted *risk multipliers*.

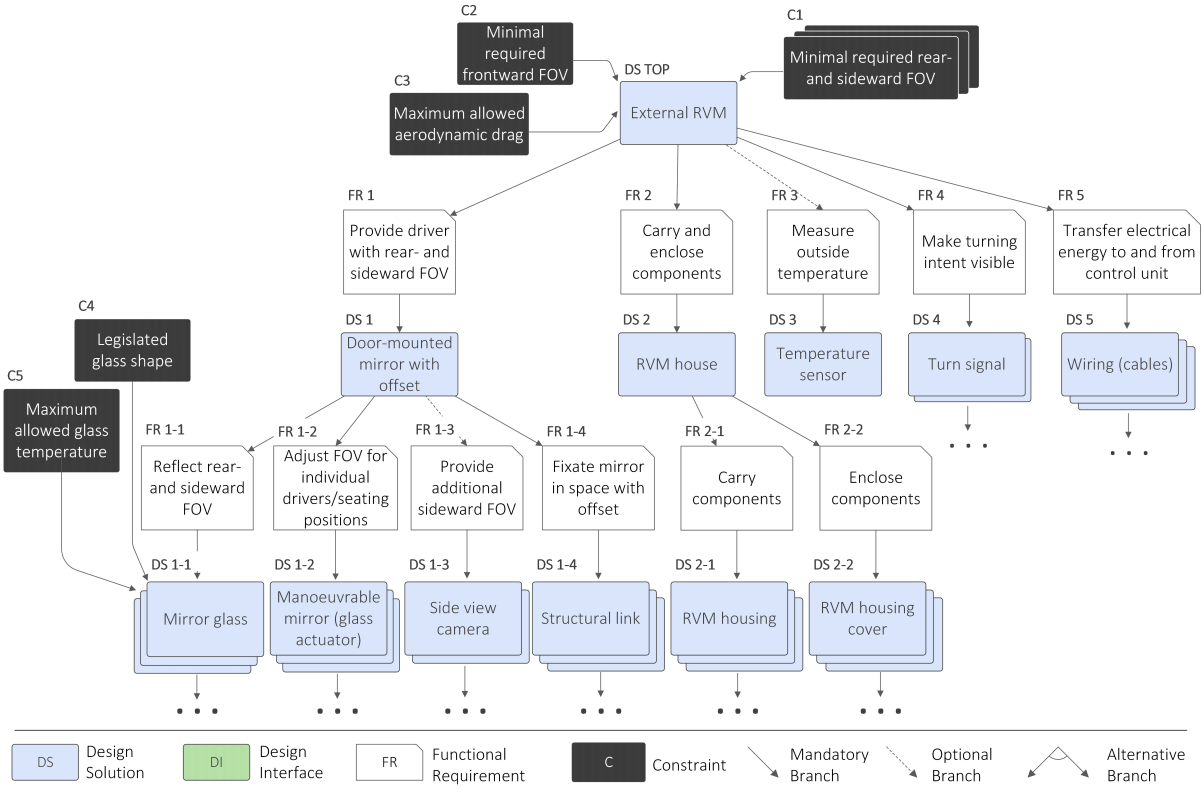


Figure 6.2: The top level DS decomposed into five top level FRs. The main architectural branches of DS 1 and DS 2 are decomposed one additional level. Layered DSs indicate alternative DSs.

The ultimate aim is to incorporate these criteria in decision-making. The remainder of this chapter however describes how the *model* is constructed and assessed. Not how the criteria are incorporated in decision-making.

6.1 Modelling the Rear-view Mirror

The RVM is a critical safety system. Its main FR is to provide the driver with a rear- and sideward FoV. It has four additional FRs at the highest level. These are solved with DSs as specified in Figure 6.2. Variety exists already at the top level as a temperature sensor is located in right-hand side designs only. The main constraints are, as specified in Figure 6.2: The minimum legislated effective rear- and sideward FoV; How much of the frontward FoV the mirror is allowed to conceal with its size; And the maximum aerodynamic drag it is allowed to generate. The first constraint sets a limit for how small the mirror glass may be. The latter two define a limit for how large the mirror may be. This trade-off ultimately affects the design and physical placement in relation to the driver.

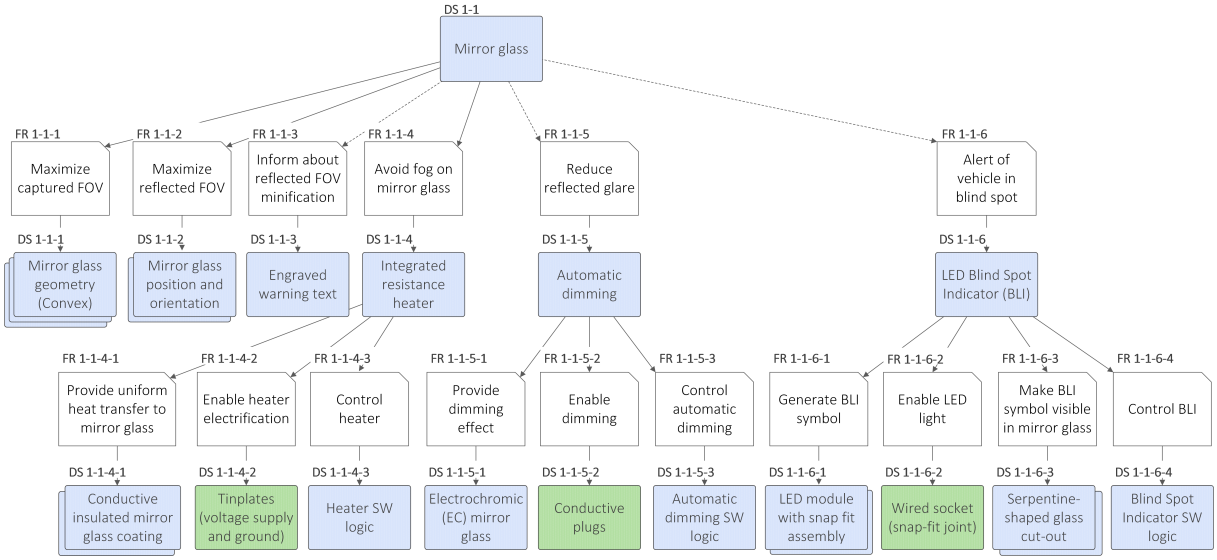


Figure 6.3: Complete decomposition of DS 1-1 *mirror glass*, showing its optional FRs.

The design has additional safety features, all optional, as specified in Figures 6.2 and 6.3. These include an extended sideward FoV (solved by a camera), indication that another vehicle exists in the blind spot (solved by a signalling LED light) and reduction of reflected glare from trailing vehicles (solved by automatic dimming enabled by an electrochroamatic mirror glass). These result in optional architectural branches.

Essentially every architectural branch has variety in the form of optional FRs or alternative DSs[2]. The ability to reduce the vehicle's width while parked (Figure 6.4) is for instance solved by alternative DSs. All configuration-related dependencies need to be captured and modelled. These define compatibility between DSs or the combinations of feature variants that are viable from a commercial perspective. The complete function-means model of the RVM is appended in Section A.1 of Appendix A.

[2]By letting design parameters represent varying characteristics, alternative DSs need not be modelled. For example the right-hand and left-hand side geometries of the mirror glass are not modelled as alternative DSs as they do not represent unique interactions.

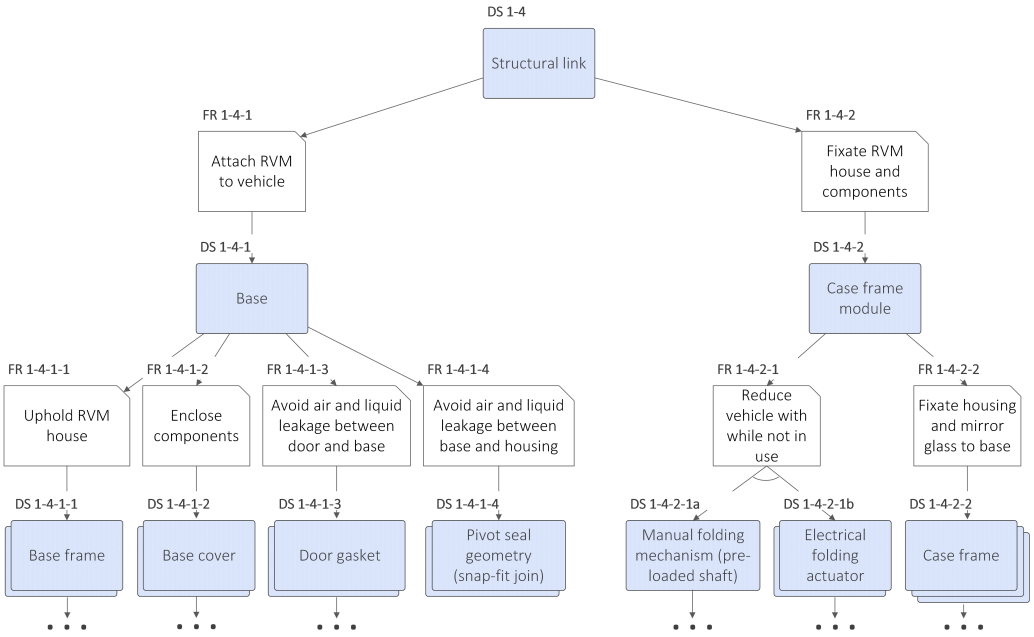


Figure 6.4: Additional decomposition of DS 1-4 *structural link*.

6.2 Specification of the Rear-view Mirror Bandwidth

Variability needs to be explicitly incorporated into the model in order to make the architecture configurable. This section provides a description about the RVM's variety. The system's *complete bandwidth* results from its combined functional- and design bandwidths. The functional bandwidth streams from market segments and customer needs while the design bandwidth is triggered by internal requirements, legal requirements and design decisions.

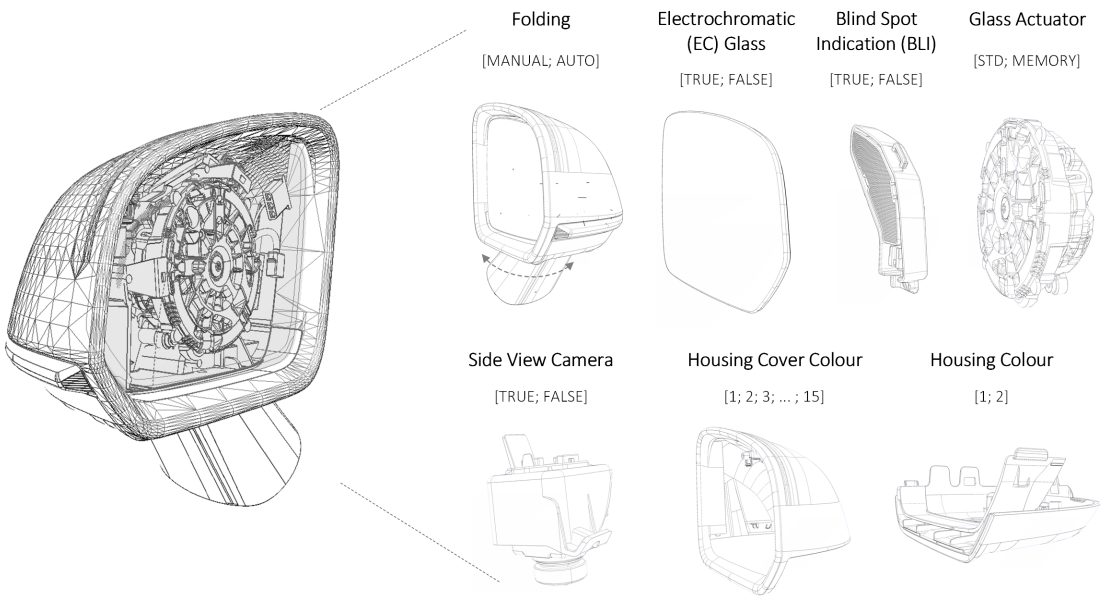


Figure 6.5: The RVM's functional bandwidth; feature families and their variants.

If one disregards colour, the RVM is influenced by five *commercial feature families* — perceived by the customer — as shown in Figure 6.5. Each feature family has two variants, resulting

in $2^5 = 32$ possible combinations (960 considering colour). Five VPs with two VPVs each are used to model the functional bandwidth. Figure 6.6 expresses all possible permutations. Variety steering is used to reduce the functional bandwidth with 78%, resulting in a configuration space of manageable size. It is defined by seven feature bundles that lie along a continuum from low-cost-low-performance[3] too high-cost-high-performance as Figure 6.6 shows.

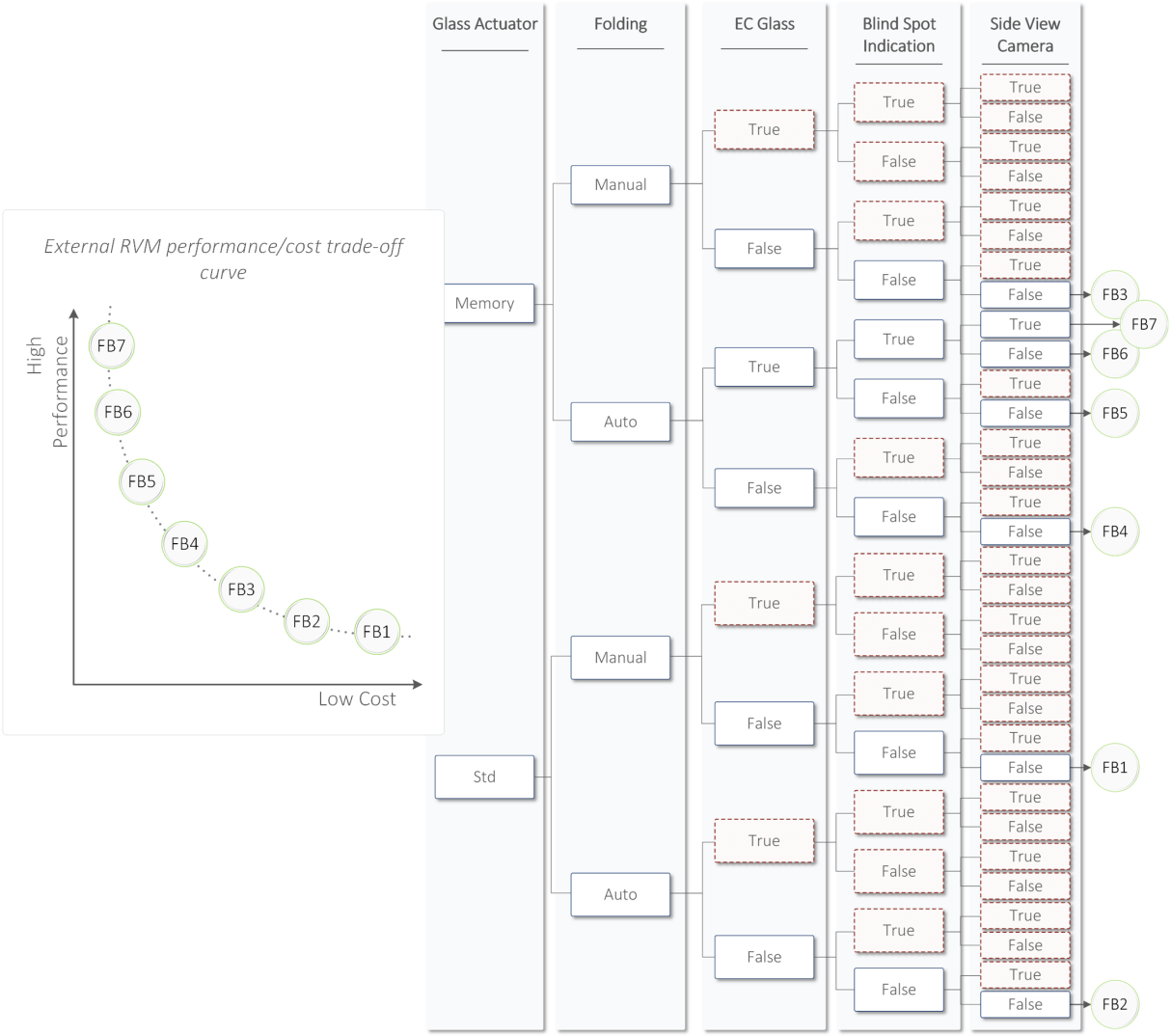


Figure 6.6: Variant tree and feature bundles for the RVM.

