

Eero Talus

STRUCTURED METHODS FOR THE DESIGN AND OPTIMIZATION OF MODULAR EMBEDDED SYSTEM PLATFORMS

Axiomatic design, design structure matrices and optimization
methods in architectural system design

Master's Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Professor Karri Palovuori
University Teacher Jouko Heikkinen
October 2024

ABSTRACT

Eero Talus: Structured Methods for the Design and Optimization of Modular Embedded System Platforms
Master's Thesis
Tampere University
Electrical Engineering
October 2024

Interest towards electric mobility is on the increase due to global climate targets and the development of electric vehicle (EV) technologies. While EVs were not a satisfactory alternative to internal combustion engine (ICE) vehicles before, most problems have already been solved. Consequently, design and implementation of large-scale charging infrastructure is the next step in supporting the growing fleet of EVs around the world. Such infrastructure requires complex systems composed of mechanics, electronics and software, and suitable design methods are therefore needed. Methods which solve complex design problems generally deal with multiple-criteria decision making (MCDM).

In this thesis, the architectural design of modular embedded system platforms is studied. The thesis consists of three parts. In the first part, the concepts of design, architecture and modularity are introduced and a selection of existing design methods is shown. Based on a comparison between these methods, axiomatic design and design structure system are selected for further analysis. Subsequently, the theoretical bases for axiomatic design and design structure system are introduced and an integrated design flow is shown. Novel methods for design parameter utilization and modularity cost analysis are also developed. In the second part, the common properties and functionalities of public EV charging devices are introduced. In the third part, an exemplary EV charging platform architecture is designed using the introduced methods. Five different architectural options are evaluated and a single one is selected based on developed optimization indicators. Finally, an overview of the effects, risks and advantages of structured design methods and modularity is given.

Architectures can be categorized in two ways: Based on the level of modularity and based on production paradigm. There is also a mapping between these two views: Modular architectures are most suitable for mass customization and mass individualization, whereas integral architectures are most suitable for mass production. While many design methods for modular architectures exist, axiomatic design and design structure system represent strong contenders. In general, using axiomatic design tends to result in modular functional architectures, but it does not impose restrictions on physical modularity. Therefore, physical design must be done using another method such as design structure system. These methods and the novel cluster utilization and cost-analysis methods are also used successfully in the thesis to design and optimize an exemplary EV charging system platform. While structured design methods and modularity offer many advantages, implementing them in an R&D organization also poses some risks and implementation must therefore be carefully planned in advance.

Keywords: architecture, design, modularity, axiomatic, dss, dsm, electric, vehicle, charging

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Eero Talus: Strukturoidut menetelmät modulaaristen sulautettujen järjestelmäalustojen suunnittelussa ja optimoinnissa
Diplomityö
Tampereen yliopisto
Sähkötekniikka
Lokakuu 2024

Kiinnostus sähköistä liikennettä kohtaan on kasvussa maailmanlaajuisten päästötavoitteiden sekä sähköautojen teknologisen kehityksen vuoksi. Vaikka sähköautot eivät olleet aiemmin järkevä vaihtoehto polttomoottoriautoille, useimmat niihin liittyvät ongelmat on jo ratkaistu. Seuraava askel sähköisen liikenteen mahdollistamiseksi onkin latausinfrastruktuurin suunnittelu ja rakentaminen. Sähköautojen latausjärjestelmät koostuvat esimerkiksi mekaniikasta, elektroniikasta ja erilaisista ohjelmistoista. Tällaisten monimutkaisten järjestelmien suunnittelu vaatii myös sopivia suunnittelumenetelmiä, joita kutsutaan yleisesti usean kriteerin päätöksentekomenetelmiksi (engl. multiple-criteria decision making, MCDM).

Tässä diplomityössä tutkitaan modulaaristen sulautettujen järjestelmäalustojen arkkitehtuurisuunnittelua. Työ koostuu kolmesta osasta. Ensimmäisessä osassa esitellään suunnittelun, arkkitehtuurin ja modulaarisuuden peruskäsitteitä sekä valikoima suunnittelumenetelmiä. Jatkokatsumukseen valitaan esitellyistä menetelmistä aksiomaattinen suunnittelu (engl. axiomatic design, AD) sekä suunnittelun rakennejärjestelmä (engl. design structure system, DSS). Ensimmäisessä osassa esitellään lisäksi valittujen menetelmien teoriapohja sekä menetelmiä yhdistelevä suunnitteluvuoro ja kehitetään kaksi uutta menetelmää suunnitteluparametrien hyödyntämistä ja modulaarisuuden kustannusvaikutusten analysointiin. Työn toisessa osassa esitellään sähköautojen latausjärjestelmien yleisiä ominaisuuksia. Työn kolmannessa osassa suunnitellaan esimerkinomainen arkkitehtuuri sähköautojen latausjärjestelmän sulautetulle järjestelmäalustalle. Suunnittelun kohteena on viisi arkkitehtuurivaihtoehtoa, joista valitaan työssä kehitettyjen menetelmien perusteella paras. Lisäksi kolmannessa osassa esitellään tutkimuksessa havaitut strukturoituihin suunnittelumenetelmiin ja modulaarisuuteen liittyvät riskit ja hyödyt.

Arkkitehtuureja voidaan luokitella joko modulaarisuuden tai valitun tuotantosuuntauksen perusteella. Luokittelutapojen välillä on kuitenkin myös yhteys: Modulaariset arkkitehtuurit soveltuvat parhaiten massakustoimintaan tai massayksilöllistämiseen, kun taas integraaliset arkkitehtuurit soveltuvat parhaiten massatuotantoon. Aksiomaattinen suunnittelu ja suunnittelun rakennejärjestelmä ovat työn tulosten perusteella hyviä vaihtoehtoja modulaarisen järjestelmäsuunnittelun tueksi. Aksiomaattinen suunnittelu johtaa kuitenkin vain funktionaalisen arkkitehtuurin modulaarisuuteen, joten fyysisen arkkitehtuurin suunnittelussa on käytettävä esimerkiksi suunnittelun rakennejärjestelmää. Näitä menetelmiä sekä työssä kehitettyjä uusia analysointimenetelmiä hyödynnetään onnistuneesti työn kolmannessa osassa. Työn tulosten perusteella strukturoituihin suunnittelumenetelmiin ja modulaarisuuteen liittyy kuitenkin myös joitakin riskejä, minkä vuoksi menetelmien käyttöönottoa suunnitteluorganisaatiossa pitää suunnitella ja harkita tarkoin.

Avainsanat: arkkitehtuuri, suunnittelu, modulaarisuus, aksiomaattinen, dss, dsm, sähkö, auto, lataus

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

USE OF ARTIFICIAL INTELLIGENCE IN THIS WORK

Artificial intelligence (AI) has been used in generating this work:

- No
- Yes

Acknowledgement of risks

I hereby acknowledge, that as the author of this work, I am fully responsible for the contents presented in this thesis. This includes the parts that were generated by an AI, in part or in their entirety. I therefore also acknowledge my responsibility in the case, where use of AI has resulted in ethical guidelines being breached.

PREFACE

The topic of this thesis came to life as a sidetrack from work projects I was working on at Kempower at the turn of 2023 and 2024. It was already well known at the time that modularity is often a preferred property of systems, and my team already had extensive knowledge about modularity in embedded systems as well. However, there was a lack of theoretical understanding on aspects such as how to arrive at an optimal module division for a modular system and what are the fundamental laws concerning modular design. As a consequence, I started researching the subject as the topic of my master's thesis beginning February 2024. Even though the journey from a topic idea to a complete thesis included some bumps along the way, I am joyful that I managed to push through. I am similarly glad about my fantastic work colleagues who I got to work with during the times I was writing my thesis.

Now, it is time to finish my studies at Tampere University. What a fantastic five and a half years it has been! It has been a great journey. It has also been a difficult journey. It has been a journey with a lot of ups and downs. It has been a journey with a lot of joy, but also some sadness. It has been a journey during which I have grown immensely as a person. Since I was just a kid, it had always been a dream of mine to be an electronics engineer. Now I can say, with confidence, that I am one, and I am sure the kid to whom a career in electronics was only a distant dream somewhere far, far away, would be extremely proud of the accomplishments he would come to make. Now it is time to start chasing new dreams, and let me tell you, I am thrilled about it! But there is still an important thing to remember! As Albert Einstein once said:

"The important thing is not to stop questioning. Curiosity has its own reason for existence. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvelous structure of reality. It is enough if one tries merely to comprehend a little of this mystery each day." -Albert Einstein in Old Man's Advice to Youth: "Never Lose a Holy Curiosity", LIFE Magazine (2 May 1955)

In Tampere, 17th October 2024

Eero Talus

CONTENTS

1.	Introduction	1
2.	Principles of System Design: A Literature Review	3
2.1	What is Design?	3
2.1.1	System Design Methods	4
2.1.2	Pugh Controlled Convergence	5
2.1.3	Analytic Hierarchy Process	7
2.1.4	Taguchi Method	8
2.1.5	Design Structure System	9
2.1.6	Axiomatic Design	12
2.1.7	Modular Function Deployment	15
2.2	What is Architecture?	16
2.2.1	Integrality and Modularity	17
2.2.2	Production Paradigms	19
2.2.3	Product Families, Platforms and Family Architectures	20
2.3	Conclusions of the Literature Review	21
3.	Axiomatic Design and Design Structure System	22
3.1	Theoretical Basis of Axiomatic Design	22
3.1.1	The Independence Axiom	24
3.1.2	The Information Axiom	26
3.2	Integration Analysis	27
3.3	Time-variant and Configurable Architectures	29
3.4	Designing Module Interfaces	31
3.5	Domain Based Product Configuration	32
3.6	Cluster Utilization	35
3.7	A Practical Integrated Design Flow	38
3.8	On Design Framework Selection	39
4.	Modelling the Cost-effects of Modular Architectures	40
4.1	Qualitative Model	40
4.2	Cost of Modularity	42
5.	Electric Vehicle Charging Technology	48
5.1	Applications Processing	48
5.2	Charging Standards	49
5.3	Networking and the Internet of Things	52
5.4	User Interfaces	54

5.5	Payment Solutions	55
5.6	Modular Connectivity	56
5.7	Safety Requirements	56
6.	Embedded System Architecture for an EV Charging System Platform	57
6.1	Methods	57
6.2	Architecture Design	57
6.3	Cluster Utilization Analysis	63
6.4	Modularity Cost Analysis	64
6.5	Results.	67
7.	Effects, Risks and Advantages of Structured Design Methods	69
7.1	Evaluating Product Architectures	69
7.2	Risks of Structured Design Methods	70
7.2.1	Cost of Modularity.	70
7.2.2	Total Complexity	70
7.2.3	Organizational Fit	70
7.2.4	Lack of Expertise	70
7.2.5	Amount of Work	70
7.3	Advantages of Structured Design Methods.	71
7.3.1	Project Management	71
7.3.2	Theoretical Proof	71
7.3.3	Product Quality	71
7.3.4	Product Maintenance	71
7.3.5	Product Configuration	72
8.	Conclusions	73
	References.	75
	Appendix A: FR-DP Hierarchy for an EV Charger.	80
	Appendix B: Design Matrices for an EV Charger	82
	Appendix C: Product Family Specification for an EV Charger	83
	Appendix D: Composite Design Matrix for an EV Charger	84
	Appendix E: Composite DP DSM for an EV Charger	85
	Appendix F: Naive Clustering Based on DP Hierarchy (Option A)	86
	Appendix G: Naive Clustering Based on a Constant Cluster Size (Option B)	89
	Appendix H: Modular Clustering Based on DP Interactions (Option C)	92
	Appendix I: Semi-Modular Clustering Based on DP Interactions (Option D)	95
	Appendix J: Completely Integral Clustering (Option E).	98
	Appendix K: Architecture Cost Comparison	101
	Appendix L: Design Parameter and Interface Cost Data	103

GLOSSARY

I	information
$P_x(v)$	cost of product variant v in architecture x
A	design matrix
CA	product assembly matrix
C	product configuration matrix
D	design parameter DSM
F	functional requirement DSM
H_B	binary cluster utilization matrix
H	cluster utilization matrix
P	product variant DSM
\bar{H}	average cluster utilization vector
\bar{P}_{DM}	module design parameter cost vector
\bar{P}_{IM}	module interface cost vector
\bar{P}_{MM}	module manufacturing cost vector
\bar{P}_x	variant cost vector of architecture x
\bar{V}	product variant vector
\overline{DP}	design parameter vector
\overline{FR}	functional requirement vector
$\eta_i(v)$	cluster utilization of cluster i in product variant v
$\det(\mathbf{X})$	determinant of matrix X
$\text{diag}(\bar{\mathbf{X}})$	diagonal matrix containing elements of vector $\bar{\mathbf{X}}$
$\log_2(x)$	binary logarithm
$\text{sgn}(x)$	sign function
$p_D(x)$	design parameter cost function
$p_I(x)$	interface cost function
$p_M(x)$	manufacturing cost function
p	probability