The complete bandwidth is solution-dependent - it is influenced by technical and legal feasibility. The most important interaction is the Human-machine Interface (HMI) that provides a visual of the area otherwise invisible to the driver. This interface constitutes a boundary condition as the minimum FoV is both critical and regulated. The FoV is influenced by the shape of the mirror glass where a non-planar mirror (convex or aspheric) yields a larger FoV and a unit magnification mirror (planar) yields a smaller. For planar mirrors, the effective FoV is a result of three variables: *(1)* The dimensions of the mirror glass (a large mirror yields a larger FoV); *(2)* The aim and orientation of the mirror glass (a large portion of the vehicle visible in the mirror yields a smaller FoV); And *(3)* the distance between the mirror glass and the driver's

[3]Low cost configurations are, from strategic standpoint, important both to present a competitive listing price, attract customers with less purchasing power and to increase the attractiveness of the more profitable 'high-value' configurations. Thus they are often included in the offer, although with low volumes.

eyes (a short distance yields a larger FoV). Non-planar mirrors add a fourth variable: *(4)* The convexity (a high convexity yields a larger FoV). Thus, varying body types and heights of the chassi trigger unique glass geometries.

The glass shape is regulated per market. E.g. North America requires a planar glass on the driver's side. The passenger's side always carries a convex glass. Some markets however also require a warning text on the convex glass as it provides a minified FoV. Ergonomic constraints for mirror sampling and forward FoV place further requirements on the dimensions. These interrelated variables, together with the steering position trigger internal variety manifested as unique geometries.

		Commercial Variety							
		FB 1	FB 2	FB 3	FB 4	FB 5	FB 6	FB 7	Feature Bundle
		Manual	Auto	Manual	Auto	Auto	Auto	Auto	Folding
		Std	Std	Memory	Memory	Memory	Memory	Memory	Glass Actuator
		False	False	False	False	True	True	True	EC Glass
		False	False	False	False	False	True	True	Blind Spot Indication
	Mirror Glass Shape	False	False	False	False	False	False	True	Side View Camera
Internal Variety	LHD, Aspheric LHS	var_01	var_02		var_03	var_04	var_05	var_06	
	LHD, Planar LHS			var_07		var_08	var_09	var_10	
	LHD, Convex Rx RHS	var_11	var_12		var_13	var_14	var_15	var_16	
	LHD, Convex Ry RHS				var_17	var_18	var_19	var_20	
	LHD, Convex Rx RHS, English Warning			var_21		var_22	var_23	var_24	
	LHD, Convex Rx RHS, Arabic Warning					var_25		var_26	
	RHD, Convex Rx LHS				var_27	var_28	var_29	var_30	
	RHD, Convex Ry LHS				var_31	var_32	var_33	var_34	
	RHD, Aspheric RHS				var_35	var_36	var_37	var_38	

Figure 6.7: Variety matrix indicating the developed RVM variants (simplification of the actual Excel-based variety matrix used by design engineers at the studied company to overview the variety in their design area where each variant is represented by a part number in the PDM system).

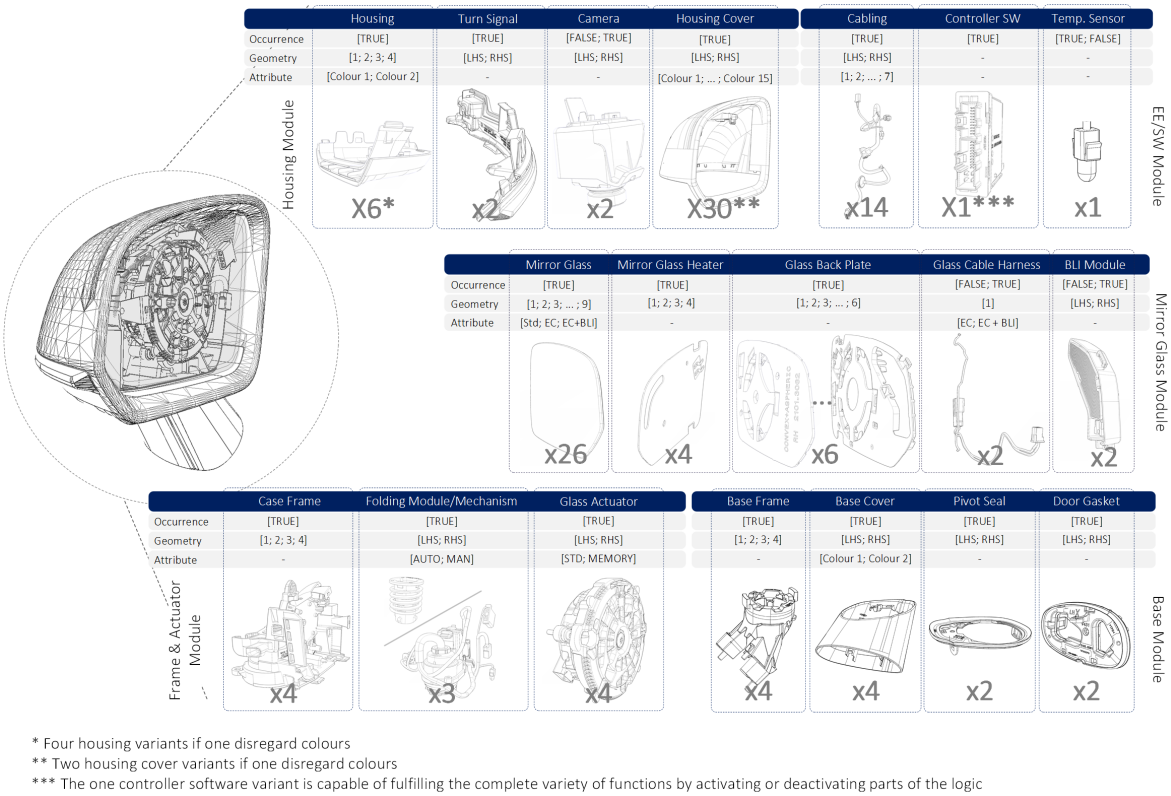
The mirror glass is influenced by only two of the commercial feature families (glare reduction and blind spot indication). Even so, without any variety steering practices, this would result in 52 unique mirror glass designs. Through variety steering however, 26 designs remain. A variant tree for the mirror glass is presented in Figure A.10 of Appendix A[4]. There are now 63 feasible RVM variants from a commercial, technical and legal perspective. These are reduced to the offered 38 by disqualifying specific combinations of feature bundles and design solutions as illustrated in the variant matrix in Figure 6.7.

A vast majority of the components composing the RVM come in two or more configurations, Figure 6.8. There are thus theoretically approximately $1.16\text{x}10^{10}$ possible ways of assembling an RVM out of which 38 are valid ($5.22\text{x}10^{11}$ adding colour to the equation). Without any variety steering there would be 288 feasible RVM variants (8640 considering colour). This would increase the theoretical number to approximately $5.65\text{x}10^{11}$ combinations ($3.39\text{x}10^{13}$ considering colour).

6.3 Configuring the Rear-view Mirror

Out of all possible configurations of the RVM, only the 25 that actually affect the architecture's behaviour are modelled in this section. The sub-systems composing the mirror are modelled as

[4]The selection of the complete bandwidth is typically characterised by heuristics, experience and competitor behaviour. Not seldom, sub-system variants are excluded from the offer as they turn out to raise low interest on the market. This reactive variety steering however has low positive impact on pd, as time and resources are already wasted. The sub-system variants still need to be developed, tested and verified.



* Four housing variants if one disregard colours
** Two housing cover variants if one disregard colours
*** The one controller software variant is capable of fulfilling the complete variety of functions by activating or deactivating parts of the logic

Figure 6.8: Illustration of RVM components and their respective variants.

CCs as shown in Figure 6.9. The CCs and their EF-M trees are modelled in CCM. Each CC's complete bandwidth is represented by the VPs (in bold) and their respective parameter set (in brackets) in its CI as described in Figure 6.10. The CI contains the variability visible outside the CC and is used to configure the complete system.

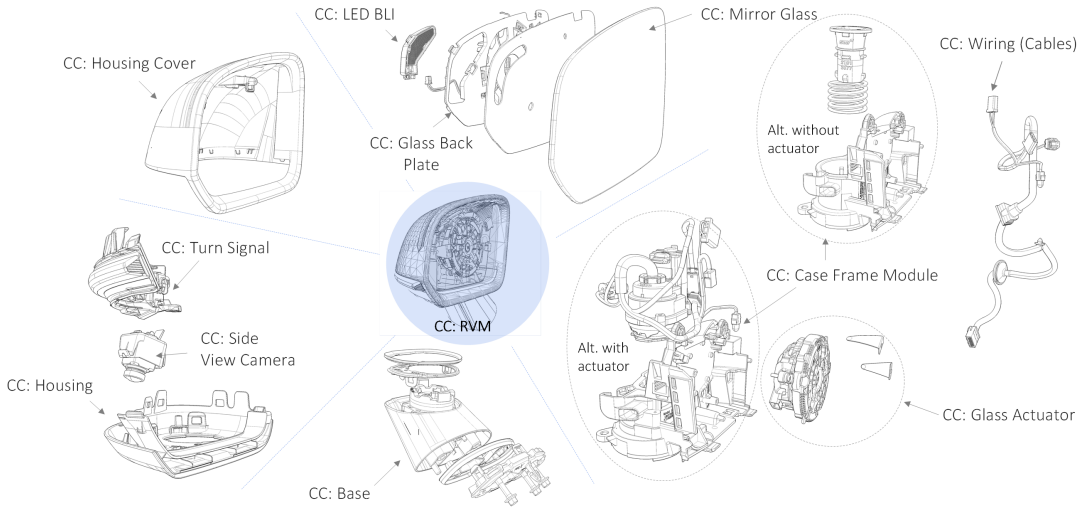


Figure 6.9: Illustration of the RVM components modelled as individual CCs.

Architectural branches are allocated to their respective CCs by letting a CC be represented by a DS. The CC *mirror glass* is for example represented by the DS *mirror glass*. One unique EF-M tree per RVM variant is obtained by using VPVs and Boolean operators to 'turn on' or 'turn off' specific *requires function (rf)* or *is implemented by* relationships (Figure 6.11).

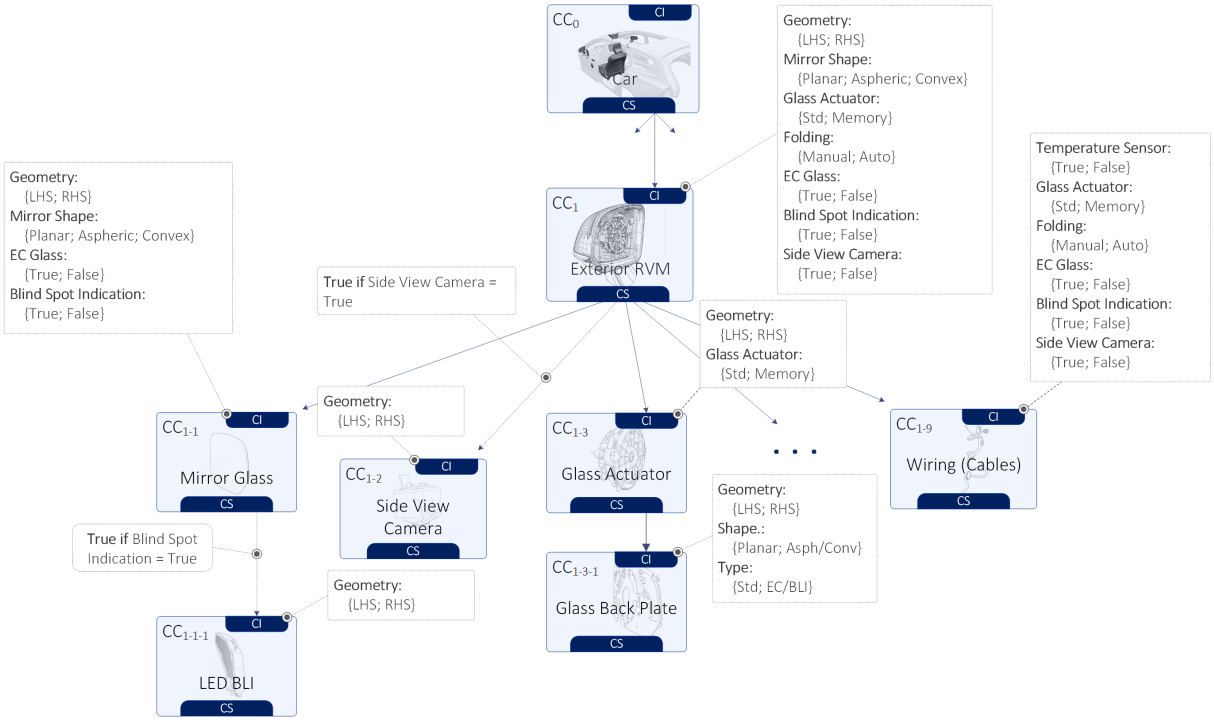


Figure 6.10: Illustration of the CCs used to configure the RVM and their respective VPs and VPVs.

An EF-M tree per configuration is obtained when the CCs are instantiated. More importantly this automates the process of creating DSMs, obtained by first specifying design dependencies, elaborated in the next section.

6.4 Specification of Design Dependencies

Interacts with (iw) relationships model design dependencies between DSs. Upon configuration, all such relationships to or from a 'turned off' DSs are excluded for that variant. As some branches are more coupled or have more severe likelihood or impact values, more complex configurations can be identified. For example the *planar mirror glass* has a higher probability of affecting the mirror's position and orientation (and vice versa) than its non-planar counterparts.

The CCM can model four types of dependencies: *spatial interaction*; *energy exchange*; *material exchange*; and *information exchange*. For each interaction, the direction (one-way or two-way), likelihood (between -1 and 1) and impact (between -2 and 2) were specified.

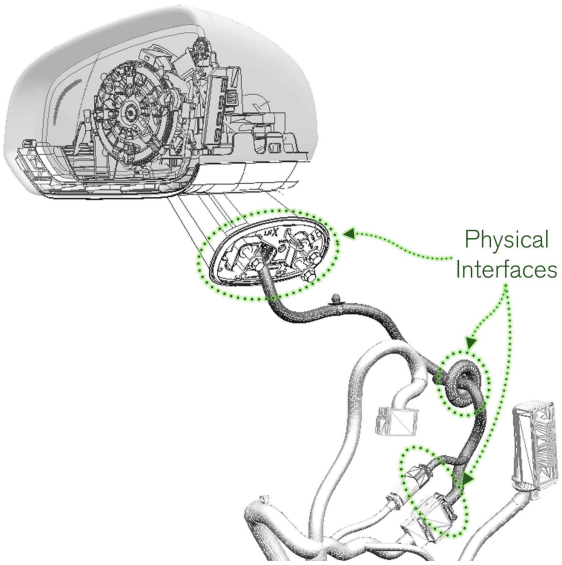


Figure 6.12: Illustration of the RVMs physical interfaces to the vehicle.

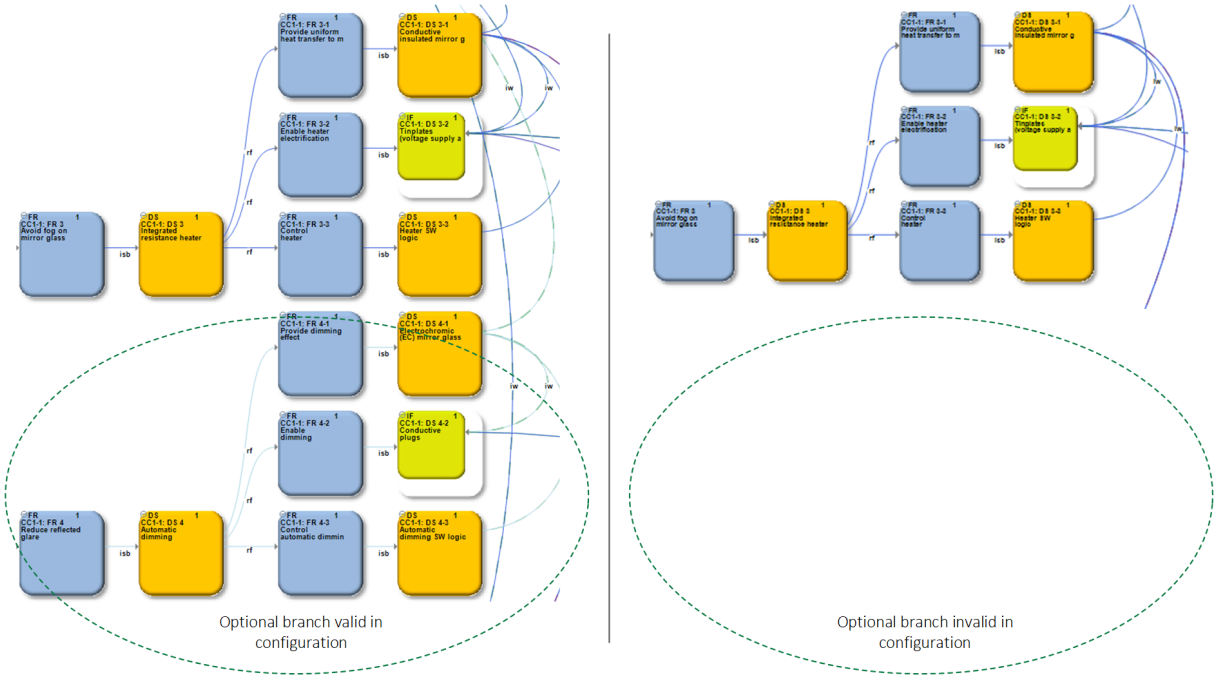


Figure 6.11: Extract from the CCM model. To the left is a configuration where the automatic dimming branch is valid and to the right a configuration in which it is turned off.

Quantification requires extensive experience and engineering work. In absence of real data, impact and likelihood were estimated. To serve the purpose of this study, four levels of likelihood were used. Definite interactions were quantified with likelihood '1'. Sensitive dependencies were quantified with likelihood '0.75'. Unlikely interactions, such as the geometry of the convex mirror glass affecting the glass' orientation and position, were quantified with '0.25' and rather likely ones with '0.5'. Interactions crucial or devastating for key functionality were quantified with impact 2 and -2, respectively. All others with 1 or -1.

Sub-systems that interact with the RVM were also modelled to capture important dependencies. Apart from the physical interfaces to the door (Figure 6.12), several information-based interfaces exist. HMIs in the door are for instance used to control functionality. The signal that activates the automatic dimming is triggered based on input from a sensor in the interior mirror. The signal that turns on the BLI is triggered based on input from a radar located in the rear of the vehicle, and so forth.

The EF-M tree visually depicts not only *iw* but also *is implemented by (iib)* relationships. That is, if one DS needs an *iw* relationship in order to fulfil its FR. That FR is coupled in the sense that it is implemented by multiple DSs. This can serve as a valuable complexity metric. Many such coupled FRs are however necessary as a function might require a sensor, a software component and wiring to function. This study thus exclusively uses *iws* in the evaluation.

6.5 Change Propagation Analysis

The ways the system interact are captured with *iw* relationships between the lowest level DSs. For each variant the set of DSs and *iws* are unique. Thus 25 unique DSMs were obtained, describing the couplings of each specific configuration. The 25 DSMs were generated from the CCM model. The DSM for one particular configuration is exemplified in Figure 6.13 .

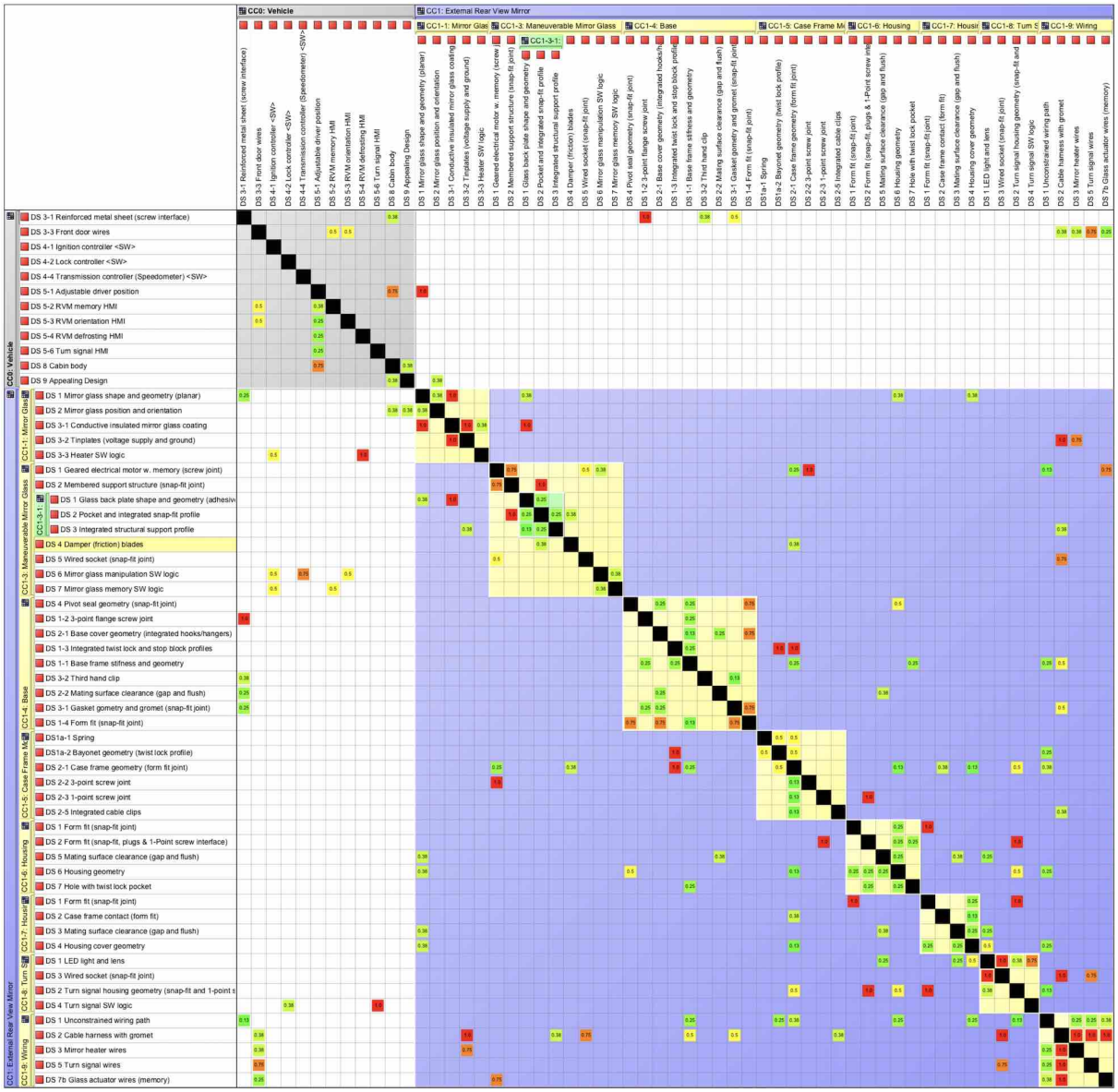


Figure 6.13: DSM with risk values for one particular configuration.

When studying how *iws* diverge between configurations, it is clear that the number of DSs increase as variants go from low-cost-low-value ones to high-cost-high-value ones. This is intuitive as an increased number of FRs results in a larger number of DSs and thus also a larger number of couplings. The different geometries of the mirror glass and overall RVM does not influence the number of couplings. The number of DSs increased from 46 to 72 between the configuration with the least compared the most functionality. The corresponding increase in couplings is 107 to 174, respectively.

As the system is rather integral, with a high number of couplings, and as the more complex configurations also yield a higher market value, it is difficult to only visually assess the architecture. That is, it is difficult to draw conclusions about what the difference in couplings actually means in terms of cost versus performance.

CPM provides a way to further analyse each configuration. It uses the DSMs as input, simulates a change to a DS and mathematically calculates the risks of that change propagating through

Figure 6.14: Combined risk matrix for a left-hand side mirror with a planar mirror glass and configured with a glass actuator with memory, manually foldable and without dimming, blind spot indication or camera. A majority of the DSs have high ingoing and outgoing risks, an evidence of the integral nature of the design.

the system. This can provide information about parts of the system that are suitable candidates for differentiation, for change or information about individual variants with high complexity costs as it can unveil parts of the system influenced, by not explicitly affected by a change.

The CAM (Wynn, 2007; Wynn et al., 2009; Wynn, Wyatt, et al., 2010) application was used to apply the methodology. It generates a result in the form of a *combined risk matrix*. Such a matrix is shown in Figure 6.14. From the combined risk matrix it is clear how integrated the RVM design is. The result from this matrix can be interpreted and displayed in multiple ways, e.g. in a *risk portfolio plot*. (Figure 6.15). It provides a graphical view of the overall risks of change propagation. In particular it distinguishes between DSs that absorb and multiply changes. This allows identification of configurations or parts of a system sensitive to changes. A DS is plotted with its average outgoing risk on the x-axis and its average incoming risk on the y-axis. DSs suitable for differentiation are thus located amongst the risk absorbers in the bottom left quadrant.

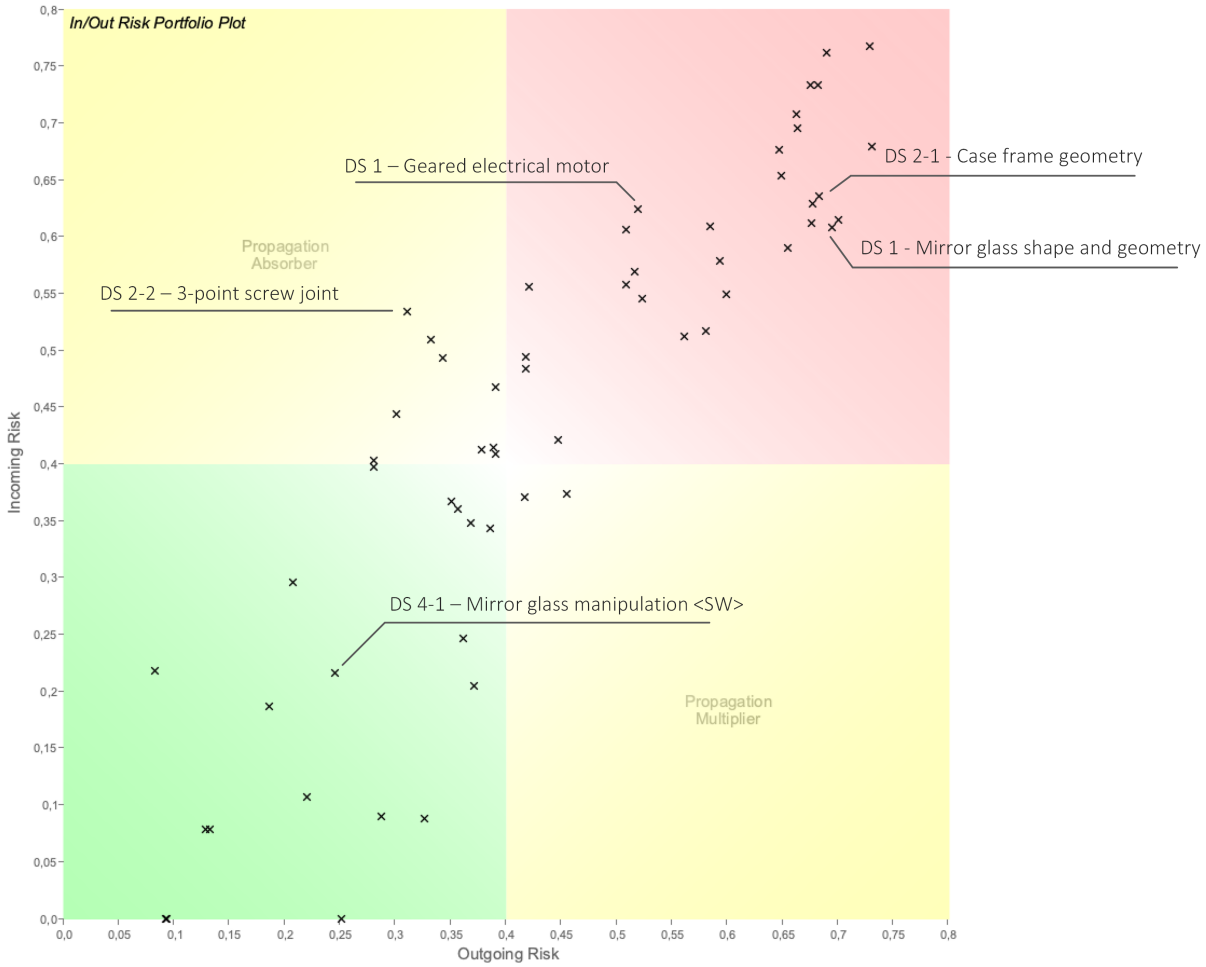


Figure 6.15: Example of a risk portfolio plot containing the DSs of a left-hand side mirror with a planar mirror glass and configured with a glass actuator with memory, manually foldable and without dimming, blind spot indication or camera. Half of the DSs in this configuration appear in the high-risk quadrant.