AC	Alternating Current
AD	Axiomatic Design
AFIR	Alternative Fuels Infrastructure
AHP	Analytic Hierarchy Process
ASIC	Application Specific Integrated Circuit
BEV	Battery-Electric Vehicle
BLE	Bluetooth Low Energy
CAN	Controller Area Network
CCS	Combined Charging System
CHAdemo	Charge de Move
CN	Customer Need
CP	Control Pilot
CPO	Charge Point Operator
CPU	Central Processing Unit
CS	Constraint
DC	Direct Current
DFA	Design for Assembly
DFM	Design for Manufacturing
DFX	Design for Excellence
DMM	Domain Mapping Matrix
DP	Design Parameter
DSI	Display Serial Interface
DSM	Design Structure Matrix
DSS	Design Structure System
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
FR	Functional Requirement
HDMI	High-Definition Multimedia Interface
HMI	Human-Machine Interface
I/O	Input/Output
I2C	Inter-Integrated Circuit
IC	Integrated Circuit

ICE	Internal Combustion Engine
IEA	International Energy Agency
IEC	International Electrotechnical Commission
IoT	Internet of Things
IP	Internet Protocol
LCD	Liquid-Crystal Display
LOS	Line of Sight
LVDS	Low-Voltage Differential Signaling
MC	Mass Customization
MCDM	Multiple-Criteria Decision-Making
MCS	Megawatt Charging System
MCU	Microcontroller
MDM	Multiple-Domain Matrix
MFD	Modular Function Deployment
MI	Mass Individualization
MIM	Module Indication Matrix
MIPI	Mobile Industry Processor Interface
MP	Mass Production
MPU	Microprocessor
NACS	North American Charging Standard
PAYG	Pay as You Go
PFA	Product Family Architecture
PHY	Physical Layer
PLC	Power Line Communication
PMIC	Power Management Integrated Circuit
PP	Proximity Pilot
PuCC	Pugh Controlled Convergence
PV	Process Variable
PWM	Pulse Width Modulation
QFD	Quality Function Deployment
RFID	Radio Frequency Identification
SAC	Standardization Administration of China

SAE	Society of Automotive Engineers
SoC	System on Chip
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network

1. INTRODUCTION

Mobility is turning electric. According to the International Energy Agency (IEA), 14% of cars sold in the year 2023 were electric, when the share was 9% and 5% in the years 2021 and 2020 respectively. Additionally, IEA forecasts that in 2030 at least 10% of the total vehicle fleet in the world, excluding two and three wheel vehicles, is electric with an annual growth rate of approximately 30% in 2023. [2] It is therefore clear that there is a large interest in moving towards electric mobility which also means appropriate large-scale charging infrastructure will be necessary in the coming years. According to IEA, the total light-duty and heavy-duty vehicle charging capacity in the world were approximately 200 GW and 75 GW respectively in 2022 with the forecasted capacities in 2030 being 1.9 TW and 420 GW [2].

The reasons behind the interest towards electric mobility are multifaceted. On one hand, the mobility transformation ties into the green transformation of the 21st century. For example, the European Union tries to achieve climate neutrality by 2050 by implementing their European Green Deal policy [21]. China, having emitted 35% of the world's CO₂ emissions in 2023 [1], has also been actively promoting the green transformation of its industrial sector [14] and was a leading contributor to the global green energy economy in 2023 [1]. Overall, electric mobility is a major factor in all countries' green transformation aspirations [1]. On the other hand, new developments in technology have made electric vehicles (EV) a viable alternative to traditional fossil fuel based internal combustion engine (ICE) vehicles. For example, range anxiety, caused by the lack of electric vehicle range, has traditionally been a major barrier to EV adoption. With recent developments in EV and battery technology, however, battery-electric vehicles (BEV) are already suitable for the vast majority of trips. [3]

The electric vehicle charging industry is a key enabler of the electric mobility transformation. Without a solid charging infrastructure in place, an ever-growing fleet of electric vehicles cannot be sustained. In addition to the increasing demand for electric vehicles and chargers, the technologies related to EV charging are constantly being improved with better batteries, improved charging protocols, new charging plugs and increased charging power just to name a few [19]. This puts even more pressure onto charging system manufacturers to provide customers with the latest technology in order to advance the growth of the electric vehicle fleet around the world.

The constant technological improvements require charging system manufacturers to adapt which poses challenges for charging system design. These kind of large-scale systems can be extremely complex and designing them requires a mix of cross-disciplinary knowledge from mechanics, power grid integration, high-power electronics design, embedded systems design, software design and cloud technologies just to name a few. A common issue with all of these fields is architecture, that is the high-level specification of how a system is built and how different parts of the system interface with each other. A well-designed architecture is a key element in building a successful and maintainable system which stands the test of time.

In this thesis, I study the structured architectural design of embedded system platforms from the electronics design viewpoint with an emphasis on the effects of modularity and integrality on system design. The thesis is roughly divided into three parts. In the first part, I introduce the principles of system engineering and a selection of system design methods used in existing literature. I also introduce the concepts of architecture, modularity and product families and how the architecture of a product relates to the product strategy of an R&D organization. In the second part, I give a more in-depth introduction to axiomatic design and design structure system in order to build a theoretical basis for applying these methods in product design. Building on axiomatic design, I introduce a novel method for analyzing the material utilization of a product platform and a novel method for analyzing the cost-effects of product modularity. In the third part of the thesis, I introduce the key functionalities of EV charging systems based on existing literature. Using this information, I apply the principles of axiomatic design and design structure system as well as the novel platform analysis methods to design and analyze an exemplary embedded platform for an EV charging device. Finally, I give an overview of the effects, risks and advantages of using structured design methods in system design.

The aim of this thesis is to give answers to and motivate further research on four research questions related to the architectural design of systems:

1. What structured methods for architectural system design can be found in existing literature and how do these methods compare with each other?
2. What are the main differences between integral and modular product architectures and on what basis should the level of modularity be chosen for a new product or product platform?
3. What are the production cost-effects of different levels of modularity and can the cost of a product be modeled as a function of modularity?
4. What technological and organizational effects, risks and advantages are there in implementing a structured system design method in an R&D organization when the aim is to modularize product architectures?

2. PRINCIPLES OF SYSTEM DESIGN: A LITERATURE REVIEW

This chapter introduces the main principles for designing engineered systems in a structured manner. The chapter discusses the theoretical basis for different design methods and includes a review and some examples of widely used frameworks for successful product design.

2.1 What is Design?

The concept of design depends heavily on the context of the word. For example, the Oxford Dictionary of English [64] defines the noun design as

1. "a plan or drawing produced to show the look and function or workings of a building, garment, or other object before it is made",
2. "a decorative pattern" and
3. "purpose or planning that exists behind an action, fact, or object".

These definitions capture the essence of the word quite well in an everyday context; a design can be, for example, a plan for a building, a piece of art, or the design of a piece of fabric. However, these definitions also leave some ambiguities in a broader context. For example, questions such as the following may arise.

- Does a design need to have an artistic purpose?
- Does a design need to have a physical form?
- What commonalities do all designs share?
- What are the main principles behind creating a successful design?

In the engineering context, we must assign the word a more strict but generic definition in order to build a theoretical framework for applying design principles to real-world problems. Therefore, when defining the meaning of the word design, we must think of the commonalities between all kinds of designs regardless of the field in question.

Benavides [27] proposes that motivation and design are always inherently linked together. A design is something that's created in response to a motivation in order to fulfill a need.

The motivation and need are also related to each other, because the need or the list of needs defines the motivation. Between the motivation and the design is the actual design process. The design process is the work of a designer who consumes some resources in order to fulfill the need. However, even though a motivation for a particular design may exist, this does not mean that a satisfactory design exists. This is because during the design process, there are always some restrictions and limits to what is possible. For example, a person may have the motivation to travel to the Moon, and while a design that satisfies this need has been proven to be possible, an ordinary person most likely will not be able to afford the design process. This is a constraint imposed on the need. While it is not a part of the original motivation or need, it is still an integral part of the design process. Other constraints might be, for example, the laws of physics, ie. there's an upper limit to how fast a person could travel to the moon without breaking the speed of light.

2.1.1 System Design Methods

The process of transforming a motivation into a design which satisfies a need is a complex issue which many people have tried to address by developing design methods with varying levels of theoretical rigour. Some of these methods are introduced in the following sections. The motivation behind developing new and better design methods consists of many things. For example, better design methods may be needed in order to

- maximize the odds of satisfying a need,
- maximize value,
- minimize cost of change and
- minimize resource consumption.

According to the earlier definition of design, success to satisfy a need is the ultimate goal of any design process. However, the degree to how well a design actually satisfies the need can vary. Assuming the motivation and needs are clearly defined, a successful design process is a necessary condition for a successful design [27]. The design process therefore has a major impact on the creation of a satisfactory design and good design methods are a necessity.

Cost of change, ie. the cost of making changes to an original plan when the plan is already being implemented, is a traditional problem in engineering design. The cost of making changes is generally fairly low early in the design and implementation process but grows rapidly as the design and implementation process progresses. [33] Therefore, designers should strive to make correct design decisions early in the design process to avoid changes later. The design method used for designing a system should therefore be built to minimize incorrect design decisions to being with.

Maximizing the value of a design refers to creating the best design possible, where value is the measure of perfection. Although value is difficult to define exactly, it often includes qualities such as functionality, performance and cost. [56] Value is tightly linked to business incentives which makes it an important parameter in product design. After all, one of the main reasons for a business to exist is to create something that provides value to customers.

It is clear that there is a need for design methods which try to achieve these and various other important goals in product design. The general problem of making design decisions based on a complex set of criteria while trying to satisfy the needs of various stakeholders is called multiple-criteria decision-making (MCDM). In the end, this is the essence of engineering design: Engineers need to make satisfactory design decisions based on various design criteria so that the initial pool of possible solutions is reduced to one solution that is implemented. The following sections introduce some tools and methods which help engineers make these decisions. The set of introduced methods is not a comprehensive set of all methods in existence. It is merely a selection of methods with widespread appearance in existing literature and suitability for, but not limited to, the design of embedded systems in particular.

2.1.2 Pugh Controlled Convergence

The traditional way to design a system is somewhat heuristic and empirical: Experts use their existing knowledge to design a system which is tested and refined until it satisfies the original need. During initial phases of design, the design decisions made by experts are mainly based on empirical and practical knowledge. This way experts extrapolate previous knowledge to fill in gaps where there is no data to support a particular design decision. The issue with such an approach is that it has a large risk of failure to satisfy the original need. [67] Heuristic design methods have been researched and formalized in order to address these issues. While still lacking some theoretical rigour, such methods have been seen to work well in practice. [28]

A widely used heuristic framework for making design decisions is the Pugh controlled convergence (PuCC) process [6, 4]. The PuCC process is based on creating a set of concept designs and coming up with a list of judgement criteria for comparing the designs. One of the concept designs, usually a known strong contender, is first chosen as a datum design, and the other ones are compared to it in matrix form. The result of each comparison is marked with the symbols S (same), + (better) or - (worse) in the cells of the comparison matrix. Finally, the number of S, + and - symbols is calculated for each concept design and the net score is calculated by subtracting the number of - symbols from the number of + symbols. The net scores can be used to rank the concept designs from best to worst. [6] Figure 2.1 shows an example of a single PuCC run.

Concept Criterion	1	2	3	4	5	6
Battery Life	DATUM	+	-	S	S	+
Weight		S	-	+	+	-
Size		+	+	S	+	+
Lifetime		S	+	-	+	S
Usability		-	-	+	-	+
Manufacturing Cost		-	S	S	S	S
$\Sigma+$	0	2	2	2	3	3
$\Sigma-$	0	2	3	1	1	1
ΣS	0	2	1	3	2	2
Net Score	0	0	-1	1	2	2

Figure 2.1. A hypothetical example of a single-run PuCC matrix for a battery powered product.