Figure 6.15 highlights some examples of DSs that exhibit different behaviour. It is notable that half of the design solutions are located in the quadrant with both high ingoing and outgoing risk. The geared electrical mirror glass motor (actuates force on the mirror glass) serves as a link between the glass and the structural elements of the mirror. The case frame is the main structural element as it serves as a link between the glass motor and the mirror's base. It is kept in place by both the housing, the base and the housing cover so any change is thus very likely to both propagate to and from the case frame.

The software that controls the actuating force on the mirror glass has less risk (a fact shared by most software solutions). The screw interface to the vehicle door is standardised and successfully designed as a propagation absorber. The incoming risks are however still high as it interacts with the base of the mirror and as it provides a path for the cables to and from the mirror.

Integration ability, design process efficiency and development process efficiency were calculated for each of the 25 configurations. The values are presented in the plot in Figure 6.16. Five clusters with similar configurations are easily identified. The more functional content a configuration has, for instance if it is electrically foldable or is equipped with a camera, the more complex it generally is. The least complex cluster for example contains the configurations with manual folding (without an electric motor). The most complex cluster contains the configurations with

dimming capability, with blind spot indication and with camera. It also contains the right-hand side configurations with dimming capability, blind spot indication but without camera.

A right-hand side mirror often belongs to a more complex cluster than a left-hand side mirror with identical functional content. This is due to the fact that a right-hand side mirror is configured with a temperature sensor which due to its central placement and interface to the infotainment system has a handful of unique design dependencies.

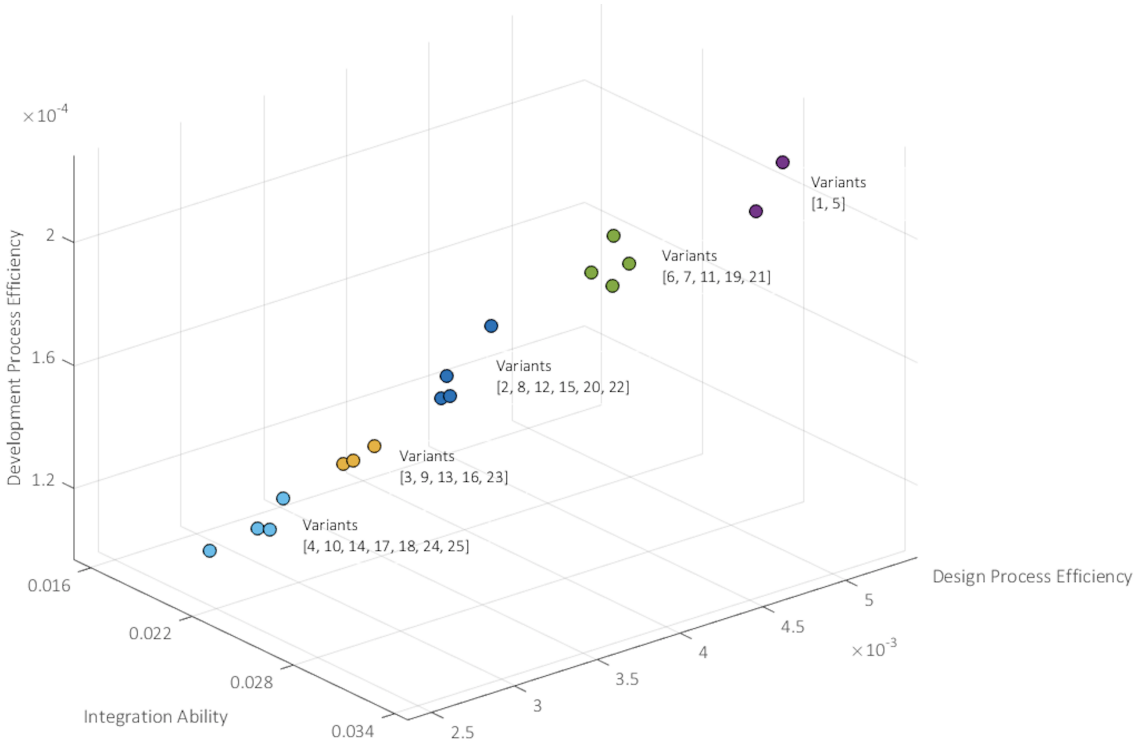


Figure 6.16: Rear-view mirror configurations plotted based on calculated integration ability, design process efficiency and development process efficiency. Configurations are colour-coded to belong to one out of five clusters (some configurations have identical values).

Although the integration ability is identical (for a left-hand side mirror with identical functional content) with a planar, a convex or an aspheric mirror glass, the planar glass is constantly an outlier in each cluster. These configurations have slightly lower design- and development process efficiency than those with a non-planar mirror glass. Although the number of design dependencies are the same, a change to the planar mirror glass is more likely to negatively affect the FoV. These configurations thus possess a higher risk of change propagation.

7

Discussion

This chapter provides a discussion about the thesis mission, approach and findings. First, it is dedicated to answering the research questions. Second, the contributions are discussed. Finally, the quality and validity of the findings are considered along with an ethical discussion.

7.1 Answers to Research Questions

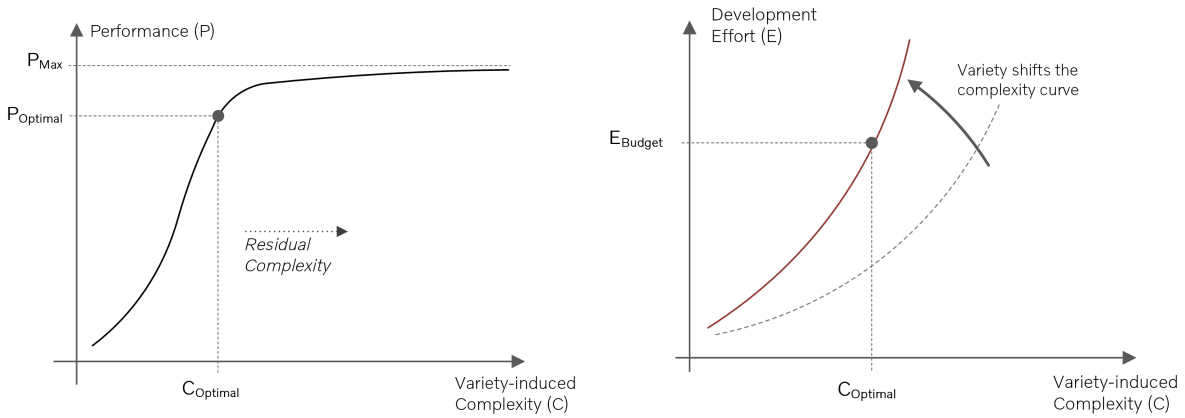
The answers to the research questions are derived from a combination of published literature, empirical results and the case study. Here, the answer to each of the three questions are summarised and discussed.

> ***RQ 1)*** *What are the main challenges that can be traced back to dealing with variety-induced complexity in pd?*

Platforms, with an appropriate bandwidth, allow a multitude of product variants to be derived with functionality diverse enough to serve multiple customer segments and preferences, while capitalising on commonality and re-use. The development of such platforms is very different from developing individual products. It makes designing 'good' solutions an increasingly complicated task. Development deals with the aggregated complexity and trade-offs of multiple product variants. Thus, detailed knowledge about all product variants to be fulfilled is important. The difficulty lies in managing the high amount of information representing complicated, often contradicting design pre-conditions. Typical challenging pre-conditions are physical and functional design dependencies, often between sub-systems of varying maturity due to design re-use. With different maturity, safety margins shrink as flexibility to change diminishes when the surroundings cannot absorb changes as easily.

Sinha and de Weck (2016) identify the value of a system as a function of performance over development effort, where the latter grows exponentially with increasing complexity. They assume that system performance saturates at some point along an S-curve. The *complexity budget* identifies the optimal level of complexity along this S-curve. I.e. the performance where complexity no longer adds value. The 'goodness' of a system is as this study has shown, not only a function of performance and development effort. The system's contribution to the complexity costs of the product architecture is an additional factor. Moreover, the development effort is not limited to a single project but related to the complete product architecture.

This study assumes that the relationship between performance and development effort can be extended with variety. The performance of a generic system is assumed to saturate at some point P_{max}. The curve in Figure 7.1b is assumed to be steeper for configurable products as this is more complicated and prone to waste. With knowledge about the behaviour of this relationship, one may identify the optimal performance level, $P_{Optimal}$. This is where variety-induced complexity turns from value-adding to residual, Figure 7.1a. The concept of a complexity budget is thus applicable also for the phenomenon of variety-induced complexity.



(a) Relationship between system performance and variety-induced complexity.

(b) Relationship between development effort and variety-induced complexity.

Figure 7.1: Illustrations of the relationships between system performance, variety-induced complexity and development effort (applied from Sinha and de Weck (2016)).

The complexity budget are used to summarise the main challenges of variety-induced complexity identified in this study:

1. Generating knowledge about the relationship between variety-induced complexity, performance (customer utility) and development effort — *doing the right things.*
2. Shifting the complexity curve in Figure 7.1b downwards to the right by designing 'good' solutions — *doing things right.*
3. Shifting the complexity curve in Figure 7.1b downwards to the right more efficiently spending the complexity budget on value adding activities — *spending time on the right things.*

Has this thesis provided a profound enough answer to the research question? Many problems have been highlighted, in particular established structures and practices, waste and insufficient, fragmented and cryptic information. Although a bit abstract, this study provides an umbrella for zooming in on these three main challenges. Thus, I believe that an answer to the first research question has been provided. The capabilities defined in Section 5.1 provide a structure for zooming in on these challenges. By discussing them, the second research question is answered.

> ***RQ 2)*** *Given a subset of challenges (from RQ 1); what are the root-causes and results on product performance and/or organisational efficiency?*

The answer focuses on the second and third challenge. Four key symptoms were identified; *waste*, *complexity debt*, *deteriorating quality* and *lost market shares.* Variety-induced complexity drives waste when time is spent on none-value adding activities such as procuring information, administration and verification. A complexity debt is built up if not managed properly, as complexity can be self-reinforcing. It also increases the risk of poor quality and without the appropriate support, innovation is inhibited. The firm might end up offering the wrong commercial variety if flexibility is not leveraged, resulting in lost market shares. Ultimately, profit margins deteriorate as internal costs may arise. The root-causes to these challenges are divided into three groups: *1)* Strategies: *2)* Information and structures; And *3)* modelling and tools.

Strategies

In the MC paradigm internal variety exists not only at one point in time — for a product model. It spans both in time and across the product architecture. Time is thus a critical factor as new

product variants, functionality and technology artefacts are incorporated into the product architecture over time. Making design decisions with the accumulated variety in mind proves to be a challenge in practice. In project-based pd — where a large project organisation collaborates to realise one product model at a time — information and priorities are often biased towards the first model. Significant focus in terms of resources, control and governance is spent in such vehicle projects. Decisions that favour individual vehicle models thus have authority, often to the disfavour of architectural strategies. In a project, the organisation works with a similar cadence where flexibility and resources exist to resolve issues late in development. When this results in a failure to embrace flexibility and commonality, *continuously evolving* the products can become impossible in practice.

Some parts of the organisation only see the upsides of variety, others see the downsides and some are indifferent as they only see to their own area of responsibility. When complexity is viewed in silos, efficient complexity management is inhibited. Project-based pd seems to promote complexity management per product model based on the complexity of previous models as a baseline. A key challenge is breaking down silos and holistically incorporating variety-induced complexity in decision-making. Strictly controlled vehicle projects, one model at a time, seems to work in strong disfavour of holistic platform-based development and complexity management.

Information and Structures
Architectural decision-making requires a focus on the interplay between commercial variety, internal variety and complexity. Information is however scant in early pd phases, when many decisions are made. Decisions are instead typically based on heuristics and experience. This is partly due to the fact that information might be isolated in organisations, product functions or projects. And partly due to the inefficient ways to generate, structure and transfer knowledge related to why a design looks the way it does, the product contexts where it is used, the customer utility it fulfils and its impact on complexity costs.

Problems occur when the project-based organisation is reflected in product structures and information flows. This can lead to an inability to assess variety across the product platform. A traditional product structure, with an upper level — the product specification — and a lower level — a generic BOM — is physical in nature as its main purpose is to support *assemble-to-order* (the studied company uses a traditional product structure). The *result* of decisions are documented rather than *why* the design looks the way it does. Knowledge about variety is also more related to attributes notable by the customer. Thus, the product structure only represents how a vehicle family *will* vary at a given point in time. Not how the product architecture *can* vary to support new needs and requirements. The possible solution space is not explicitly documented . As a result component re-use is enabled rather than re-use of design knowledge.

When variety is planned for the *vehicle project*, it is difficult to derive the systems variety originates from. Complexity and modularization are arguably better assessed and managed on system level. Costs are however often derived from components. Efficiently moving between component-level, system-level and product-level is thus required to relate costs to customer utility. If variety is defined for individual sub-systems, one can work pro-actively with complexity management. Many areas in the studied organisation, from marketing to manufacturing expressed a need to be able to plan for variety early and on a more abstract level than the component-level.

Tools and Modelling
This study has identified issues when *development* is decoupled from *documentation*. The product structure defines the complete vehicle in terms of parts, assemblies and features. Other views

and levels of abstraction are however often required during development. This discrepancy, if not mitigated, results in different terminology, lack of transparency and lack of traceability between development, integration and documentation.

Gedell (2011) argues that, in order to re-use a design, the design engineer needs more information about it. E.g. what function it satisfies and why it looks as it does. If one better understands a sub-system and its interfaces it is easier to adhere to them. A non-modular and inflexible product structure may result in longer lead times even if the architecture itself is flexible. The studied company has for example an inability to simulate and validate changes to the variety offer, resulting in significant administration to modify the product.

An important component of the product structure is the configuration rules that define buildable variants. With product configuration being an automated activity following a customer order, the set of configuration rules must be complete before the product can be released. This is typically done during the industrialisation phase. Rule authoring consists of transforming informal knowledge to formal knowledge, a specific type of *knowledge acquisition process* (Tidstam, 2014). This process, typically based on natural language and informal tools, is complex and known to be error prone. The product configuration system is thus an expert system, with the formal knowledge about product variety being inaccessible for most design engineers.