The end result of the PuCC process is a ranking between the different concept designs. A key thing to note is that the ranking of the first PuCC run is not necessarily grounds for selecting any particular design. Instead, the ranking can be used to select a subset of the designs for further development and subsequent PuCC runs. This is the convergence process in Pugh controlled convergence as illustrated in figure 2.2. [6]

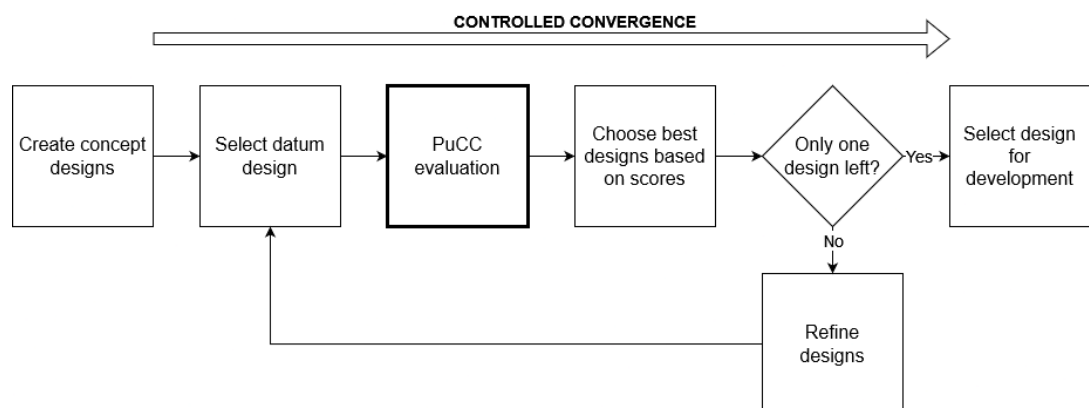


Figure 2.2. Pugh Controlled Convergence process flow. [6]

It is clear that this process is not very theoretically rigorous. Problems may arise, for example, if multiple experts disagree on whether a particular concept is better than

another in regards to a specific judgement criteria. In this case, experts must discuss their opinions and the judgement criteria to arrive at a consensus. If no consensus is found, an S symbol should be marked into the comparison matrix. [6]

2.1.3 Analytic Hierarchy Process

Another heuristic framework for making design decisions is the analytic hierarchy process (AHP) [62, 28, 4]. AHP is based on constructing a multi-level hierarchical representation of a complex decision making process as illustrated in figure 2.3 for a hypothetical battery powered product. First, the main objective of the decision process, ie. the first level of the AHP hierarchy is identified. Next, the objective is broken down into a set of judgement criteria onto consecutive layers of the hierarchy. Finally, alternative decision outcomes are identified and collected into the last level of the hierarchy. Pairwise comparison of all criteria within the hierarchical levels and hierarchic synthesis between levels is then done to figure out the relative importance of each criterion. This process is illustrated in a simplified manner in figure 2.4. The end result of the process is a relative ranking for all of the judgement criteria for the different decision outcomes. The AHP process is also somewhat subjective, since the pairwise comparisons are subjective in nature. Therefore, the AHP process includes a consistency verification step which aims to ensure that the comparisons are consistent with each other. [62, 45]

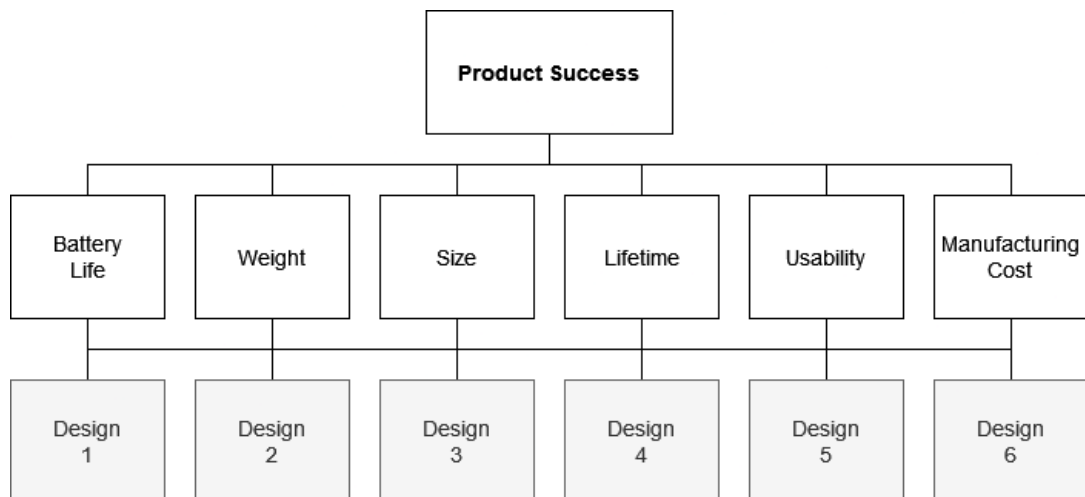


Figure 2.3. A hypothetical example of an AHP hierarchy for a battery powered product.

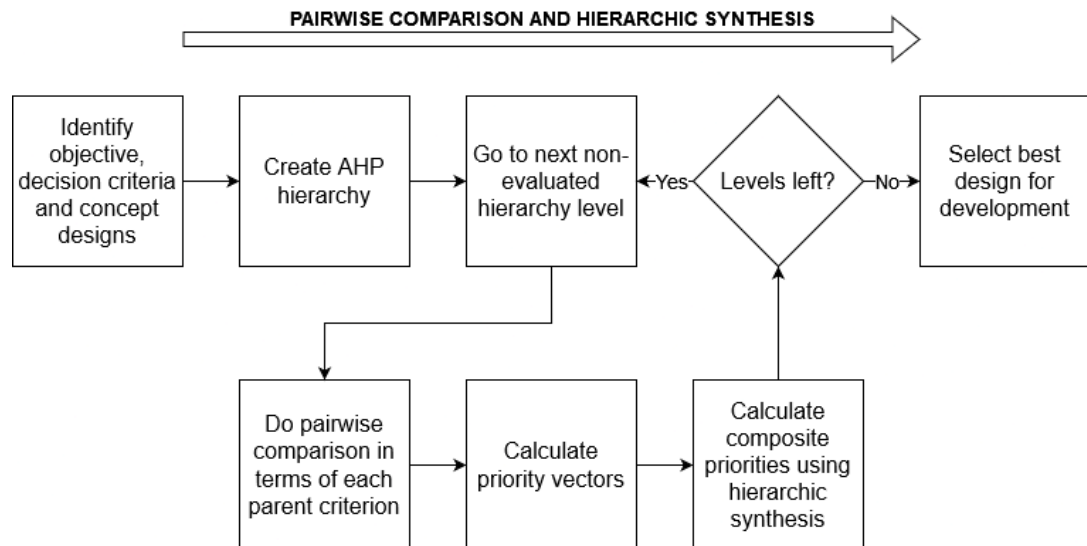


Figure 2.4. Analytic hierarchy process flow.

PuCC and AHP have been used successfully in making design decisions. Nevertheless, some researchers, such as Hazelrigg [44] and Dyer [41], argue that they are conceptually flawed and may lead to suboptimal design decisions. There have also been proposals of more rigorous decision making frameworks that use mathematical, logical or axiomatic approaches, or combinations of them, in making design decisions. The premise in these approaches is that by following a logically sound chain of inference while making design decisions, the most optimal decisions are always made. Such design methods are introduced next.

2.1.4 Taguchi Method

Structured design methods may focus on optimizing designs or processes in order to make products insensitive to, for example, environmental and manufacturing variables. This way, the quality of products can be improved. One such method is the Taguchi method which aims to increase the quality of goods by applying statistical methods. In particular, the method proposed by Taguchi minimizes a set of loss functions, where the loss function associated with a particular design objective may be one of

- larger-the-better,
- smaller-the-better or
- on-target.

The larger-the-better loss function maximizes the value of the design objective, the smaller-the-better loss function minimizes the value of the design objective and the on-target loss function aims to keep the value of the design objective at an exact value. [15]

The Taguchi method has been used successfully for optimization tasks, but a limitation of the method is that it is most suited to optimizing existing designs instead of creating new ones. Because the Taguchi method uses statistical methods extensively, applying it during the conceptual design of a system may be difficult. Other methods must therefore be applied to arrive at a conceptual design before the Taguchi method can be applied for optimization.

2.1.5 Design Structure System

Design structure system (DSS) is a tool which can be used to plan the engineering design process and to represent and analyze dependencies between design variables. In DSS, dependencies between variables are represented in a square matrix. [42, 65] This is illustrated in figure 2.5 for a hypothetical battery powered product.

		1	2	3	4	5	6
Battery Life	1	(X)					
Weight	2	x	(X)	x			
Size	3	x		(X)			
Lifetime	4				(X)		
Usability	5	x	x	x	x	(X)	
Manufacturing Cost	6	x		x			(X)

Figure 2.5. A hypothetical example of a design structure matrix for a battery powered product.

A mark in a cell in the design structure matrix (DSM) indicates that there is a dependency between two variables [65]. For example, the marks on row 2 of figure 2.5 indicate that the variable weight is dependent on the variables battery life and size. The marks in column 2, on the other hand, indicate that the variable weight is a dependency of the variable usability.

DSS is a flexible tool for representing the structure of systems and the interactions between system variables. A DSM can represent, for example, product architectures, organizations or processes. System variables can be, for example, components, people or activities and interactions can represent interfaces, communications or information flows respectively. During system design, a DSM can be used to analyze, cluster and sequence a system. [42]

If a DSM of a system is lower triangular, each variable is only dependent on variables preceding it in the DSM. Therefore, the values of the variables can be determined in the

order specified in the DSM. [65] In the DSM of figure 2.5, there is one mark above the diagonal making the matrix non-triangular. However, by reordering the DSM we can make it lower triangular. This operation is called sequencing. Reordering the DSM is allowed as long as the same reordering is applied to its rows and columns [65]. Figure 2.6 shows the previous DSM with variables 2 and 3 swapped. Now the DSM is lower triangular and the values of all variables can be determined.

		1	3	2	4	5	6
Battery Life	1	(x)					
Size	3	x	(x)				
Weight	2	x	x	(x)			
Lifetime	4				(x)		
Usability	5	x	x	x	x	(x)	
Manufacturing Cost	6	x	x				(x)

Figure 2.6. Reordered example of a DSM matrix.

The kind of reordering applied to this DSM is not always possible in the general case because real-world designs often contain feedback loops called circuits. In case a design contains circuits, its DSM can be partitioned into smaller blocks and the values of variables within the blocks can be determined by iteration. [65]

DSM sequencing mostly applies to DSMs which represent system variables that can be thought to exist in a sequence. However, if a DSM represents a non-orderable structure, such as a product architecture, clustering is a more useful operation. In clustering, the DSM rows and columns are ordered so that system variables are grouped into clusters according to a particular design objective. In practice, clustering is often used to minimize inter-cluster interactions and maximize intra-cluster interactions to partition a system into independent modules. [42] Figure 2.7 shows hypothetical DSMs representing product architectures and their clustering.

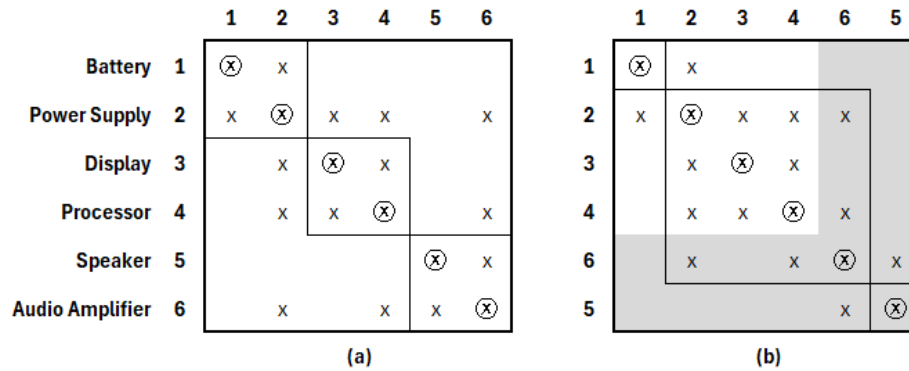


Figure 2.7. (a) An example of an architectural DSM and (b) a reclustered architectural DSM where rows 5 and 6 are switched and cluster boundaries are changed in order to reduce inter-cluster interactions.

In addition to modelling interactions within a single domain, DSS can also be extended to model interactions between domains using domain mapping matrices (DMM). A domain is a realm of system variables that form a DSM. Because DSMs can represent different kinds of systems, different DSM types can be thought to exist in different domains. For example, an R&D organization and a product architecture can be represented as two DSMs in different domains. The interactions between variables in different domains can in turn be represented using DMMs. The DSMs and DMMs of a system can also be combined to form a larger Multiple-Domain Matrix (MDM) which is a concise way of representing the intra-domain and inter-domain interactions in a single matrix. [42] The form of such a matrix is illustrated in figure 2.8.

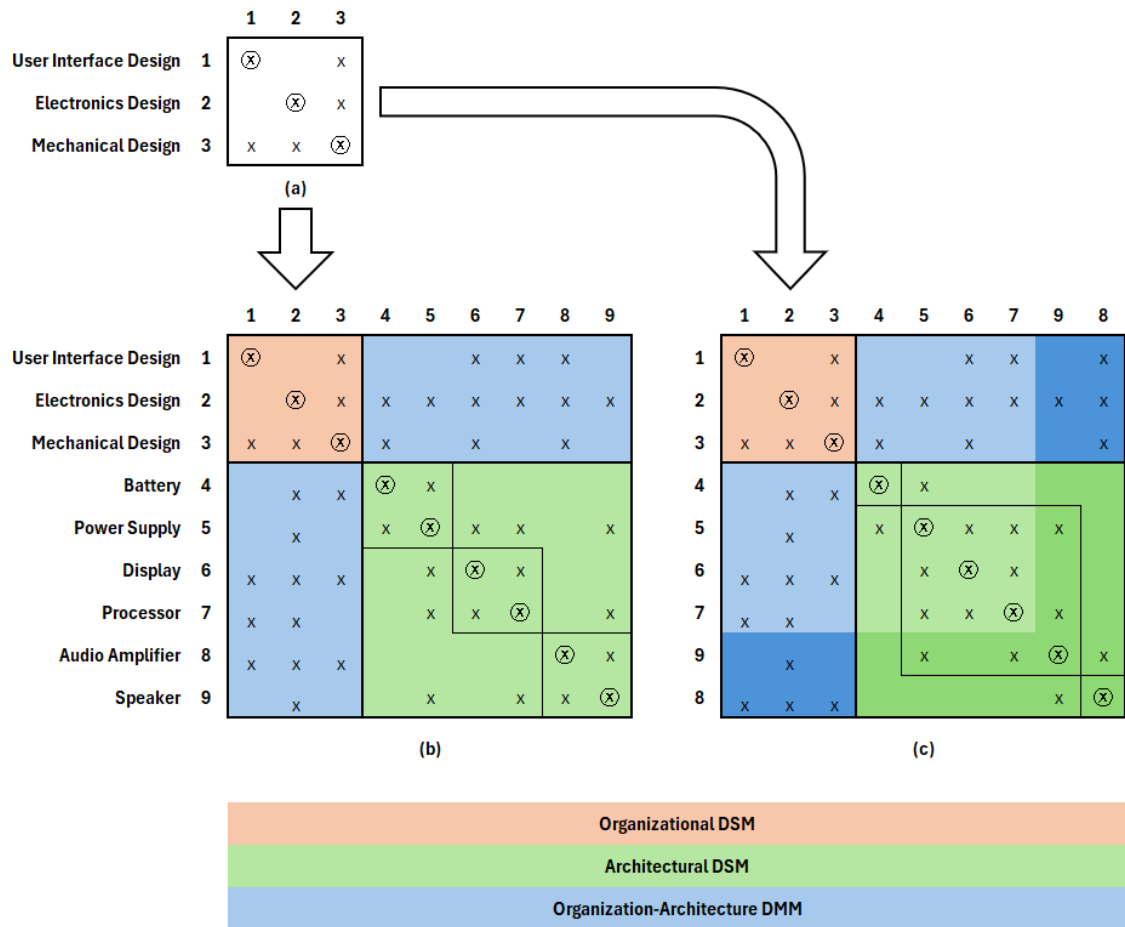


Figure 2.8. (a) An example of an organizational DSM, (b) an MDM based on the original architectural DSM from figure 2.7 and (c) an MDM based on the reclustered architectural DSM from figure 2.7. Combining an organizational DSM and an architectural DSM via a DMM forms an MDM.