A key concept of product platforms as a mean to manage complexity is knowledge re-use, not only in the sense of already designed parts but also in terms of ideas, concepts and technology. The framework presented in Chapter 5 provides a way to model platforms, and using this knowledge to integrate variety-induced complexity in decision-making. The framework is proposed in order to answer the third research question:

> ***RQ 3)*** *How can design engineers and related functions be supported in better managing these root-causes?*

The framework represents an answer, although it is not the sole solution, to this question as it provides a variety-centric way of modelling configurable product architectures in order to generate and visualise knowledge. Development of complex products is to a large extent based on practical reasoning, rule of thumb and experience. This seems to be very much because companies lack the competence, resources or tools to establish knowledge-based practices, resulting in lack of knowledge: Is it possible to implement the change? Can we afford to develop another variant? Can we save enough money eliminating this variant? Which variants does this change affect and which do we need to verify?

The results down the road are extremely difficult to foresee for the individual design engineer, making development difficult when balancing on the knives edge between complexity costs and market shares. Modifying the product architecture in this context is likely to be haphazard and revolution rather than evolution will ultimately prevail, imposing huge development costs.

The framework aims at closing some of the knowledge gaps, making the development of configurable products more systematic and predictable. One strength with CPM is that it may be used with rather high reliability already in conceptual phases when information about FRs and DSs exist but the physical structure is incomplete. With an EF-M tree, a high-level description of the system may be obtained when details are scarce in order to assess for example how coupled alternative solutions are. Abstract models can thus support early architectural decision-making.

The EF-M model also serves as a knowledge base, describing among other things how DSs interact. Analysis of *context models* rather than *content models* allows targeting complexity in

more dimensions than simply assessing design dependencies between physical components. According to the principles of axiomatic design, a design with few functional couplings is preferred in terms of complexity. Moreover, centralised product architectures are preferred in terms of topological complexity. Complexity can thus be avoided, in particular as product architectures today often are prepared by studying the components and structures of existing products (Raudberget et al., 2015).

The CC approach provides means to integrate and encapsulate variability. The solution space of each sub-system is therefore integrated in the knowledge base. The framework's affinity with the concept of product modularity is clear, although with the CC approach it adds another characteristic, namely the number of variants. Different solution spaces can for instance be compared. Let us again consider exterior RVMs. The rear FoV is a function of the shape and dimensions of the mirror glass, its aim, orientation and placement. Different vehicle geometries, legislations and functionality result in a rather complex network of influencing factors. In particular as the number of mirror glass variants per vehicle model may exceed 20, and as each variant may contribute differently to system behaviour. The framework, including assessment of change propagation, allows navigating this network of influencing factors.

Local regulations require different glass shapes; planar, convex and aspheric. For a non-planar glass, the rearwards FoV is also a function of the glass' convexity. This knowledge is incorporated into the likelihood and impact values. As the size of the glass is constrained by visual and ergonomic requirements (concealed frontward FoV and mirror sampling) there is a trade-off between rear FoV and mirror size. This is depicted as a trade-off curve in Figure 7.2. As the figure shows, the trade-off curve for the non-planar variants is shifted to the right since they provide the same FoV with a smaller size. Due to commonality reasons however, all glasses have the same size. The planar glass thus regulates the size and the two variants V_1 and V'_2 are employed rather than the, from a trade-off perspective, optimal set V_1 and V_2. Hence, there is often a trade-off between commonality and performance for the design.

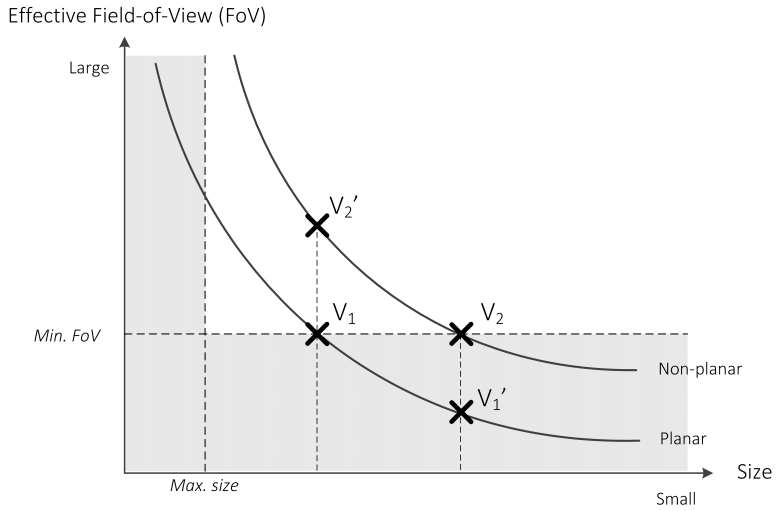


Figure 7.2: Trade-off curves for RVM size and effective rearward FOV.

As Figure 7.2 shows, the non-planar variants have a larger safety margin against changes that risk affecting the FoV negatively. Two conclusions can be drawn from this insight: *(1)* Any such change needs to be verified at least for the planar glass variant; And *(2)* variants with a lower safety margin (with a lower design- and development process efficiency) regulates the flexibility of the design.

7.2 Proof of Research Hypothesis

The following research hypothesis was stated in order to propel the work with the framework:

> *Analysing change propagation in the functional domain enables design engineers to assess variety-induced complexity and to design flexible and robust products at reduced complexity costs.*

It is shown that modelling product architectures in terms of their functional and organ structures (FRs and DSs) is a promising way of presenting configuration-related knowledge. Such knowledge, along with analytical data obtained from CPM, show great potential in assessing variety-induced complexity and incorporating this into product architecture- and portfolio decision-making.

The 'goodness' of an architecture may be assessed in terms of multiple criteria using this framework. The applicability however depends on the model's level of abstraction. The higher the degree of decomposition, the more details. For each additional level, the CPM may be iterated, achieving more precise estimates. This allows making decisions affecting variety-induced complexity early, and rather using the detailed model to validate these decisions. The proposed framework thus has the potential to support all stages of pd, from conceptual design, to system-level design and detailed design. Additional work is however required to investigate how this knowledge can be turned into concrete design decision for the hypothesis to be fully proven.

7.3 Contributions and Future Work

The thesis goal — *investigate and describe challenges in automotive pd resulting from variety-induced complexity and to demonstrate support thereof* — has been fulfilled. The identified challenges pose a real threat to the efficiency and efficacy of the studied company. The research environment is also likely to become more complex in general in a near future. At the time this study was conducted, many of the trends driving overall complexity were only in early stages, and the automotive industry will continue to transform. Complexity combined with demands on efficiency strengthen the challenges related to development of product architectures that capitalise on commonality and differentiation. This is believed to be a topic relevant for continued focus and several company goals, for instance the *continuous evolution of product architectures*, validate this hypothesis. The findings in this thesis can help the company focus its work on managing variety-induced complexity in the future.

The challenges presented have been discussed, detailed and exemplified extensively. The findings can therefore serve as an umbrella for continued work on the phenomena of variety-induced complexity. The proposed framework enables assessment of the topological complexity for each architectural configuration. How the insights can be integrated in decision-making however needs continued work. In particularly related to early development of larger architectural systems. The framework also lacks means quantifying the aggregated complexity of generic systems, and the contribution of each configuration. Future work should include more holistic complexity metrics.

Commercial variety — the features that the customer is willing to pay for — should be integrated in the framework in order to relate internal variety to customer utility. A common approach to limit internal variety is for example to build product variants with components capable of fulfilling a function, but with the functionality disabled for the customer (via a flag in the software). Both dimensions are needed to express that a customer can choose for example a RVM without possibility to store the glass orientation while receiving a design solution capable of doing it.

Finally, the proposed framework is limited to architectural modelling in the function-means domain. A physical design often aspires to embody as much variety as possible to maximise commonality. Future work could include integrating the physical domain, to make component commonality explicit.

7.4 Evaluation of Thesis

As several conclusions have been drawn from the findings, it is important to discuss their validity. This section provides a brief discussion about the validity of the thesis.

7.4.1 Validity of Empirical Data

The studied company served as an excellent source of data — offering significant product variety while being sensitive to industry trends. In addition, it has seen a paradigm shift in terms of how products are developed and offered. It shall however be noted that no benchmarks against competitors or similar companies in other industries have been made. Instead, published literature had to confirm the identified challenges and their generalisability. The findings in fact seem much aligned with the general conceptions within the research community.

The conclusion that variety is triggered by both internal and external factors is supported by the study conducted by Götzfried (2013) in which a fifty-fifty split between externally and internally driven variants were reported for a premium car manufacturer. That manufacturer went through a similar paradigm shift where it reportedly fell into the typical 'pitfall' of product variety; deteriorating return on investments fuelled by increasing variety in particular aiming at niche markets. The result was an overload of low-volume variants triggering a costs disproportional to the sales increase. Several similarities exist, for instance sub-optimal complexity- and cost reductions and an increased usage of option bundles that despite the intentions resulted in an explosion of complexity costs. ElMaraghy et al. (2013) state that variety-induced complexity arise *'due to the increasing number of variants and their features, more multi-disciplinary complex components and modules, lack of process streamlining and insufficient use of adaptable Computer Aided Design (CAD) models and planning tools to benefit from similarity and commonality'.* This validates the conclusions of lack of appropriate modelling methods and tools.

The deep involvement and theoretical study of the phenomena risk over-interpretation of the empirical data. Due to the complex nature of the research topic, a large portion of the empirical result is based on interpretations and extrapolation of anecdotal evidence. A large and cross-functional base of interviewees was selected to mitigate this risk. The empirical results clearly saturated at the end of the interviews. The sample size is thus considered satisfactory. A shortcoming is however a lack of quantitative data to confirm the qualitative data. The interview sample consisted of, among other roles, design engineers working with fuel lines and RVMs. To enhance the generalisability, more design areas could have been included. In particular those with a more central role in the product architecture (for example engines).

7.4.2 Validity of the Case Study

The proposed framework rests on a solid theoretical base. Although the RVM case study served its purpose; to demonstrate the framework, a re-design task could have been used to better prove its usability. As it were, an already finished design was modelled and assessed bottom-up. This approach resulted in a very large EF-M model as it allowed a high level of detail. At that level, the CCM proved to be rather tedious. The framework does not replace analysis in the physical domain. Rather it adds the dimension of abstract modelling, thus benefiting from modelling

larger systems. To prove the framework further, a more complex system could thus have been selected (e.g. an engine, a climate system or a brake system), modelled at a lower level of detail.

The RVM design included some software solutions mapped to the same FR as their related hardware solutions. For example, the FR to signal about a vehicle in the blind spot were broken down into a LED light and a software to send a signal to light it up based on input from a radar in the rear of the vehicle. Since the radar was excluded from the case study, the software only had an information interface to the LED light. Instead the cables from the ECU running the software were given significant importance. Cables have many other dependencies as they spread through the vehicle. A large portion of the risks of change propagation can thus likely be traced to cables.

To improve the results (and to make the model less administrative), more focus should have been spent on how to cluster software solutions and on the information interfaces software-to-software and software-to-hardware. As it were, dependencies between software solutions deployed together on an ECU and between the ECU and the sensors and actuators to which it sent and received signals over various networks were the predominant focus. The software architecture has a different set of dependencies if one look at how it is designed versus how software is deployed. These aspects should receive more focus in future studies.

7.5 Ethical Considerations

Assemble-to-order manufacturing is, from an ethical and sustainability perspective, positive as every manufactured vehicle is derived from a customer order. Few vehicle variants are thus produced to stock, minimising potential waste. On the other hand, variety is inevitable. With poor predictions, low volume parts and assemblies are pushed up the value chain. Again, this emphasise the importance of understanding variety and how the product architecture can be designed to enhance commonality and flexibility.

Another benefit with assemble-to-order is that it gives the consumer more power to make ethical decisions. If the environmental impact, e.g. CO_2 emissions per vehicle configuration, is transparent during the sales interaction, the customers are given the power to more directly steer demand. This ultimately gives manufacturers incentives to develop more environmentally friendly products. Increased abilities to model and understand a certain decision's effects on product behaviour might propel this evolution.

Concepts such as re-use and architectural flexibility will play an important role in circular economies. If modularity and flexibility are leveraged, sub-systems can live longer with less resources spent on massive projects with the sole aim of integrating new hardware. Vehicles may instead be designed to live longer by enabling functionality on demand and upgradeability. Re-configuration of existing products could thus become an important concept, a candidate for future studies.

Better tools for modelling and assessing variety also play a key role in enabling virtual verification. Being able to virtually test and verify a high number of variants, along with better processes to select the ones to test, not only reduce the need for costly physical prototypes. It is also necessary to verify the safety-critical autonomous drive and advanced driver assist systems on the rise. This is crucial to ensure the safety of future citizens.

8

Conclusion

This chapter provides the reader with a brief conclusion of the results and their applicability. In addition, the contribution of the thesis is proposed as well as suggestions for future work.

Product variety keeps its foothold and may seriously endanger the success of firms if complexity outgrows sales. The mass customising firm needs to be able to offer commercial variety valued by its customers while managing complexity and lean structures. At the same time, market volatility and inertia caused by complexity require companies to better leverage change as an enabler for innovation and responsiveness to market needs.

Customer utility increases with increasing variety to a point of saturation where mass customisation turns into mass confusion (Huffman & Kahn, 1998). Development effort is also shown to grow exponentially with increasing complexity (Sinha & de Weck, 2016). This thesis contributes with a comprehensive survey — based on empirical data from a Swedish automotive company — of variety-induced complexity and how it seems to multiply the rate of which development effort grows. For firms developing complex products subjected to variety, understanding the interplay between complexity, customer utility and development effort is key in order to offer the *right* variety. This expands to maximising the value obtained with a limited complexity budget, which imply two main challenges; the ability to develop robust and stable product architectures with low internal variety and a minimal complexity debt, and minimising time spent on wasteful activities such as administration, knowledge acquisition, test and verification.

The studied company has fallen into one of many pitfalls of variety; approaches to manage complexity are reactive rather than proactive. The approaches are in addition often influenced by legacy, experience and rule of thumb rather than on facts. This study has described how mature systems, combined with re-design in order to fulfil new requirements or functionality, poorly governed by strategies, corporate culture and modelling capabilities result in quality issues, late changes, re-design and increased variety. Firms must work strategically with variety-induced complexity. In particular, the findings in this study establish the importance of holistically integrating variety and re-use in the information flows and tools that support design engineers and system architects in their daily work.

The presented framework addresses the issue of effectively addressing variety-induced complexity during development of product architectures. In particular, inabilities to efficiently model, visualise and assess the architecture's solution space from a variety perspective are addressed. The framework is based on architecture modelling in the function-means domain, combined with a systems-oriented product structure and CPM. Serving as a knowledge base, this has the ultimate aim of data driven decision-making. The framework consists of three stages; *model the generic system architecture*, *build an architectural knowledge base* and *assess the architecture*. When conducted in a concurrent manner, development activities are supported.

The thesis contributes with a detailed description of the framework, supported by a simple

illustrating case from the automotive industry. The product architecture is modelled in CCM which builds on EF-M modelling. This methodology specifies how design solutions are allocated to functions and how they interact. The CC approach allows encapsulating parts of the product which vary into *configurable components (CCs)*. The knowledge base is completed by specifying the design- and functional bandwidths of each CC. Variability is thus visually represented as optional FRs or alternative DSs in the EF-M tree. This systems-oriented product structure enables analysis in early development phases when information is rare. Change propagation analysis is used to assess the complexity of different system configurations based on their number of couplings. As a more coupled architecture is more complex, the analysis can provide important knowledge about the 'goodness' of a design. In particular, re-design, new variants or alternative solutions can be evaluated during early development phases based on the required development effort or based on the resulting complexity and robustness of the product architecture.