Design structure system brings much needed rigour into parts of the conceptual design process and offers a concise way of analysing interactions between design variables. However, it does not guide the designer in selecting the proper design variables in order to arrive at a satisfying conceptual design. A solution to this problem is offered by axiomatic design introduced in the next section.

2.1.6 Axiomatic Design

Although the previously discussed methods have successfully formalized parts of the engineering design process, they all suffer from a major issue in the context of this thesis: The set of concept designs and judgement criteria is simply expected to exist before the design process begins. Although these methods provide great ways to select the best design or optimize a design, they do not even attempt to provide insight into how a concept design should be devised. Another issue in some of the previously introduced design frameworks is their apparent lack of formalism. While all frameworks are based on

rules and guidelines, the origin of these rules is often heuristic which may mean that they should not be considered the best practice. After all, there is no formal proof that these methods always result in good designs.

Many people argue that the field of engineering design cannot be adequately described using formal methods, and therefore using methods based on heuristics or expert knowledge is the only way forward. As a result, an abundance of formal frameworks for engineering design is not available. Nevertheless, one proposal of a formalized take into engineering design is axiomatic design (AD) proposed by Suh [69]. Suh identified the need for a formal description of engineering design, just as the ones which exist for other fields of science such as mathematics and physics, and created axiomatic design in response.

Axiomatic design is based on identifying the needs of customers, formulating functional requirements based on these needs, creating physical solutions that satisfy the requirements and coming up with processes to realize the physical solutions. Axiomatic design uses a domain-based approach where a product design is hierarchically divided into four domains: The customer domain, functional domain, physical domain and process domain. These domains consist of customer needs (CN), functional requirements (FR), design parameters (DP) and process variables (PV) respectively. [67] What these terms mean is not immediately obvious and they are therefore introduced in more detail in chapter 3. However, their relation to each other is illustrated in figure 2.9.

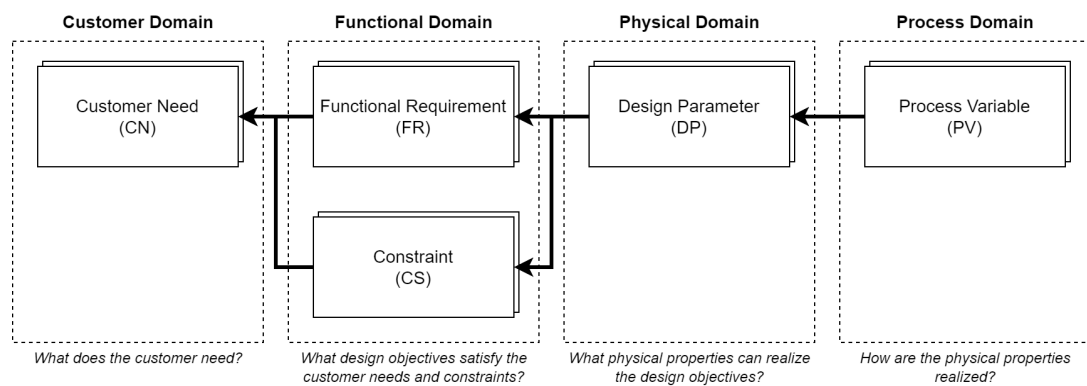


Figure 2.9. Domain hierarchy in axiomatic design. [67]

Axiomatic design is based on two core axioms: The independence axiom and the information axiom. The aim of the independence axiom is to ensure each FR can be satisfied independently of other FRs. The aim of the information axiom, on the other hand, is to minimize the information content of the system in order to maximize the probability of creating a system which satisfies its FRs. [67] These are the only pieces of information taken without proof, and the rest of the framework is built on top. While the axioms are taken without proof, the beauty of the axiomatic approach is that axioms can be considered to hold unless a counterexample is found. [69]

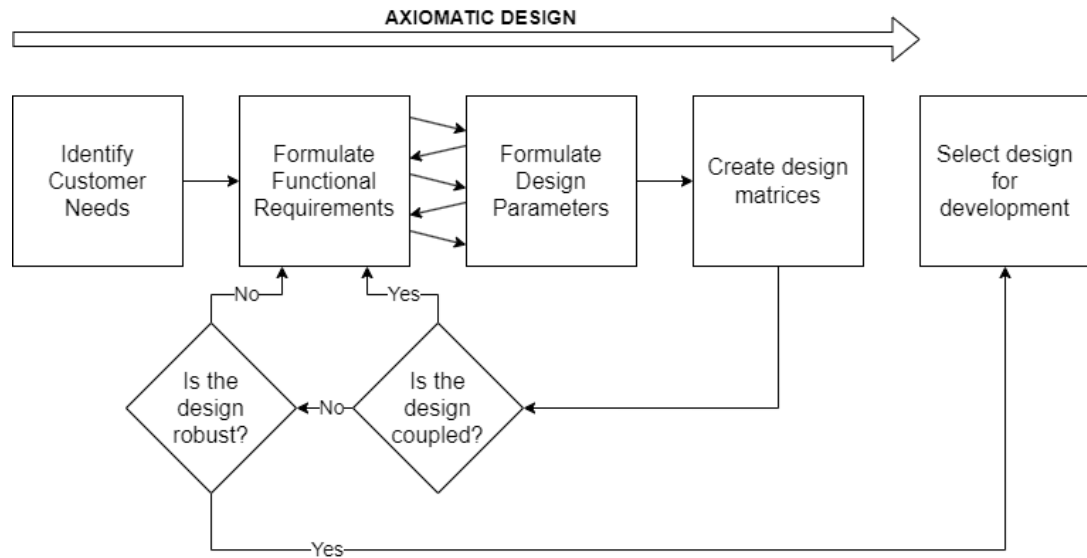


Figure 2.10. Axiomatic design process flow. [67]

The flow of the axiomatic design process is illustrated in figure 2.10. The axiomatic design process begins with identifying customer needs and then proceeds to formulating a comprehensive set of functional requirements and design parameters to satisfy these functional requirements. Formulating the FRs and DPs is a zigzagging process between the functional and physical domains. After a functional requirement is selected for a particular hierarchical level, the corresponding design parameter must be selected before considering the lower levels of the hierarchy. This zigzagging must be done because the selection of higher-level DPs affects the FRs on the lower levels of the hierarchy. [68] After selecting the FRs and DPs, the independence axiom and the information axiom must be applied to make sure that the design is not coupled and is robust enough in terms of the probability of satisfying the FRs of the design. At this point the selected FRs and DPs can still be changed to remove coupling and improve robustness. As hinted in the figure, axiomatic design uses matrix methods to describe the relationships between different domains and as a tool to analyze a design.

A shortcoming of axiomatic design also identified in existing literature is that axiomatic design only proposes how the functional architecture of a system should be designed and does not provide information on how the design parameters of a system should be composed to physical objects to create the design. [47] Other tools must therefore be used to obtain a physical architecture based on a functional architecture designed with axiomatic design. The design structure system is one such tool, and its application in conjunction with axiomatic design is covered in chapter 3.

2.1.7 Modular Function Deployment

Modular function deployment (MFD) is a holistic, structured system design method developed in the 1990s for the conceptual design and optimization of modular systems. MFD places strong emphasis on the business perspectives of design, and is therefore well suited to practical design problems. The MFD method is composed of a five-step process: [18, 55]

1. Clarify customer requirements
2. Select technical solutions
3. Generate concepts
4. Evaluate concepts
5. Optimize modules

While MFD is a complete framework on its own, it uses other structured design methods in its different steps:

1. Quality function deployment (QFD) is used to capture customer requirements and map them against product properties in the first step. [18, 55, 22]
2. A form of functional decomposition, such as axiomatic design, is used to define the functional requirements and technical solutions of the system in the second step. Pugh controlled convergence can be used for solution evaluation if needed. [18, 55, 22]
3. Module drivers are applied to technical solutions in the module indication matrix (MIM) to impose a company strategy onto the system in the third step. Clustering the MIM generates module concepts. [18, 55, 22]
4. Module concepts are evaluated based on their physical interfaces in the fourth step. [18, 55, 22]
5. Modules are optimized with design for excellence (DFX) methods in the fifth step. Aspects such as design for manufacturing (DFM) and design for assembly (DFA) may be optimized. In this step, specifications for all modules are also created. [18, 55, 22]

MFD can therefore be considered an extension and amalgamation of other design methods. Because of its holistic nature, it is a good candidate for the design of practical modular systems. However, in the context of this thesis, the scope of MFD is unnecessarily large.

2.2 What is Architecture?

With some of the design methods for engineering design introduced, it is still necessary to introduce the term architecture. This word is in common use in the engineering discipline, but just like for the word design, an accurate definition is needed. Following the same path as previously, the Oxford Dictionary of English [64] defines the word architecture as

1. "the art or practice of designing and constructing buildings" and
2. "the complex or carefully designed structure of something".

In the context of this thesis the former definition can be dismissed, but the latter one is quite suitable. Architecture is the structure of something. Buede [29] separates architecture into three different categories: Physical, functional and allocated architecture. In engineering design, physical architecture describes the physical structure of a system, the parts comprising the system and the relationships between the parts of the system. The functional or logical architecture of the system, on the other hand, describes the functional modules and information flow in the system. These two types of architecture are distinct from each other, but do not exist in isolation. They must always be developed side by side to ensure that they fit together seamlessly. In practice these architectures are hierarchical in nature. The third type of architecture, the allocated architecture, describes the integration between the requirements of the system, the physical architecture and the functional architecture. To this end, the allocated architecture describes how the functional architecture fulfills the requirements of the system by utilizing the physical architecture and what is the mapping between these two domains.

Less formally, architecture can be understood as a high-level description of a system or a plan that is created before any detailed design of the physical reality of the system takes place. The architecture of a system can be used as a planning tool and documentation for the detailed implementation of the system. After implementation, the system architecture can be used to document the structure of a system, for example to aid in maintenance tasks. It is therefore clear that the system architecture is a critical piece of information in all stages of system design. Nevertheless, creating a well-planned system architecture is a step in engineering design that is often overlooked, maybe due to hurry, ignorance or neglect. Especially in complex and large-scale systems, such neglect may lead to problems later on. Luckily, design methods such as the ones presented earlier can be used to aid in architectural design of systems. In the field of embedded systems, architectural design is often the job of system, software and hardware architects who work in conjunction with other design engineers. This group of people is also the main audience of this thesis.

2.2.1 Integrality and Modularity

Product architectures are traditionally categorized into two different types: integral and modular. Integral architectures are characterized by a high degree of functional and spatial cohesion which means that changes in one part of a system have many side effects in other parts of the system. Modular architectures, on the other hand, are comprised of modules with low spatial cohesion, loose coupling and therefore also a low degree of functional cohesion. [61, 46, 16]. However, the categorization into integral and modular architectures is not this clear-cut in reality. Instead, it should be treated as a spectrum from integral to modular with many variants in between. The measure of where an architecture lies on the integral-modular spectrum may then be called the modularity of the architecture.

A modular architecture can also be defined as having a one-to-one mapping between the elements of its functional and physical domains. This definition also agrees with the Independence Axiom of axiomatic design, because an architecture with a one-to-one mapping between its functional and physical domains constitutes an uncoupled design [46, 67, 11]. In axiomatic design terms, such a design is ideal because it contains minimal coupling. In other words, such a design may be considered to be completely modular. A completely integral architecture, on the other hand, is an architecture where all functionality is accomplished by a single physical module [11].

At this point, it is perhaps important to distinguish between functional and physical modularity. Because architecture can be divided into separate functional and physical architectures, modularity should also be handled in a similar manner. This means that the level of modularity of a design may be different in the functional and physical domains. Because designs created using axiomatic design must adhere to the Independence Axiom, functional modularity is always maximized in such designs. Physical modularity, on the other hand, depends on how design parameters are mapped into physical realities. Therefore, the physical modularity of a design may be varied quite freely. For example, a single electrical module may contain multiple design parameters while all functional requirements must always be independent. In this thesis, when a reference to the integrality or modularity of an architecture is made without specifying the domain explicitly, physical modularity is implied.

Integral and modular architectures are characterized by some advantages and disadvantages which are summarized in table 2.1. A particularly interesting concern is that there is a distinct lack of methods to determine whether a product should use a modular or an integral architecture. [46] This issue is approached in the next sections from a business perspective instead of using purely technical reasoning.

Table 2.1. Advantages and disadvantages of modular systems compared to integral systems. [46]

Category	Advantage	Disadvantage
Commonality	Modularity enables greater product variety through common components.	
Design re-use	Modular designs can utilize existing components which may reduce time-to-market and increase revenue.	
Product modification	Existing modular products can easily be modified post-manufacture.	
Project management	Design of well-defined and loosely-coupled modules is easy to manage in projects.	
Product architecture	Modularity reduces the complexity of system architectures.	
Assembly and disassembly	Assembly and disassembly of modular products is easier.	
Design process		Clear guidelines on deciding on the degree of modularity are not readily available.
Performance		Designers may need to compromise on performance due to commonality between products.
Mechanical properties		Modular designs tend to be mechanically larger and heavier due to more parts and more interfaces.

2.2.2 Production Paradigms

In order to serve customers, manufacturing companies must create products which satisfy customer needs. An obvious way to satisfy customer needs is to produce fully customized products, that is involve the customer in the design process. This way of designing products is, however, usually characterized by low production volumes. Another approach is to figure out a set of product attributes that appeal to a wider group of possible customers and to make products based on these attributes. While such products may not satisfy the needs of any particular customer exactly, they may satisfy the needs of many customers well enough to elicit buying decisions. The target in this kind of product design is therefore to achieve economy of scale. [20]

According to Hu [49], the manufacturing paradigms of products can be roughly divided into three categories: mass production, mass customization and personalization or personalized production. Rincon-Guevara et. al. [16] call the third type mass individualization or mass personalization. The terms mass production (MP), mass customization (MC) and mass individualization (MI) are used in this thesis.

Mass production is the production paradigm introduced by Henry Ford for manufacturing the world's first mass produced car, the Ford Model T. Mass production is typically characterized by little customizability in products, large volumes and good productivity. The ultimate target of mass production is economy of scale, that is reducing cost by increasing efficiency and production volumes. Nevertheless, mass production may lead to decreased profits because of issues such as neglected customer needs and quality issues. [49]

Mass customization was introduced as a response to increased demands for product variety. In mass customization, the manufacturer of a product defines the product family architecture (PFA) while the customer is free to select a product within the limits set by the product family. A PFA defines the common and specialized modules used in products, and by combining these, high variety can be obtained. [49, 16, 71, 52] Mass customization is therefore a middle-ground between mass production and completely customized products. The goal of mass customization is to offer a large variety of products with efficiency that closely matches that of mass production [24, 70, 10]. This is done by utilizing economy of scope [49]. Nevertheless, mass customization comes with its own set of problems. For example, the complexity of manufacturing a product is increased with high product variety [49]. In addition, properly implementing mass customization requires wider consideration in company operation and strategy [24]. While mass customization aims to fulfill customer needs better than mass production, it still does not necessarily satisfy all customer needs as well as fully customized products [49]. In extreme cases, too many product variants can even lead to a phenomenon called mass confusion where too many options deter customers from making purchases [70].

Mass individualization is the newest production paradigm made possible by advances in technology. Mass individualization is similar to mass customization in terms of a shared architecture between products, however in addition to manufacturer-defined modules, a mass individualized product may also contain personalized modules at least partly designed by the customer. This enables the product to suit the customer's needs more accurately. A mass individualized product is characterized by an open architecture with a set of common modules included in all products and a set of customized and fully personalized modules added on top. Mass individualization brings in more challenges compared to mass customization because now the customer is also a designer with possibly little experience in practical product design. In addition to design challenges, mass individualization also requires flexible manufacturing systems which are able to produce the personalized modules. [49, 16]

The integral-modular and MP-MC-MI spectra provide two different views into product architecting. The integral-modular spectrum is mostly defined in the technological domain, while the MP-MC-MI spectrum is mostly defined in the business and production domains. Nevertheless, there clearly exists a mapping between these spectra: A specific product architecture is inherently suited to a specific production paradigm. It is clear that a product with a very integral physical architecture cannot be easily modified according to a customer's specific requirements unless the design is based on customer input from the beginning. On the other hand, modules of a modular physical architecture can potentially be reordered and combined in new ways to arrive at a completely new product variant without redesigning the modules themselves. According to a taxonomy proposed by Rincon-Guevara et. al. [16], an integral product architecture always leads to a product that's suitable for mass production. A product with a modular architecture, on the other hand, always leads to a product that's suitable for either mass customization or mass individualization.

Because the modularity and production paradigm of a product are inherently related to each other, product architectures should be designed such that they are compatible with the chosen production paradigm from the beginning. The choice of production paradigm, on the other hand, is largely a business decision affected by factors such as market demands, customer value creation and shareholder value creation [49].

2.2.3 Product Families, Platforms and Family Architectures

A product family can be defined as a set of products which share common technology and functionality. A product platform, on the other hand, is the technological basis and physical implementation on which a product family is built. A product platform is described by a product family architecture which describes the structure of products and commonality between product variants. [52] As discussed in section 2.2.2, the concept of product

family architecture is especially important in a mass customization context because it enables customization, variety and commonality between different products of the same family.

Design for mass production has been the traditional way to design products, but if an organization aims to use a mass customization strategy instead, a product family and platform based approach should be used. This also means that instead of developing individual product architectures, the organization should aim to develop a product family architecture which encompasses all products sharing a common technology basis. This way economies of scale and scope can be achieved. [52] While a product family architecture is by nature more complex than the architecture of a single product, similar design methods can be used for its design. For example, Tseng and Jiao propose a design method based on axiomatic design [71]. A product family can also be considered a large flexible system as described by Suh [67], because a platform implies constant reconfiguration of products as a function of time and according to different customer needs. Design of time-variant architectures and PFAs using axiomatic design is covered in chapter 3.3.

2.3 Conclusions of the Literature Review

In this chapter, the concepts of engineering design and product architecture were introduced along with the basic principles governing the process of successful product design. Various design methods were also introduced and compared in order to give practical examples of how a design process may be organized. The set of introduced design methods is by no means a complete list. Many other methods exist, but the covered methods were chosen due to their widespread occurrence in existing literature.

Of the introduced design methods, axiomatic design was determined to be the most complete framework for designing new product architectures. While the other design methods are also good tools for multiple-criteria decision-making, product optimization and product structure analysis, axiomatic design additionally offers insight into how the architecture of a product should be designed beginning from customer needs and functional requirements. This way axiomatic design guides the design process from start to finish in order to arrive at a satisfying design solution. However, even axiomatic design is not a complete solution because it mostly gives guidance for designing the functional architecture of a product. The physical architecture must therefore be designed by other means while still keeping the axioms of axiomatic design in mind. To this end, design structure system may be used. How axiomatic design can be applied in conjunction with design structure system is covered in chapter 3.

3. AXIOMATIC DESIGN AND DESIGN STRUCTURE SYSTEM

In this chapter, a complete design flow incorporating axiomatic design and design structure system is introduced. Additionally, a method for optimizing the physical architecture of a design is developed. Because axiomatic design is often introduced in existing literature on a rather theoretical level, the approach taken in this thesis is more pragmatic in order to give the reader a good understanding of the framework. To this end, the chapter begins with an introduction to the theoretical basis of axiomatic design and steadily progresses towards a complete design flow which may be applied in practice.

3.1 Theoretical Basis of Axiomatic Design

Axiomatic design begins the conceptual design process by identifying what a customer needs which is nothing out of the ordinary for any design process. Customer needs are defined in the customer domain.

Definition 1*Customer Need, CN*

A customer need (CN) is a need which a customer wants satisfied. A CN describes what a customer is seeking in a product. [27] For example, a CN for a battery-powered device could be long battery life.

Because customer needs may be somewhat vague or unspecific, they must be turned into a more exact specification of what is required from the design. In axiomatic design, such a specification is called a set of functional requirements in the functional domain.

Definition 2*Functional Requirement, FR*

A functional requirement (FR) is a characterization of a single design goal based on a CN. The set of FRs used in an axiomatic design process must be the minimum set of independent functional requirements that completely characterizes all of the intended design goals. [67, 27] For example, an FR for the battery powered device could be a large enough battery capacity.

In addition to functional requirements, a design process usually has to deal with

requirements that do not originate from the customer, but are nevertheless limitations that affect the conceptual design. In axiomatic design, such limitations are called constraints (CS).

Definition 3

Constraint, CS

A constraint (CS) is an external limitation imposed on a design. A CS is something which must be included in the set of functional requirements, but which is not explicitly required by the customer.

After the requirements of the design are well specified, the next task is to figure out how these requirements can be fulfilled. Because fulfilling the requirements deals with the physical reality of a design, it is linked to the physical domain where design parameters are defined.

Definition 4

Design Parameter, DP

A design parameter (DP) is a physical variable related to a design [67, 27]. For example, a DP for the battery-powered device could be the battery capacity specification in watt-hours.

The last part of any design process is to figure out how the physical realities of a design can be produced. In axiomatic design, this is linked to the process domain where process variables are defined.

Definition 5

Process Variable, PV

A process variable (PV) is a process which can generate a DP [67, 27]. For example, a PV for the battery-powered device could be the resources needed to obtain the battery.

The different domains used in axiomatic design are organized into a hierarchy as previously illustrated in figure 2.9. The CNs, FRs, DPs and PVs are inherently related to each other. FRs describe how CNs are achieved, DPs describe how FRs are achieved and PVs describe how DPs are achieved. By beginning the design process by analyzing what the customer needs and then moving between the different design domains, a designer can create a design which satisfies the original needs. The CNs, FRs, DPs and PVs of a particular product are also organized in a hierarchical manner. For example, the top-level FRs are decomposed onto lower levels of the hierarchy until they cannot be decomposed any further. Such a hierarchical representation is in fact also one representation of the system architecture. [67]

Axiomatic design uses matrix methods to analyze the relationships between different domains. For example, FRs and DPs are related to each other by the design equation

$$\overline{\mathbf{FR}} = \mathbf{A} \cdot \overline{\mathbf{DP}}, \quad (3.1)$$

where $\overline{\mathbf{FR}}$ is a vector of functional requirements, \mathbf{A} is a design matrix and $\overline{\mathbf{DP}}$ is a vector of design parameters. [67] A similar transformation can also be formulated between other successive domains.

3.1.1 The Independence Axiom

As indicated by its name, axiomatic design is based on two core axioms. The first one is the independence axiom which ensures all FRs can be satisfied independently of each other. [67]

Axiom 1

The Independence Axiom

The independence of the functional requirements of a design must be maintained. [67]

In practice, the independence axiom is applied when creating the design matrix \mathbf{A} , because the FRs of a design are independent when the design matrix is either diagonal or triangular. [67] Selecting the proper design matrix is therefore a crucial step in applying axiomatic design. The elements of the matrix can be constants or functions of DPs, but in practice, during conceptual design phases, the elements are often simply marked with X or 0 symbols to indicate relationships between FRs and DPs on a conceptual level.

For example, assuming three FRs and three DPs, the matrix

$$\mathbf{A}_{\text{coupled}} = \begin{bmatrix} X & X & X \\ 0 & X & X \\ X & X & 0 \end{bmatrix} \quad (3.2)$$

is an example of an unsatisfactory design. Substituting it into equation 3.1 gives

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ 0 & X_{22} & X_{23} \\ X_{31} & X_{32} & 0 \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{bmatrix} = \begin{bmatrix} X_{11}DP_1 + X_{12}DP_2 + X_{13}DP_3 \\ X_{22}DP_2 + X_{23}DP_3 \\ X_{31}DP_1 + X_{32}DP_2 \end{bmatrix}. \quad (3.3)$$

We can see that all FRs depend on multiple DPs which means that changing one DP affects multiple FRs and therefore the independence axiom is violated. A design with a design matrix that is neither diagonal nor triangular is called a coupled design.

Definition 6*Coupled Design*

A coupled design is a design whose design matrix is neither diagonal nor triangular. Such a design is not satisfactory because it violates the independence axiom, ie. FRs cannot be satisfied independently. [67]

The matrix

$$\mathbf{A}_{\text{uncoupled}} = \begin{bmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{bmatrix} \quad (3.4)$$

is an example of a design with a diagonal design matrix. Substituting it into equation 3.1 gives

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{bmatrix} = \begin{bmatrix} X_{11} & 0 & 0 \\ 0 & X_{22} & 0 \\ 0 & 0 & X_{33} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{bmatrix} = \begin{bmatrix} X_{11}DP_1 \\ X_{22}DP_2 \\ X_{33}DP_3 \end{bmatrix}. \quad (3.5)$$

It is clear that in this case, each FR is satisfied by exactly one DP. This design does not violate the independence axiom and may therefore be satisfactory. A design with a diagonal design matrix is called an uncoupled design.