Through the case study we showed that the framework provides ways of visually representing variability, formalising knowledge about variability and assessing variety-induced complexity. For instance, generating knowledge about which configurations that posses the smallest safety margins against certain requirements, thus exhibiting the greatest risks during re-design. It is shown that a large number of variants can be efficiently modelled, assessed and compared. The knowledge gained should thus be able to spark discussions related to both product portfolio- and product architecture decision-making, with the potential of balancing the selected configuration space more in favour of low complexity costs.

8.1 Future Work

Future work will enhance the usefulness and validity of the framework. Steps to show how the it can be applied on larger architectures and how it can be integrated in actual decision-making need to be taken. For instance, applying the framework on a re-design task or on a task modifying the configuration space of the product architecture. The case study presents interesting challenges related to software and mechatronic platforms, in particular how to partition the system into CCs. This is a suitable candidate for further work.

Moreover, the framework should be expanded to cover more dimensions of variability and complexity. One such area is the development of holistic complexity metrics in order to quantify the costs of complexity. Such metrics could quantify the combined effects of structural- and variety-induced complexity for the generic system as well as for each individual variant. In order the integrate such metrics in decision-making, the value dimension should be added to the framework. That is, modelling the relationships between internal and commercial variety. Complexity would thus be possible to relate to customer utility.

References

Aigbedo, H. (2007). An assessment of the effect of mass customization on suppliers' inventory levels in a JIT supply chain. *European Journal of Operational Research*, *181*(2), 704–715.

Alford, D., Sackett, P., & Nelder, G. (2000). Mass customisation - an automotive perspective. *Int. Journal of Production Economics*, *65*(1), 99–110.

Anderson, D. (1997). *Agile product development for mass customization: How to develop and deliver products for mass customization, niche markets, JIT, build-to-order, and flexible manufacturing.* Irwin Professional Pub.

Andreasen, M. M. (1992). Designing on a designer's workbench (DWB). *Proceedings of the 9th WDK Workshop.*

Asadia, N., Jackson, M., & Fundin, A. (2016). Drivers of complexity in a flexible assembly system: A case study. *Procedia CIRP*, *41*(1), 189–194.

Baldwin, C., & Clark, K. (1997). Managing in an age of modularity. *Harvard Business Review*, *75*(5), 84–93.

Baldwin, C., & Clark, K. (2000). *Design rules, vol. 1: The power of modularity* (4th ed.). The MIT Press.

Bashir, H. A., & Thomson, V. (2001). Models for estimating design effort and time. *Design Studies*, *22*, 141–155.

Belt, A., von Hagel, K., & Ferguson, S. (2015). Navigating redesign and market desirability implications when considering increased product variety. *Journal of Engineering Design*, *26*(7-9), 236–258.

Ben-Arieh, D., Easton, T., & Choubey, A. M. (2009). Solving the multiple platforms configuration problem. *International Journal of Production Research*, *47*(7), 1969–1988.

Ben-Yehudaa, T., Zonnenshaina, A., & Katza, R. (2015). Evaluating complicatedness in mechanical design. *Procedia CIRP*, *36*, 41–47.

Berglund, F., & Claesson, A. (2005). Utilizing the concept of design bandwidth to achieve product platform effectiveness, In *International conference on engineering design, iced05*, Melbourne, Australia.

Bhise, V. D. (2013). *Designing complex products with systems engineering processes and techniques* (1st ed.). CRC Press.

Blecker, T., & Abdelkafi, N. (2006). Controlling variety-induced complexity in mass customisation: A key metrics-based approach. *International Journal of Mass Customisation*, *1*(2), 272–298.

Blecker, T., Abdelkafi, N., Kaluza, B., & Kreutzler, G. (2004). A framework for understanding the interdependencies between mass customization and complexity. *Proceedings of the 2nd International Conference on Business Economics, Management and Marketing.*

Blessing, L. T. M., & Chakrabarti, A. (2009). *DRM, a design research methodology* (1st ed.). Springer-Verlag.

Brière-Côté, A., Rivest, L., & Desrochers, A. (2010). Adaptive generic product structure modelling for design reuse in engineer-to-order products. *Computers in Industry*, *61*(1), 53–65.

Bryman, A., & Bell, E. (2011). *Business research methods* (3rd ed.). Oxford University Press.

Campbell, D. (1988). Task complexity: A review and analysis. *The Academy of Management Review*, *13*(1), 40–52.

Chen, C., & Wang, L. (2008). Product platform design through clustering analysis and information theoretical approach. *International Journal of Production Research*, *46*(15), 4259–4284.

Claesson, A. (2006). *A configurable component framework supporting platform-based product development* (Doctoral dissertation). Chalmers University of Technology.

Clarkson, J. P., Simons, C., & Eckert, C. (2004). Predicting change propagation in complex design. *Journal of Mechanical Design*, *126*(5), 788–797.

Collier, W. (1999). *A common specification for systems-based product modelling*. DH Brown Associates.

Cova, B. (1996). The postmodern explained to managers: Implications for marketing. *Business Horizons*, *39*(6), 15–23.

Crawley, E., de Weck, O., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., & Whitney, D. (2004). The influence of architecture in engineering systems.

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

Cutherell, D. (1988). Product architecture. In M. Rosenau, G. Griffin, G. Castellion, & N. Anschuetz (Eds.), *The PDMA handbook of new product development*. JohnWiley & Sons.

Du, X., Jiao, J., & Tseng, M. M. (2001). Architecture of product family: Fundamentals and methodology. *Procedia CIRP*, *9*(4), 309–325.

ElMaraghy, H., Schuh, G., ElMaraghy, W., Piller, F., Schonsleben, P., Tseng, M., & Bernard, A. (2013). Product variety management. *CIRP Annals - Manufacturing Technology*, *62*(2), 629–652.

ElMaraghy, H. A., & ElMaraghy, W. H. (2013). Variety, complexity and value creation. *Proceedings of the 5th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV)*.

Ericsson, A., & Erixon, G. (2000). Controlling design variants: Modular product platforms. *American Society of Mechanical Engineers*, *122*(5), 83–91.

Ferguson, S. M., Olwnik, A. T., & Cormier, P. (2014). A review of mass customization across marketing, engineering and distribution domains toward development of a process framework. *Research in Engineering Design*, *25*(1), 11–30.

Fisher, M. L., & Ittner, C. D. (1999). The impact of product variety on automobile assembly operations: Empirical evidence and simulation. *Management Science*, *45*(6), 771–786.

Fisher, M. L., Jain, A., & Macduffie, J. P. (1995). Strategies for product variety: Lessons from the auto industry. In E. H. Bowman & B. M. Kogut (Eds.), *Redesigning the firm* (pp. 116–154.). New York, USA, Oxford University Press.

Fricke, E., Gebhard, B., Negele, H., & Igenbergs, E. (2000). Coping with changes: Causes, findings, and strategies. *Syst Eng*, *3*(4), 169–179.

Gedell, S. (2011). *Efficient means for platform-based development emphasizing integrated, information-rich system models* (Doctoral dissertation). Chalmers University of Technology.

Gedell, S., & Johannesson, H. (2013). Design rationale and system description aspects in product platform design - focusing reuse in the design lifecycle phase. *Concurrent Engineering*, *21*(1), 39–53.

Götzfried, M. (2013). *Managing complexity induced by product variety in manufacturing companies complexity evaluation and integration in decision-making* (Doctoral dissertation). University of St. Gallen.

Griffin, A. (1997). Modeling and measuring product development cycle time across industries. *Journal of Engineering and Technology Management*, *14*(1), 1–24.

Gustafsson, G., Raudberget, D., & Ström, M. (2016). Understanding product family for mass customization by developing commonality indices. *Procedia CIRP*, *50*, 204–209.

Haag, A. (1998). Sales configuration in business processes. *Intelligent systems and their applications*, *13*(4), 78–85.

Hamraz, B., Caldwell, N. H. M., & Clarkson, J. P. (2013a). A matrix-calculation-based algorithm for numerical change propagation analysis. *IEEE Transactions on Engineering Management*, *60*(1), 186–198.

Hamraz, B., Caldwell, N. H. M., & Clarkson, P. J. (2013b). A holistic categorization framework for litterature on engineering change management. *Syst Eng*, *16*(4), 473–505.

Hamraz, B., Caldwell, N. H. M., Ridgman, T. W., & Clarkson, J. P. (2015). FBS linkage ontology and technique to support engineering change management. *Research in Engineering Design*, *26*(1), 3–35.

Hamraz, B., Hisarciklilar, O., Rahmani, K., Wynn, D. C., Thomson, V., & Clarkson, J. P. (2013). Change prediction using interface data. *Concurrent Engineering*, *21*(2), 141–154.

Hu, S. J., Zhu, X., Wang, H., & Koren, Y. (2008). Product variety and manufacturing complexity in assembly systems and supply chains. *CIRP Annals - Manufacturing Technology*, *57*(1), 45–48.

Huang, G. Q., & Mak, K. L. (1999). Current practises of engineering change management in uk manufacturing industries. *International Journal of Operations & Production Management*, *19*(1), 21–37.

Hubka, V., & Eder, W. E. (1988). *The theory of technical systems: A total concept theory for engineering design* (1st ed.). Springer Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-52121-8

Huffman, C., & Kahn, B. (1998). Variety for sale: Mass customization or mass confusion. *Journal of Retailing*, *74*(4), 491–513.

INCOSE. (2017). What is systems engineering? [Accessed: 2017-10-29]. www . incose . org / AboutSE / WhatIsSE

Jacobs, M., & Swink, M. (2011). Product portfolio architectural complexity and operational performance: Incorporating the roles of learning and fixed assets. *Journal of Operations Management*, *29*(7), 677–691.

Jarratt, T. A. W., Eckert, C. M., Caldwell, N. H. M., & Clarkson, P. J. (2011). Engineering change: An overview and perspective on the literature. *Research in Engineering Design*, *22*(2), 103–124.

Jiao, J., Simpson, T. W., & Siddique, Z. (2007). Product family design and platform-based product development: A state-of-the-art review. *Journal of Intelligent Manufacturing*, *18*(1), 5–29.

Jiao, J., & Tseng, M. M. (1999). A methodology of developing product family architecture for mass customization. *Journal of Intelligent Manufacturing*, *10*(1), 3–20.

Jiao, J., & Tseng, M. M. (2000). Understanding product family for mass customization by developing commonality indices. *Journal of Engineering Design*, *11*(3), 225–243.

Kim, K., & Chhajed, D. (2001). An experimental investigation of valuation change due to commonality in vertical product line extension. *Journal of Product Innovation Management*, *18*(4), 219–230.

Kübler, A., Zengler, C., & Küchlin, W. (2010). Model counting in product configuration. *Proceedings First International Workshop on Logics for Component Configuration (LoCoCo)*, 44–53.

Landahl, J. (2016). *Development of interdisciplinary platforms using system objects* (Licentiate Thesis). Chalmers University of Technology.

Landahl, J., Bergsjö, D., & Johannesson, H. (2014). Future alternatives for automotive configuration management. *Procedia Computer Science*, *28*, 103–110.

Landahl, J., Lewandowski, C., Johannesson, H., Söderberg, R., Wärmefjord, K., Carlsson, J. S., Kressin, J., Isaksson, O., & Valhagen, J. (2016). Using product and manufacturing system platforms to generate producible product variants. *Procedia CIRP*, *44*(1), 61–66.

Lewandowski, C., Jiao, J. R., & Johannesson, A. (2015). A two-stage model of adaptable product platform for engineering-to-order configuration design. *Journal of Engineering Design*, *26*(7-9), 220–235.

Lewandowski, C., Raudberget, D., & Johannesson, H. (2014). Set-based concurrent engineering for early phases in platform development, In *The 21st ispe international conference on concurrent engineering - ce2014*, Beijing, China,

MacDuffie, J. P., Sethuraman, K., & Fisher, M. L. (1996). Product variety and manufacturing performance: Evidence from the international automotive assembly plant study. *Management Science*, *42*(3), 350–369.

Maier, A. M., & Langer, S. (2011). *Engineering change management report 2011: Survey results on causes and effects, current practice, problems, and strategies in denmark* (tech. rep.). Technical University of Denmark.

Martin, M. V., & Ishii, K. (2002). Design for variety: Developing standardized and modularized product platform architecture. *Research in Engineering Design*, *13*(4), 213–235.

Mather, H. (1982). *Bills of materials, recipes and formulations*. Wright Publishing Company Inc.

McGrath, M. (1995). *Product strategy for high-technology companies* (2nd ed.). McGraw-Hill Education.

McKay, A., Eres, F., & Bloor, M. S. (1996). Relating product definition and product variety. *Research in Engineering Design: Theory, Applications, and Concurrent Engineering*, *8*(2), 63–80.

McQuarrie, E. F. (2016). *The market research toolbox: A concise guide for beginners* (4th ed.). Sage Publications Inc.

Mesihovic, S. (2004). *On the development of sales-delivery process of configurable products* (Licentiate Thesis). Chalmers University of Technology.

Meyer, M., & Lehnerd, A. P. (1997). *The power of product platform – building value and cost leadship*. The Free Press.

Mittal, S., & Frayman, F. (1989). Towards a generic model of configuration tasks. *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI89)*, 1395–1401.

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

Newcomb, P. J., Bras, B., & Rosen, D. W. (1996). Implications of modularity on product design for the life cycle, In *Asme design engineering technical conferences, detc96/dtm-1516*, Irvine, CA.

Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decompositions, In *Asme design theory and methodology conference*, Minneapolis, MN, USA.

Pine, B. (1999). *Mass customization: The new frontier in business competition* (1st ed.). Harvard Business Review Press.

Qu, T., Bin, S., George, Q., Huang, Q., & Yang, H. D. (2011). Two stage product platform development for mass customisation. *Int. Journal of Production Research*, *49*(8), 2197–2219.

Raudberget, D., Lewandowski, C., Isaksson, O., Kipourus, T., Johannesson, H., & Clarkson, J. (2015). Modelling and assessing platform architectures in pre-embodiment phases through set-based evaluation and change propagation. *Journal of Aerospace Operations*, *3*(3-4), 203–221.

Robertson, D., & Ulrich, K. (1998). Planning for product platforms. *Sloan Management Review*, *39*(4), 19–31.

Saeed, B., & Young, D. (1998). *Managing the hidden costs of complexity*. Boston Consulting Group.

Salvador, F., de Holan, P. M., & Piller, F. (2009). Cracking the code of mass customization. *MIT Sloan Management Review*, *50*(3), 71–78.

Salvador, F., & Forza, C. (2004). Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities. *Int. J. Production Economics*, *91*(3), 273–291.

Sanderson, S. W., & Uzumeri, M. (1997). *Managing product families*. McGraw-Hill Education.

Sawhney, M. S. (1998). Leveraged high-variety strategies: From portfolio thinking to platform thinking. *Journal of the Academy of Marketing Science*, *26*(1), 54–61.

Schachinger, P., & Johannesson, H. L. (2000). Computer modelling of design specifications. *Journal of Engineering Design*, *11*(4), 317–329.

Schuh & Company. (2015). Lean innovation (part 3). *Complexity Management Journal*, *4*(1), 1–26.

Scupin, R. (1997). The KJ method: A technique for analyzing data derived from japanese ethnology. *Human Organization*, *56*(2), 233–237.

Simpson, T. W. (2004). Product platform design and customization: Status and promise. *AI EDAM*, *18*(1), 3–20.

Simpson, T. W., Maier, J. R. A., & Mistree, F. (2001). Product platform design: Method and application. *Research in Engineering Design*, *13*(1), 2–22.

Sinha, K., & de Weck, O. L. (2013). A network-based structural complexity metric for engineered complex systems, In *2013 ieee international systems conference (syscon)*, Orlando, FL.

Sinha, K., & de Weck, O. L. (2016). Empirical validation of structural complexity metric and complexity management for engineering systems. *Journal of Intelligent Manufacturing*, *18*(1), 5–29.

Soininen, T., Tiihonen, J., Männist, T., & Sulonen, R. (1998). Towards a general ontology of configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *12*(4), 357–372.

Sosa, M. E., Eppinger, D. D., & Rowles, C. M. (2003). Identifying modular and integrative systems and their impact on design team interactions. *ASME J. Mech. Des.*, *125*(2), 240–252.

Sosa, M. E., Eppinger, D. D., & Rowles, C. M. (2007). A network approach to define modularity of components in complex products. *Journal of Mechanical Design*, *129*(11), 1118–1129.

Stark, J. (2015). *Product lifecycle management (volume 1): 21st century paradigm for product realisation* (4th ed.). Springer international Publishing. https://doi.org/10.1007/978-3-030-28864-8

Steward, D. (1981). The design structure: A method for managing the design of complex systems. *IEEE Trans. Eng. Management*, *28*(3), 71–74.

Suh, N. P. (1990). *The principles of design* (1st ed.). Oxford University Press.

Suh, N. P. (1991). A theory of complexity, periodicity and the design axioms. *Research in Engineering Design*, *11*(2), 116–131.

Suh, N. P. (2001). *Axiomatic design: Advances and applications*. Oxford University Press.

Suh, N. P. (2005). Complexity in engineering. *CIRP Annals - Manufacturing Technology*, *54*(2), 46–63.

Suh, N. P. (2007). Ergonomics, axiomatic design and complexity theory. *Theoretical Issues in Ergonomics Science*, *8*(2), 101–121.

Sundgren, N. (1999). Introducing interface management in product family development. *Journal of Production Innovation Management*, *16*(1), 40–51.

Svendsen, K. H., & Hansen, T. H. (1993). Decomposition of mechanical systems and breakdown of specifications. *Proceedings of ICED*.

Svensson, D., & Malmqvist, J. (2001). Integration of requirement management and product data management systems. *Proceedings of the 2001 ASME Design Engineering Technical Conference and Computers and Information in Engineering Conference*, *1*, 199–208.

Tang, V., & Salminen, V. (2001). Towards a theory of complicatedness: Framework for complex systems analysis and design. *Proceedings of International Conference on Engineering Design*.

Terwiesch, C., & Loch, C. H. (1999). Managing the process of engineering change orders: The case of the climate control system in automobile development. *The Journal of Product Innovation Management*, *16*(2), 160–172.

Tidstam, A. (2014). *Efficient development of vehicle configuration rules* (Doctoral dissertation). Chalmers University of Technology.

Tidstam, A., & Malmqvist, J. (2011). Authoring and verification of vehicle configuration rules. *Proceedings of Product Lifecycle Management (PLM)*, 33–46.

Tidstam, A., & Malmqvist, J. (2015). A systematic process for developing product configuration rules. *Int. J. Product Lifecycle Management*, *8*(1), 46–64.

Tidstam, A., Malmqvist, J., Voronov, A., Åkesson, K., & Fabian, M. (2016). Formulating constraint satisfaction problems for the inspection of configuration rules. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *30*(3), 313–328.

Tiihonen, J., Soininen, T., Männistö, T., & Sulonen, R. (1996). State of the practice in product configuration - a survey of 10 cases in the finnish industry. In T. Tomiyama, M. Mäntylä, & S. Finger (Eds.). Chapman & Hall.

Tjalve, E. (1979). *Systematisk udformning af industriprodukter—værktøjer for konstruktøren [systematic design of industrial products—tools for the design engineer]*. Akademisk Forlag.

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

Ulrich, K. T., & Eppinger, S. D. (2012). *Product design and development* (5th ed.). McGraw-Hill/Irwin.

Usman, A., & Kidd, C. (2015). Configuration management maturation: An empirical investigation. *Journal of Engineering Manufacture*, *229*(2), 321–327.

van Veen, E. A. (1991). *Modelling product structures by generic bill-of-materials* (Doctoral dissertation). Eindhoven University of Technology.

van Wie, M., Rajan, P., Campbell, M. I., Stone, R. B., & Wood, K. L. (2003). Representing product architecture, In *Asme design engineering technical conferences*, Chicago, Illinois.

Wahl, A., & Claesson, A. (2010). Managing design change in configurable component based product platforms. *DS 61: Proceedings of NordDesign 2010, the 8th International NordDesign Conference*.

Wang, H., Ko, J., Zhu, X., & Hu, S. J. (2010). A complexity model for assembly supply chains and its application to configuration design. *Journal of Manufacturing Science and Engineering*, *132*(2), 1–12.

Wang, H., Zhu, X., Wang, H., Hub, S. J., Lina, Z., & Chena, G. (2011). Multi-objective optimization of product variety and manufacturing complexity in mixed-model assembly systems. *Journal of Manufacturing Systems*, *30*(1), 16–27.

Wortmann, J. C., Muntslag, D. R., & Timmermans, P. J. M. (1996). *Customer-driven manufacturing* (1st ed.). Chapman & Hall.

Wynn, D. C. (2007). *Model-based approaches to support process improvement in complex product development* (Doctoral dissertation). University of Cambridge.

Wynn, D. C., Caldwell, N. H. M., & Clarkson, P. J. (2010). Can change prediction help prioritise redesign work in future engineering systems?, In *Int. design conference*, Dubrovnik, Croatia.

Wynn, D. C., Nair, S. M. T., & Clarkson, J. P. (2009). The P3 platform: An approach and software system for developing diagrammatic model-based methods in design research, In *Int. conference on engineering design, iced'09*, Stanford, CA.

Wynn, D. C., Wyatt, D. F., Nair, S. M. T., & Clarkson, J. P. (2010). An introduction to the cambridge advanced modeller. *Proceedings of the 1st International Conference on Modelling and Management of Engineering Processes, MMEP 2010*.

Yu, T. L., Yassine, A. A., & Goldberg, D. E. (2003). A genetic algorithm for developing modular product architectures, In *Asme design engineering technical conferences*, Chicago, Illinois.

Zhu, X., Hu, S. J., Koren, Y., & Marin, S. P. (2008). Modeling of manufacturing complexity in mixed-model assembly lines. *Journal of Manufacturing Science and Engineering*, *130*(5), 1–10.

Zipkin, P. (2001). The limits of mass customization. *MIT Sloan Management Review*, *42*(3), 81–87.

Appendix A

Rear-view Mirror Model

This chapter provides further details about the architectural model of the RVM, as modelled in the CCM tool. The complete model in its full level of detail is not included in the main report.

A.1 Enhanced Function-means Tree

This section outlines the complete EF-M model for the RVM modelled in the CCM tool. CCs and design dependencies are excluded in the figures to make the model more readable.

Figure A.1 show the top-level DS decomposed into five FRs. The *mirror glass* (DS 1-1) is decomposed completely in Figure A.2. The *manoeuvrable mirror (glass actuator)* (DS 1-2) and the *side view camera* (DS 1-3) are decomposed in Figure A.3. The *structural link* (DS 1-4) is decomposed in Figures A.4, A.5 and A.6. The architectural branch *RVM house* (DS 2) is completely decomposed in Figure A.7. The *turn signal* (DS 4) is decomposed in Figure A.8 and the *wiring (cables)* (DS 5) in Figure A.9.

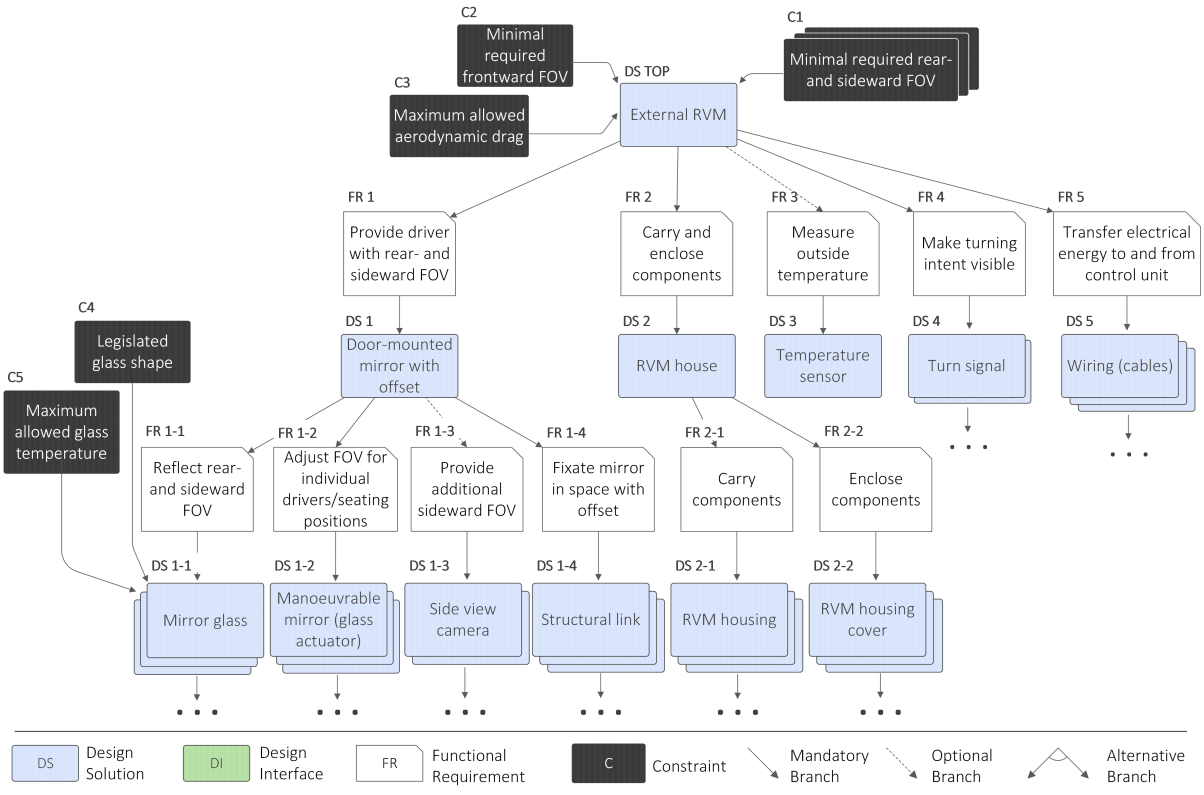


Figure A.1: Decomposition in two levels of the exterior RVM. Layered DSs indicate a variable design.

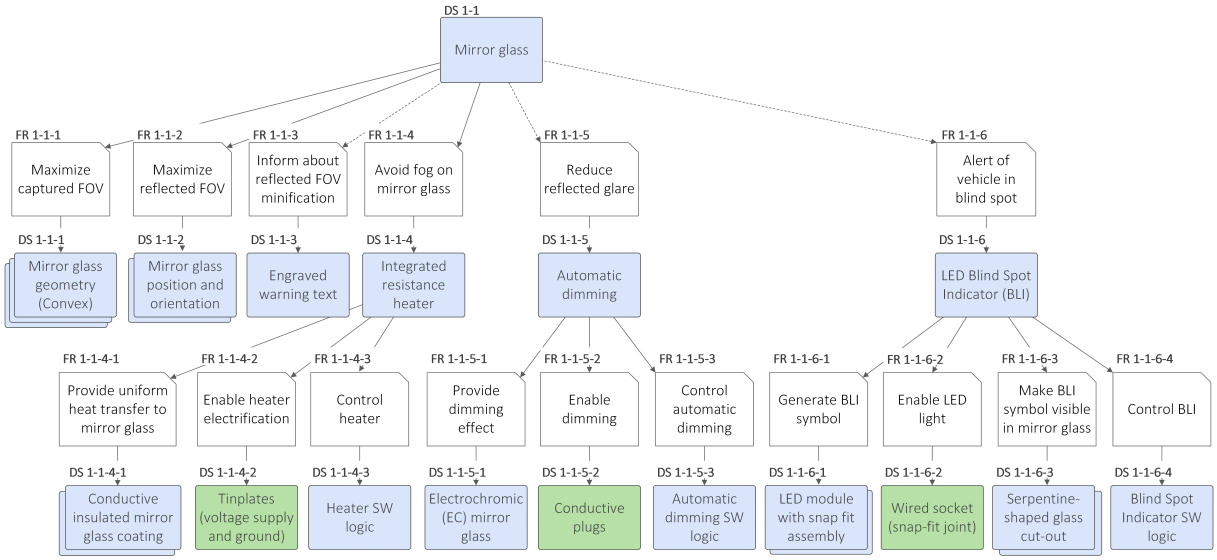


Figure A.2: Decomposition of the DS *mirror glass*.

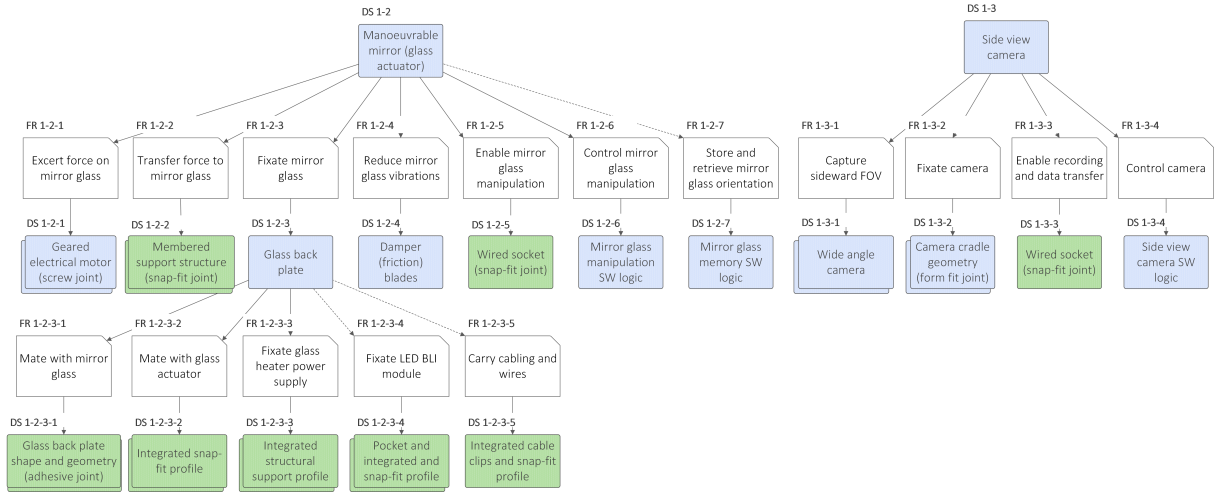


Figure A.3: Decomposition of the DSs *manoeuvrable mirror (glass actuator)* and *side view camera*.