Definition 7*Uncoupled Design*

An uncoupled design is a design whose design matrix is diagonal. Such a design may be satisfactory because each FR can be satisfied independently of others. [67]

The matrix

$$\mathbf{A}_{\text{decoupled}} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ 0 & X & X \end{bmatrix} \quad (3.6)$$

is an example of a design with a triangular design matrix. In this case, the design equation 3.1 gives

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{bmatrix} = \begin{bmatrix} X_{11} & 0 & 0 \\ X_{21} & X_{22} & 0 \\ 0 & X_{32} & X_{33} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{bmatrix} = \begin{bmatrix} X_{11}DP_1 \\ X_{21}DP_1 + X_{22}DP_2 \\ X_{32}DP_2 + X_{33}DP_3 \end{bmatrix}. \quad (3.7)$$

The functional requirements $FR_2 \dots FR_3$ depend on multiple DPs. However, as long as the values for the DPs are determined in order from first to last, the FRs can be satisfied. The design does not violate the independence axiom and may therefore be satisfactory. A design with a triangular design matrix is called a decoupled design.

Definition 8

Decoupled Design

A decoupled design is a design whose design matrix is triangular. Such a design may be satisfactory because the FRs can be satisfied if the DPs are changed in the correct order. [67]

What if the design matrix is not square? If the number of FRs is greater than the number of DPs, the design is coupled or some FRs are not satisfied. If the number of DPs is greater than the number of FRs, the design is coupled or there are redundant DPs. [67] The designer should therefore always try to arrive at a square design matrix where the number of FRs and DPs is equal, and ideally try to make the design matrix diagonal.

3.1.2 The Information Axiom

The second axiom used in axiomatic design is the information axiom which aims to minimize the total information content of the design, where information is defined in the information theoretical sense. The information content I of an FR depends on the probability p of satisfying the FR: [67]

$$I = -\log_2(p). \quad (3.8)$$

This definition of information also agrees with Shannon's [63] famous definition of information content and entropy. By minimizing the information content of a design, the probability of satisfying the FRs of the design is maximized.

Axiom 2

The Information Axiom

The information content of a design must be minimized. [67]

It can be shown that the information content of a system can be minimized by minimizing the information content of the lowest-level FRs in the FR hierarchy. This means that in order to satisfy the information axiom, the designer should make the probability of

satisfying each lowest-level FR equal to one. [67] In practice, the information axiom is not widely used during conceptual design because calculating the probabilities associated with functional requirements may be extremely difficult when accurate design choices have not been made yet. It is therefore simpler to only apply the independence axiom during conceptual design.

3.2 Integration Analysis

The independence axiom of axiomatic design imposes a structure on the architecture of a system in the functional domain by mandating that functional requirements be independent. If the functional requirements of a system are not independent, they can be made independent by selecting new functional requirements. The information axiom, on the other hand, mandates that the information content of a system be minimized to maximize the probability of satisfying FRs. Even if the information content of a system is minimized on the module-level, assembling modules together may increase information. [67] A consequence of the information axiom is therefore that the information that results from assembling modules together should also be minimized. This is not an issue in the functional domain, because in the functional domain the interactions between modules are somewhat idealistic in nature and assembling functional modules together does not introduce additional uncertainty. This is not the case in the physical domain, however. Physical modules assembled into a system may have, for example, functional, spatial or information flow interactions. The information axiom therefore also imposes a structure on the system architecture in the physical domain by mandating that the physical architecture be designed such that its information content is minimized. However, axiomatic design does not give concrete methods for devising a physical architecture that satisfies this goal. This limitation of axiomatic design has also been identified in existing research [48, 7].

An obvious way to design a physical architecture based on a modular functional architecture is to use a one-to-one mapping between the functional and physical domains, but this may result in suboptimal results due to a large number of interfaces, increased complexity and increased physical size [46]. A better way to arrive at a modular physical architecture is therefore needed. If we consider a design where all functionality is accomplished by a single module to be the least modular and a design resulting from a one-to-one mapping between the functional and physical domains to be the most modular [11], the task is to move a modular architecture towards the integral end of the integral-modular spectrum. This is called integration analysis [66]. While the target of integration analysis is to make a system more integral, this does not directly contradict the goals of axiomatic design which embraces modularity. The information axiom of axiomatic design states that the information content of a system should be minimized,

and integration analysis aligns with this goal: A system with less interaction between physical modules contains less information.

According to the independence axiom, the elements of the functional architecture of a design, the functional requirements, must always be independent [67]. Therefore, a completely modular design with a one-to-one mapping between its functional and physical architectures should not be made less modular by introducing coupling in the functional domain because it may make the design unsatisfactory. Instead, coupling should be increased in the physical domain, that is the one-to-one mapping should be converted into a many-to-one mapping. In practice, this means that multiple design parameters are combined into a single physical module. This is also allowed by the independence axiom which only mandates the independence of functional requirements and not design parameters. Therefore, axiomatic design only enforces a high level of functional modularity but it does not impose restrictions on the level of physical modularity.

Design structure matrices are commonly used for integration analysis. DSMs have been used, for example, by Stiassnie and Shpitalni [66], Hong and Park [48], Cheng and Chen [32] and Tang et. al. [7] to derive a modular physical architecture from a modular functional architecture conceptualized using axiomatic design. In their research, axiomatic design was used to arrive at a set of design matrices which were converted into DSMs. The DSMs were clustered using either visual inspection or by specialized clustering algorithms for more complex designs [66]. The ultimate objective of DSM clustering was to minimize interactions between different clusters which represent different modules in the physical architecture. Hong and Park [47] also suggested an alternative integrated approach, where DSM decomposition and the independence axiom of axiomatic design were applied iteratively to arrive at optimized functional and physical architectures in a single design process. This approach was inverted compared to the others in the sense that DSM decomposition was used first and axiomatic design principles were applied afterwards to check whether the functional independence requirements of axiomatic design were satisfied.

The aforementioned approaches for integration analysis differ mainly by their method of creating the DSM based on the conceptual design created with axiomatic design. The methods are compared in table 3.1

Table 3.1. *DM to DSM transformation according to existing literature.*

#	Reference	Description	Analysis
1	[66]	Analyze spatial, process, material, information and energy interactions between DPs graphically. Create DSMs for all interaction types based on analysis.	Holistic analysis based on all interaction types.
2	[48]	Analyze functional relationships between DPs from the function structure of the conceptual design.	A simple method but only takes functional interactions into account. This shortcoming is also demonstrated in the publication.
3	[32]	Use a mathematical transformation to convert a DM into a DSM.	A mathematically rigorous approach.
4	[7]	Use a mathematical transformation to convert a DM into a DSM.	A mathematically rigorous approach.

In table 3.1, approaches 1 and 2 are based on analyzing the interactions between DPs based on knowledge about the DPs. Approach 1 can be treated as an extension of approach 2 because in addition to functional interactions, it also takes into account other types of interactions. Approaches 3 and 4 are based on transforming the design matrices of axiomatic design into DSMs using a mathematical transformation which appears to be very similar in both cases. Any of these methods could be used to create DSMs, but a method similar to approaches 1 and 2 is used in this thesis.

3.3 Time-variant and Configurable Architectures

Product architectures can be time-variant and configurable. For example, product upgrades or product configuration may cause changes in a product architecture so that FRs of a new product architecture differ from the FRs of an older one. However, changing the FRs of an existing design has a large risk of introducing coupling into the design or making it unsatisfactory. Provided that the original design is not coupled, modifying the set of FRs of a design may be divided into three cases: [69]

Add FR: Adding an FR either introduces coupling or leaves the added FR

unsatisfied. To fix this, a new DP must also be added. Adding a new DP may introduce coupling unless the added DP only affects the newly added FR.

Remove FR: Removing an FR makes a design redundant, ie. there are more DPs than FRs. Unless redundancy is of use in the application, extra DPs may add unnecessary materials and cost to the design.

Change FR: Changing an FR may leave FRs of the design unsatisfied, although in some special cases the new set of FRs may be satisfied by the old set of DPs.

It can be concluded that modifying the FRs of an existing design is not a straightforward task in the general case. It is therefore often easier to redesign the entire FR-DP-PV hierarchy than try to modify an existing one to remove coupling [69, 67]. If modifying an existing design is necessary, it is easiest to add an FR-DP pair where the newly added DP only affects the newly added FR, because such an addition cannot introduce coupling into the system. The second type of modification, ie. removing FRs, is also interesting. Provided that the original design is uncoupled or decoupled, removing an FR from the design cannot introduce new coupling because the independence axiom guarantees independence of FRs. However, removing an FR means that now the design contains more DPs than FRs, ie. it contains redundancy. Physically, having redundant DPs means that the design may contain, for example, unnecessary physical components. In order to optimize aspects such as material usage and cost, such DPs should be eliminated. This can only be done if the redundant DP does not affect any of the remaining FRs. Considering that a satisfactory design is either uncoupled, ie. its design matrix is the identity matrix, or decoupled, ie. its design matrix is triangular, this problem can be divided into two cases.

Uncoupled Case: Any unused DP can be removed, because a single DP only affects a single FR.

Decoupled Case: In the general case, redundant DPs must be removed in the reverse-order indicated by the design matrix. In special cases where a DP does not affect any FRs, it may be removed regardless of order.

The first case is self-explanatory. However, consider a decoupled design

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \\ FR_4 \end{bmatrix} = \begin{bmatrix} X_{11} & 0 & 0 & 0 \\ X_{21} & X_{22} & 0 & 0 \\ X_{31} & X_{32} & X_{33} & 0 \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{bmatrix}. \quad (3.9)$$

If we remove FR_4 , we get the design equation

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{bmatrix} = \begin{bmatrix} X_{11} & 0 & 0 & 0 \\ X_{21} & X_{22} & 0 & 0 \\ X_{31} & X_{32} & X_{33} & 0 \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{bmatrix}. \quad (3.10)$$

It can be seen that DP_4 does not affect any FR and can be removed. However, if we remove FR_4 instead, we get the design equation

$$\begin{bmatrix} FR_1 \\ FR_2 \\ FR_4 \end{bmatrix} = \begin{bmatrix} X_{11} & 0 & 0 & 0 \\ X_{21} & X_{22} & 0 & 0 \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{bmatrix}. \quad (3.11)$$

Unless X_{43} is zero, we cannot remove DP_3 from the design. We also cannot remove DP_4 because X_{44} cannot be zero. Otherwise the original design would have been coupled. Therefore, the design cannot be made non-redundant unless FR_4 is also removed.

FR removal is related to creating product variants by product configuration from a common product family architecture. Product variants are effectively created by removing unnecessary FRs from a complete product family architecture instead of adding new FRs to an existing architecture. In order to optimize the design, unused DPs should also be eliminated after FR removal. According to the earlier reasoning, removal of unused DPs can be done even without redesigning the entire FR-DP-PV hierarchy.

Removing FRs and unused DPs from a design is theoretically simple. However, taking into account the integration applied onto the physical architecture, such arbitrary removal of FRs and DPs is not always possible. In a physical architecture which is not completely modular, a single physical module may contain multiple DPs which means the module can only be removed if all of its DPs are unused. Otherwise, the physical module must be assembled into the final product even if some unused DPs remain. It is therefore crucial to consider the product variant portfolio early when deciding how integral a physical architecture is made.

3.4 Designing Module Interfaces

In the context of modular system design, another type of FR-DP modification is also of particular interest: Adding interfaces between physical modules. Obviously, interfaces are an integral part of the system architecture and should therefore be represented by FRs and DPs just like the rest of the system. However, specifying interfacial FRs and DPs

early in the design may lock the design to a particular module division prematurely before proper integration analysis is done. Therefore, it is better to leave out the exact interface definitions in early design stages. However, the question is, can interface additions cause coupling in the design? Let's model an interface between two physical modules as a bidirectional interface consisting of two FR-DP pairs, one output pair and one input pair. In the worst case, all existing FRs of the design depend on the input interface DP and the output interface FR depends on all existing DPs of the design. This can also be expressed as the design equation

$$\begin{bmatrix} FR_{in} \\ FR_1 \\ \vdots \\ FR_N \\ FR_{out} \end{bmatrix} = \begin{bmatrix} X_{in,in} & 0 & \dots & 0 & 0 \\ X_{1,in} & X_{1,1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ X_{N,in} & X_{N,1} & \dots & X_{N,N} & 0 \\ X_{out,in} & X_{out,1} & \dots & X_{out,2} & X_{out,out} \end{bmatrix} \begin{bmatrix} DP_{in} \\ DP_1 \\ \vdots \\ DP_N \\ DP_{out} \end{bmatrix}. \quad (3.12)$$

It is clear that even in the worst case, the design stays decoupled even if interfaces are added, provided that the original design was uncoupled or decoupled to begin with. It is therefore acceptable to leave out interfacial FR-DP pairs from the initial design and add them after integration analysis.

3.5 Domain Based Product Configuration

According to axiomatic design, the design matrix \mathbf{A} is a transformation from DPs to FRs. In early stages of design, \mathbf{A} often refers to a design matrix related to a single parent FR. However, we can also construct a composite design matrix which includes all FRs and DPs of the entire design. Only the leaf nodes of the FR-DP hierarchy need to be considered in the composite design matrix because only such nodes appear in the design explicitly [67]. Higher-level nodes are obtained as combinations of lower-level nodes. The composite design matrix is useful because it can be used to analyze and optimize the complete system instead of working only on a single hierarchical level.