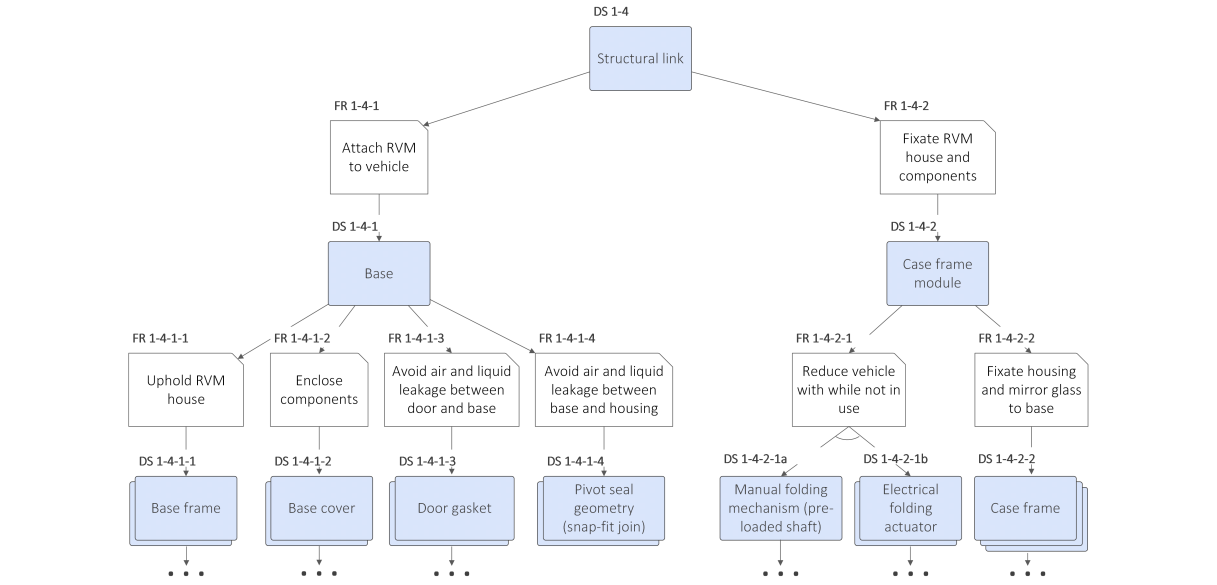


Figure A.4: Decomposition of the DS *structural link*.

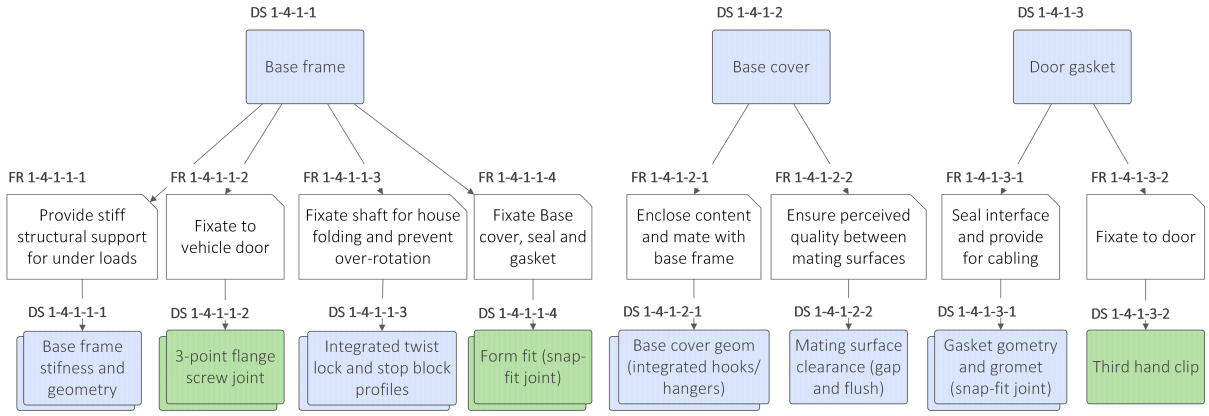


Figure A.5: Further decomposition of the *structural link*.

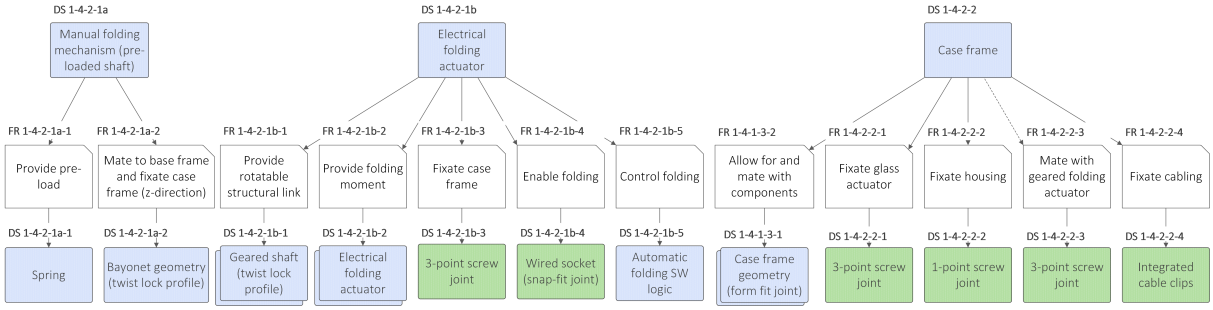


Figure A.6: Further decomposition of the *structural link*.

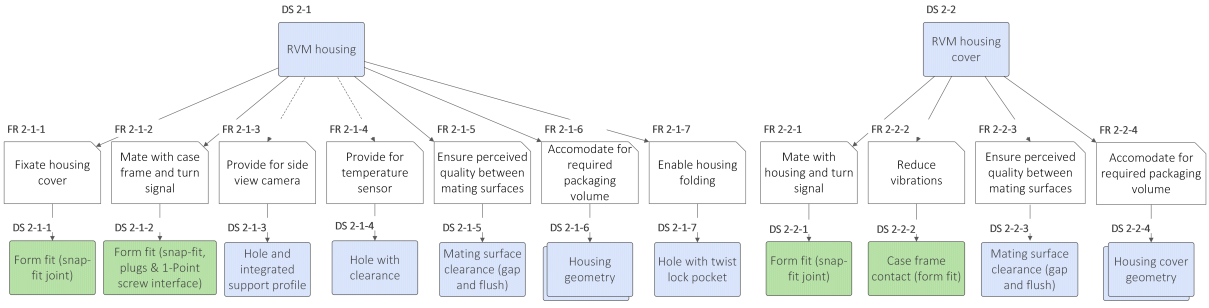


Figure A.7: Decomposition of the DS *RVM house*.

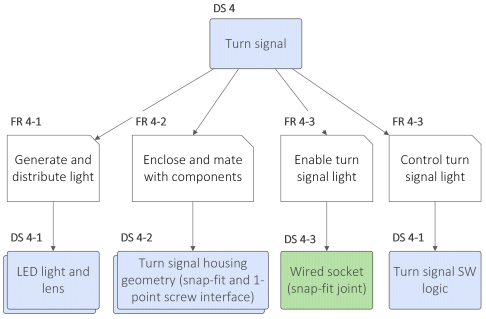


Figure A.8: Decomposition of the DS *turn signal*.

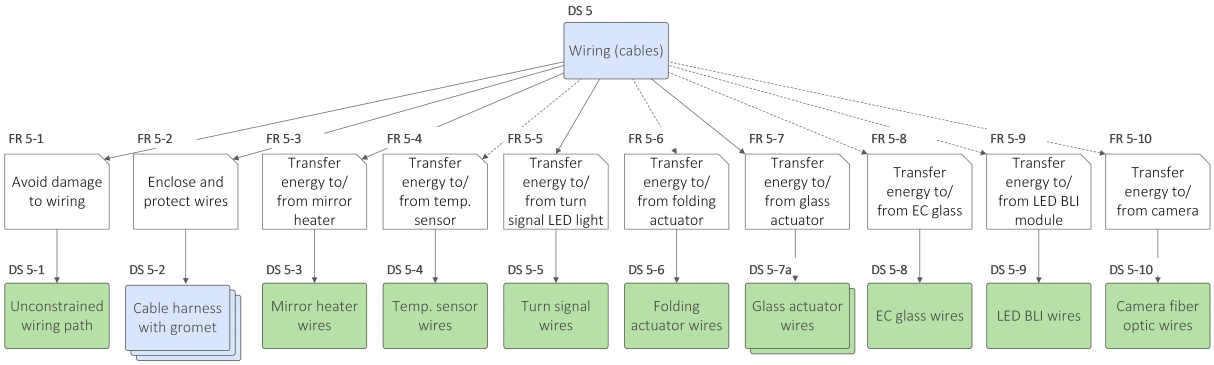


Figure A.9: Decomposition of the DS *wiring (cables)*.

A.2 Internal Variety

This section describes how commercial variety triggers internal variety for exterior RVMs. For a component, in this case the mirror glass, affected by drivers of internal variety this example shows how the complete bandwidth grows exponentially. The complete bandwidth is the solution-dependent extension of a sub-system's functional bandwidth. For the mirror glass, the functional bandwidth is defined by two features; *Electro Chromatic (EC) glass* and *Blind Spot Indication (BLI)*. Each feature has the possible values *true* and *false*. The mirror glass requires a specific design to fulfil each possible combination of these values. Moreover, the mirror glass has a design bandwidth composed of the features *geometry*, *chassis height* and *glass shape*. The glass can come in a *left-hand side (LHS)* or a *right-hand side (RHS)* geometry. It can be designed for a *low* or *high* chassis and it can come in a planar shape, an aspheric shape or in a convex shape, a convex shape with an Arabic warning text or in a convex shape with an English warning text. The possible configurations are visualised in Figure A.10. Even though only three functional embodiments are used, there are 26 unique mirror glass designs (Table A.1).

Table A.1: Mirror glass configurations.

No.	Description	No.	Description
G_1	Convex, RHS, small radius (Rx)	G_{14}	Convex, LHS, small radius (Ry), EC
G_2	Aspheric, RHS	G_{15}	Aspheric, LHS, EC
G_3	Convex, RHS, large radius (Ry) with English text	G_{16}	Planar, LHS, EC
G_4	Convex, RHS, large radius (Ry)	G_{17}	Convex, LHS, large radius (Ry), EC
G_5	Convex, LHS, small radius (Rx)	G_{18}	Convex, RHS, small radius (Rx), EC, BLI
G_6	Aspheric ,LHS	G_{19}	Aspheric, RHS, EC, BLI
G_7	Planar, LHS	G_{20}	Convex, RHS, large radius (Ry) with Arabic text, EC, BLI
G_8	Convex, LHS, large radius (Ry)	G_{21}	Convex, RHS, large radius (Ry) with English text, EC, BLI
G_9	Convex, RHS, small radius (Rx), EC	G_{22}	Convex, RHS, large radius (Ry), EC, BLI
G_{10}	Aspheric, RHS, EC	G_{23}	Convex, LHS, small radius (Rx), EC, BLI
G_{11}	Convex, RHS, large radius (Ry) with Arabic text, EC	G_{24}	Aspheric, LHS, EC, BLI
G_{12}	Convex, RHS, large radius (Ry) with English text, EC	G_{25}	Planar, LHS, EC, BLI
G_{13}	Convex, RHS, large radius (Ry), EC	G_{26}	Convex, LHS, large radius (Ry), EC, BLI

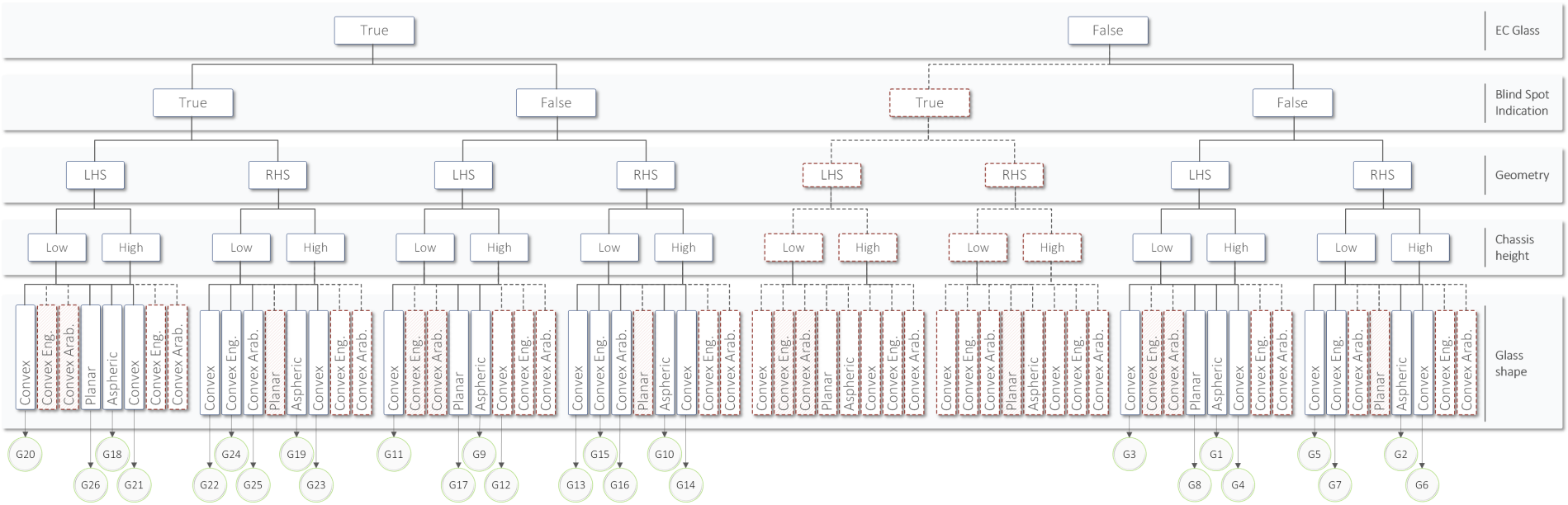


Figure A.10: Variant tree for the exterior Rear-view Mirror glass. Internal variety is driven by a proliferation of legal requirements and the height of different vehicle models. Even if variety steering practices reduce the offered commercial variety with 78%, 26 unique mirror glass configurations remain. If no combinations of solution alternatives and solution or function alternatives were disqualified (e.g. allowing vehicle models with high seating position in markets requiring English or Arabic warning texts), 39 unique mirror glass designs would exist. Without any variety steering practices, 52 unique mirror glass designs would exist. The red shaded branches of the tree are restricted by legislation.