		DP1			DP2					DP3								
					DP21			DP22										
								DP221										
		DP11	DP12	DP13	DP211	DP212	DP213	DP214	DP2211	DP2212	DP2213	DP222	DP223	DP31	DP32	DP33	DP34	DP35
FR1	FR11																	
	FR12																	
	FR13																	
FR2	FR21	FR211																
		FR212																
		FR213																
		FR214																
	FR22	FR221	FR2211															
			FR2212															
			FR2213															
	FR222																	
	FR223																	
	FR3	FR31																
FR32																		
FR33																		
FR34																		
FR35																		

Figure 3.1. An example of the form of a composite DM.

FRs can only depend on DPs on the same hierarchical level which is depicted by the larger off-diagonal cells in the composite DM. Design matrices related to leaf nodes are located on the diagonal.

During product configuration, we are initially interested in the mapping between product variants and FRs, ie. what features are included in product variants. To this end, let's introduce a new binary product configuration matrix C which is a transformation from the functional domain to a new product family domain. Let \bar{V} be a product variant vector in the product family domain. \bar{V} can now be expressed as

$$\bar{V} = C \cdot \bar{FR}. \quad (3.13)$$

Substituting equation 3.1 gives

$$\bar{V} = C \cdot A \cdot \bar{DP}. \quad (3.14)$$

The matrix CA is a transformation from the physical domain to the product family domain, ie. it describes which DPs are included in which product variants. This matrix can be

called the product assembly matrix, because it effectively indicates which DPs must be assembled in the final product based on its set of FRs. Each non-zero cell of \mathbf{CA} indicates that a DP is in some way included in a variant. The importance of this matrix is discussed more in section 3.6. It is evident that the transformation matrices \mathbf{A} and \mathbf{C} are effectively DMMs as defined by DSS. Therefore, we can also represent a product family as an MDM by relating the DSMs of the system via the matrices \mathbf{A} and \mathbf{C} . In order to construct the MDM, some auxiliary matrices are introduced next.

Let \mathbf{P} be a $W \times W$ binary DSM describing the W product variants included in a product family. This matrix can be called a product variant matrix. The interactions between product variants could be used to analyze, for example, product subgroups within the product family. However, in this case, let's assume the matrix \mathbf{P} is an identity matrix.

Let \mathbf{F} be a $N \times N$ binary DSM describing the N FRs of a product platform. By the independence axiom, this matrix must always be an identity matrix because FRs must be independent.

Let \mathbf{D} be a $N \times N$ binary DSM describing the N DPs of a product platform designed for the product family. The interactions between DPs may describe, for example, energy flows, information flows, spatial proximity or mechanical interfaces among other aspects.

During product family planning, the FRs included in product variants are selected on the columns of \mathbf{C} by marking ones for used FRs and zeros or empty spaces for unused FRs. The form of a product configuration matrix is illustrated in figure 3.2.

	FR ₁	FR ₂	...	FR _{N-1}	FR _N
Variant 1	1	1	...	1	1
...	1		...	1	
Variant W			...	1	1

Figure 3.2. An example of a product configuration matrix which maps N functional requirements into W product variants.

Once the DMM is filled according to the required product portfolio, an MDM is created by placing \mathbf{P} , \mathbf{F} and \mathbf{D} on the diagonal of a new matrix and connecting them with \mathbf{C} and \mathbf{A} on the off-diagonal positions. The form of the MDM matrix is illustrated in figure 3.3.

P	C	CA
C^T	F	A
(CA)^T	A^T	D

Figure 3.3. The form of a product family MDM which is a block matrix consisting of DSMs and DMMs.

The methods introduced in this section can be used during integration analysis to optimize the design for design objectives such as:

1. Optimal clustering based on criteria such as minimum inter-cluster interactions and maximum intra-cluster interactions.
2. Maximum utilization of DPs within physical modules across all product variants. This design objective is elaborated in section 3.6.
3. Cost optimization based on maximum utilization of DPs within physical modules. This design objective is elaborated in chapter 4.

3.6 Cluster Utilization

The product assembly matrix can be used to analyze the cluster utilization of a physical architecture in terms of the amount of used and unused DPs per cluster, ie. physical module. Suppose a design is clustered into K clusters and the cluster boundaries are defined as ordered pairs of the form

$$k_i = (a_i, b_i), \quad (3.15)$$

where a_i is the index of the first DP included in cluster i and b_i is the index of the last DP included in cluster i . While clustering is originally performed on the binary design parameter DSM D introduced in section 3.3, it is illustrated in terms of the matrix CA in figure 3.4.

	DP ₁	DP ₂	DP ₃	DP ₄	DP ₅	DP ₆	DP ₇	DP ₈	DP ₉	DP ₁₀	DP ₁₁	DP ₁₂	DP ₁₃	DP ₁₄
Variant 1	X	X		X	X		X		X		X			X
Variant 2	X	X				X		X	X				X	
Variant 3	X	X	X	X		X		X	X	X		X		
Variant 4	X	X	X			X	X		X	X			X	X
Variant 5	X			X	X	X		X	X		X	X	X	X
Variant 6	X	X				X	X	X	X					X
	a_1		b_1	a_2			b_2		a_3	b_3	a_4		b_4	

Figure 3.4. An example of cluster boundaries in a product assembly matrix \mathbf{CA} .

The cluster utilization of cluster i in product variant v can be defined as

$$\eta_i(v) = \frac{\text{num. of used DPs in } i}{\text{num. of total DPs in } i} = \frac{\sum_{t=a_i}^{b_i} |\text{sgn}([\mathbf{CA}]_{vt})|}{b_i - a_i + 1}, \quad (3.16)$$

where $\text{sgn}(x)$ is the sign function

$$\text{sgn}(x) = \begin{cases} -1, & x < 0, \\ 0, & x = 0, \\ 1, & x > 0 \end{cases} \Rightarrow |\text{sgn}(x)| = \begin{cases} 0, & x = 0, \\ 1, & \text{otherwise.} \end{cases} \quad (3.17)$$

Because the dependency between a product variant and a DP can be a non-binary value, the absolute value of the sign function is used to convert the factors in \mathbf{CA} into binary values before summation. The cluster utilizations of a product family can be expressed as a cluster utilization matrix

$$\mathbf{H} = \begin{bmatrix} \eta_1(1) & \dots & \eta_K(1) \\ \vdots & \ddots & \vdots \\ \eta_1(W) & \dots & \eta_K(W) \end{bmatrix}. \quad (3.18)$$

This can also be converted into a binary cluster utilization matrix

$$\mathbf{H}_B = \begin{bmatrix} |\text{sgn}(\eta_1(1))| & \dots & |\text{sgn}(\eta_K(1))| \\ \vdots & \ddots & \vdots \\ |\text{sgn}(\eta_1(W))| & \dots & |\text{sgn}(\eta_K(W))| \end{bmatrix} \quad (3.19)$$

which can be used to analyze which physical modules of a product platform must be assembled into a product variant. The binary cluster utilization matrix is used in chapter 4 for modelling the cost-effects of product variants.

The average cluster utilizations across all product variants can be written in vector form as

$$\bar{\mathbf{H}} = \mathbf{1}^T \mathbf{H} \cdot \text{diag}^{-1}(\mathbf{1}^T \mathbf{H}_B), \quad (3.20)$$

where $\mathbf{1}$ is a column vector of ones, $\mathbf{1}^T$ is its transpose and $\text{diag}^{-1}(\mathbf{X})$ denotes the inverse of a matrix whose diagonal elements are the elements of vector \mathbf{X} . The matrix $\text{diag}(\mathbf{X})$ is invertible if and only if $\det(\text{diag}[\mathbf{X}]) \neq 0$. The determinant of a diagonal matrix is the product of its diagonal elements. Therefore,

$$\det(\text{diag}[\mathbf{1}^T \mathbf{H}_B]) = \prod_{i=1}^K [\mathbf{1}^T \mathbf{H}_B]_i. \quad (3.21)$$

The product is zero if and only if at least one element of the vector $\mathbf{1}^T \mathbf{H}_B$ is zero, ie. a sum of a column of \mathbf{H}_B is zero. A sum of a column of \mathbf{H}_B is zero precisely when the corresponding cluster is completely unused in all product variants. Therefore, assuming that a design does not contain unused clusters, the average cluster utilization vector $\bar{\mathbf{H}}$ always exists.

The values of $\bar{\mathbf{H}}$ only take into account product variants in which the cluster is utilized, ie. product variants which do not utilize a particular cluster do not affect the average cluster utilization for that cluster. Analyzing $\bar{\mathbf{H}}$ further yields two cases:

- (a) $\bar{\mathbf{H}}_i = 1 \Leftrightarrow \eta_i(v) \in \{0, 1\} \forall v = 1 \dots W$,
- (b) otherwise $\bar{\mathbf{H}}_i \in]0, 1[$.

A value of $\bar{\mathbf{H}}_i = 1$ is the ideal case, because it indicates the capabilities of the cluster are utilized to its fullest across all product variants. A value of $\bar{\mathbf{H}}_i \in]0, 1[$ indicates that the level of clustering lies somewhere between poor and ideal with values closer to 0 being poorer and values closer to 1 being more ideal. A value approaching zero indicates the cluster is not utilized in any product variant. This may mean that the cluster is redundant and should be removed or it may mean that the cluster is not used in the current set of

product variants but is reserved for future use. However, it is important to note that adding new product variants after the physical architecture clustering is decided may lead to a suboptimal cluster utilization for new variants because clustering cannot be changed after the fact.

Cluster utilization can be used as an optimization target during integration analysis. The cluster utilization of each physical module should be optimized to be as close to unity as possible. The closer the utilizations are to unity, the better is the DP utilization in each physical module. An optimised cluster utilization also translates to reduced cost, because product variants include fewer unused DPs in an optimized design.

3.7 A Practical Integrated Design Flow

In practice, axiomatic design is often implemented only partially because during conceptual design, there is little information available to support the design process. A good example of this is the convention of marking FR-DP interactions with X symbols instead of mathematical functions and only applying the independence axiom instead of applying the information axiom as well. A practical high-level design flow incorporating axiomatic design and design structure system may be outlined as follows:

1. Collect customer needs from the customer and identify constraints.
2. Identify top-level functional requirements based on customer needs.
3. Select top-level design parameters which satisfy top-level functional requirements.
4. Move onto the next lower hierarchical level. Decompose each functional requirement on the parent hierarchical level into new functional requirements.
5. Select design parameters which satisfy the set of functional requirements on the current hierarchical level.
6. If the last hierarchical level has been reached, go to step 7. Otherwise go to step 4.
7. Compose design matrices for all hierarchical levels.
8. Verify that all design matrices are uncoupled or decoupled. If not, revise functional requirements and design parameters starting from step 2.
9. Compose a product portfolio and define product variants using the product configuration matrix introduced in section 3.3.
10. Create a composite design matrix which covers all individual design matrices.
11. Compose a binary design structure matrix which describes interactions between design parameters.
12. Cluster the DSM by visual inspection or by applying a clustering algorithm.
 - (a) Optimize inter-cluster and intra-cluster interactions.

- (b) Optimize cluster utilization based on the novel method from section 3.3.
 - (c) Optimize variant costs based on the novel method from chapter 4.
13. Verify feasibility of physical modules. If some modules are not feasible, go to step 11 and revise the design structure matrix.
 14. Begin detailed design of modules based on the designed functional and physical architectures.

This combination of axiomatic design for conceptual design and design structure system for integration analysis is applied in chapter 6 to design an exemplary system architecture for a public EV charging system platform.

3.8 On Design Framework Selection

The selection of design framework for this thesis was based on a set of requirements which were chosen to help in answering the research questions declared in chapter 1. The requirements were formulated as a set of four criteria: The selected framework must

- (a) have a sound, non-heuristic theoretical basis,
- (b) provide a mathematical formulation of the design problem,
- (c) provide a convenient basis for measuring the level of modularity of a design and
- (d) provide clear, non-ambiguous laws governing the creation of satisfactory designs.

Based on these requirements, axiomatic design was selected because it

- (a) is based on design axioms,
- (b) uses a mathematical mapping process to describe a design problem,
- (c) formulates a structure which can be used as a basis to measure modularity and
- (d) provides design axioms and corollaries which can be considered the laws of design.

Additionally, it was concluded that axiomatic design combined with design structure system for integration analysis represents one of the most complete and rigorous design frameworks for engineering design. In particular, axiomatic design is also well suited for designing embedded systems.

Even with its theoretical and mathematical rigor, researchers have identified possible issues with axiomatic design. For example, the axioms of axiomatic design may impose a preference structure on the designer [58]. As discussed in section 3.2, axiomatic design may also result in overly complex or mechanically large designs in some cases. However, in order to refute the logic used in axiomatic design, a counterexample disproving one or both axioms would be needed. While examples validating the axioms are numerous, counterexamples have not been found. [69, 59]

4. MODELLING THE COST-EFFECTS OF MODULAR ARCHITECTURES

As discussed in chapter 2, the target of designing modular architectures is to enable mass customization or mass individualization which in turn enable large variety with good efficiency. A major factor in reaching the efficiency improvements offered by mass customization or mass individualization is the choice of modularity level. As discussed in chapter 3, the naive way to design an architecture is to make it as modular as possible using a one-to-one mapping between the functional and physical domains. In this case, it may be detrimental or even impossible to try to make the architecture even more modular. On the other hand, it may be reasonable to make the architecture more integral if it enables cost optimization. Therefore, modularity versus cost is an important factor when designing architectures. Another situation in which analyzing the modularity versus cost of an architecture may be of interest is when trying to determine whether an existing product should be made more modular. In this case, the modularity and cost structure of the product are already known and the effects of varying its modularity may be studied. In this chapter, a novel mathematical model for describing the production costs of modular products is introduced. The model is also used as the basis for a cost-analysis method for assessing the cost-effects of varying the level of product modularity.

4.1 Qualitative Model

Traditionally, the cost structure of a product depends on production volume and it contains two types of costs. Variable costs depend on the number of units manufactured whereas fixed costs are independent of the number of units manufactured. When analyzing product architectures, modularity must also be taken into account. This can be done by considering that some costs are dependent on modularity while other costs are independent of modularity. The total cost is then a function of both volume and modularity. [11] If we assume that modularity and volume are independent, we can focus only on the cost versus modularity dimension. This is a simplification, of course, because changing product modularity may indirectly affect production volumes as well. Nevertheless, this is a reasonable assumption when only studying the effects of modularity in isolation.

Although modular architectures are considered advantageous in the industry, there is little research on the cost-effects of modularization. [13] Existing research mostly concludes that the extremes of the integral-modular spectrum are detrimental in terms of cost and that there is an optimum degree of modularity somewhere in between which results in a U-shaped cost versus modularity graph [23]. Highly integrated products tend to be unsuitable for high variety because oversized modules may contain unused features in some product variants. Highly modular products, on the other hand, may contain unnecessary parts which increase development and assembly costs. [13, 10]

Skirde et. al. [11] propose a method for analyzing the cost-effects of modularization by determining current product modularity and cost structure and estimating the cost structure of different modularity levels by extrapolating from available data. Hohnen et. al. [23] propose an indirect method for analyzing the manufacturing cost-effects of modularization. Their method is based on identifying modularity-dependent intermediate variables and then identifying the relationships between the intermediate variables and manufacturing costs to arrive at the total cost-effects. The two presented approaches are similar, but inverted in the sense that Skirde et. al. begin by identifying the parts of an existing cost structure, whereas Hohnen et. al. begin by identifying variables which affect the cost structure in order to describe the cost structure mathematically. The former method is therefore better suited for situations where a product already exists and the cost-effects of changing its modularity are of interest. The latter method is better suited for situations where a completely new product is being designed and its cost-structure is not known a priori. An apparent problem of the approach of Hohnen et. al. is that it may be difficult to identify the intermediate variables affecting costs without knowing what the elements of the total cost-structure of a product are first. However, neither of these methods are completely satisfactory because they are based on estimating an unknown cost by making assumptions about its relationship to the level of modularity.

The issue of cost versus modularity can also be approached by way of comparison. Let's compare a completely integral product I to a completely modular product M. The physical architecture of product I is characterized by a large main module performing all functionality whereas product M contains multiple smaller modules which together perform the same functionality as the main module of product I. To aid in analysis, the term design parameter as defined by axiomatic design is used. A design parameter of a product is a physical attribute which exists in the product to satisfy a particular functional requirement. Design parameters may sometimes correspond to physical modules, but not necessarily. The difference between them is that physical modules are joined together by well-defined interfaces, whereas design parameters are simply attributes of the product combined in one way or another. A completely integral product therefore consists of a set of design parameters contained in one module, whereas a completely modular product contains one physical module per design parameter.

It is reasonable to assume that the design parameters of I are the same as those of M, excluding design parameters related to interface implementation, because both products share the same core functionality and therefore require similar design parameters. The difference between the products I and M is therefore only in interface implementation details. A key thing to note is that two design parameters which satisfy a shared functional requirement always include an interface between the design parameters, because they must be combined in one way or another to arrive at the functional requirement. In an integral product, this interface may not be clearly separable from the design parameters themselves. In a modular product, on the other hand, interfaces are visible between design parameters located in different physical modules but not necessarily between design parameters which share the same physical module. Based on this model, the differences between integral and modular architectures are illustrated in figure 4.1.

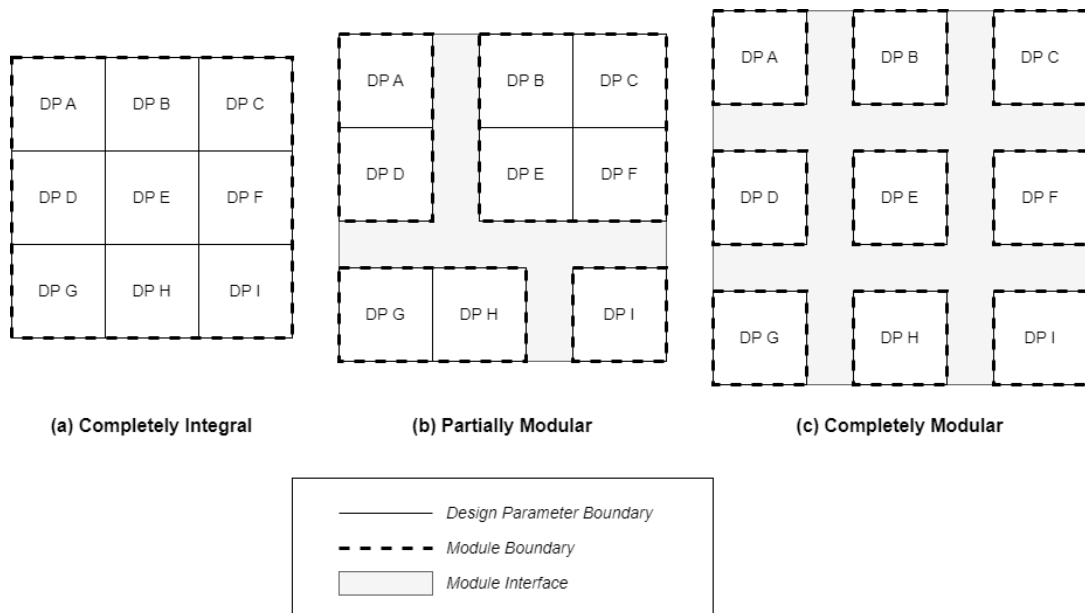


Figure 4.1. Structure of architectures with different levels of modularity.

In this figure, interfaces are simplified into a layer of glue between physical modules, whereas in reality not all physical modules are necessarily connected to each other.

4.2 Cost of Modularity

The cost of a completely integral and a completely modular product can be analyzed based on the qualitative model. Let's first analyze the cost structure of a completely integral product and later generalize the analysis to modular products.

Let x_i denote physical module i and $x_{i,j}$ denote design parameter j in physical module i . The cost of adding $x_{i,j}$ to a product can be written as $p_D(x_{i,j})$. Focusing on a completely integral product, the architecture only contains a single physical main module in which the

design parameters $x_{i,j}$ are all contained which means we can set $i = 1$. The total cost of N design parameters in an integral product can therefore be expressed as

$$P_{\text{integral}} = \sum_{j=1}^N p_D(x_{1,j}). \quad (4.1)$$

The cost of the product also contains some cost related to manufacturing. This manufacturing cost is often dependent on the mechanical size and complexity of the physical main module, because larger physical modules require larger machines, more manufacturing chemicals, more storage space etc. Complex physical modules also require more sophisticated manufacturing equipment and generally have a lower yield compared to simple physical modules. The manufacturing cost of physical module i can be written as $p_M(x_i)$ and in the case of a completely integral product, the manufacturing cost is simply $p_M(x_1)$. The total cost of the completely integral product can now be written as

$$P_{\text{integral}} = p_M(x_1) + \sum_{j=1}^N p_D(x_{1,j}). \quad (4.2)$$

As mentioned previously, there is always an interface between two design parameters which satisfy a shared functional requirement. In a completely integral product, these interfaces exist within a single physical module and are generally simple or a part of the design parameters and therefore incur no additional cost. The cost of the integral product is therefore completely described by equation 4.2.

In order to extend this analysis to modular products, the interface costs between modules must be included because interfaces in a modular product are more complex compared to the completely integral case. To this end, let's analyze a completely modular product with a one-to-one mapping between design parameters and physical modules. Let the cost of adding an interface $y_{l,k}$ from physical module x_l to physical module x_k be $p_I(y_{l,k})$ and let $p_I(y_{l,k}) = 0$ when physical modules x_l and x_k do not share an interface. The manufacturing cost is now also different compared to the integral case, because the product is not handled as a single physical module during manufacturing. Instead, smaller physical modules are handled individually, and the manufacturing costs are therefore associated with these physical modules. The total cost of the completely modular product can be expressed as

$$P_{\text{modular}} = \sum_{i=1}^N (p_M(x_i) + p_D(x_{i,1})) + \sum_{l=1}^N \sum_{k=1}^N p_I(y_{l,k}). \quad (4.3)$$

In the general case, the assumption of a one-to-one mapping between design parameters and physical modules must be dropped. Instead, design parameters may be clustered so that individual physical modules contain multiple design parameters. The manufacturing costs must therefore take into account the different numbers of design parameters included in each physical module. The interfaces between design parameters are also significantly affected by clustering. A clustered architecture may contain two types of interfaces. The first type is the same as in an integral product, ie. interfaces between design parameters which share a physical module. These interfaces do not incur additional cost, so their cost can be assumed zero. The second type are interfaces between design parameters in different physical modules. Because these interfaces are more complex, their cost must be accounted for in the total product cost. In the general case, the total cost of the modular product is

$$P_{\text{modular}} = \sum_{i=1}^K \left(p_M(x_i) + \sum_{j=1}^{N_i} p_D(x_{i,j}) \right) + \sum_{l=1}^K \sum_{k=1}^K p_I(y_{l,k}), \quad (4.4)$$

where K is the number of physical modules and N_i is the number of design parameters in physical module i . This can be simplified to

$$P_{\text{modular}} = \sum_{i=1}^K \left(p_M(x_i) + \sum_{j=1}^{N_i} p_D(x_{i,j}) + \sum_{k=1}^K p_I(y_{i,k}) \right). \quad (4.5)$$

By setting $K = 1$ which implies $N_i = N$, this equation is equal to the completely integral case of equation 4.2. By setting $K = N$ which implies $N_i = 1$, this equation is equal to the completely modular case of equation 4.3. This equation is therefore general enough to describe the cost of a product anywhere on the integral-to-modular spectrum. The total cost structure of the product can also be seen from the equation: It is a sum of the physical module costs where the cost of each physical module comprises of its manufacturing cost, design parameter cost and interface cost.

Another aspect which should be taken into account is product configuration. As discussed in chapter 3.3, whether an unused design parameter can be removed from a product depends on which functional requirements depend on the design parameter and how the physical architecture is clustered into physical modules. A completely integral architecture offers no possibility of such configuration, because all design parameters are included in a single physical module. A completely modular architecture, on the other hand, offers the most configurability because there is a one-to-one mapping between design parameters and physical modules. Removing unused physical modules has a direct effect on the cost of a product and it should therefore be included in the cost equation. To this end, the binary cluster utilization matrix \mathbf{H}_B introduced in chapter 3.3 can be used.

Taking product configuration into account, the cost of product variant v can be expressed as

$$P_{\text{modular}}(v) = \sum_{i=1}^K [\mathbf{H}_{\mathbf{B}}]_{vi} \left(p_M(x_i) + \sum_{j=1}^{N_i} p_D(x_{i,j}) + \sum_{k=1}^K p_I(y_{i,k}) \right). \quad (4.6)$$

This can also be expressed for all variants in vector form as

$$\bar{\mathbf{P}}_{\text{modular}} = \mathbf{H}_{\mathbf{B}} \cdot (\bar{\mathbf{p}}_{\text{MM}} + \bar{\mathbf{p}}_{\text{DM}} + \bar{\mathbf{p}}_{\text{IM}}), \quad (4.7)$$

where

$$\bar{\mathbf{p}}_{\text{MM}} = \begin{bmatrix} p_M(x_1) \\ \vdots \\ p_M(x_K) \end{bmatrix} \quad (4.8)$$

is a vector composed of the manufacturing costs of each physical module,

$$\bar{\mathbf{p}}_{\text{DM}} = \begin{bmatrix} \sum_{j=1}^{N_1} p_D(x_{1,j}) \\ \vdots \\ \sum_{j=1}^{N_K} p_D(x_{K,j}) \end{bmatrix} \quad (4.9)$$

is a vector composed of the design parameter costs of each physical module and

$$\bar{\mathbf{p}}_{\text{IM}} = \begin{bmatrix} \sum_{k=1}^K p_I(y_{1,k}) \\ \vdots \\ \sum_{k=1}^K p_I(y_{K,k}) \end{bmatrix} \quad (4.10)$$

is a vector composed of the interface costs of each physical module.

The costs P_{integral} and $P_{\text{modular}}(v)$ can now be compared. Let's use symbol a for physical modules and design parameters of the integral architecture, b for the physical modules and design parameters of the modular architecture and c for the interfaces of the modular architecture. The cost difference when moving from the integral architecture to the modular architecture is

$$\begin{aligned}
P_{\text{modular}}(v) - P_{\text{integral}} = & \left(\sum_{i=1}^K [\mathbf{H}_B]_{vi} \cdot p_M(b_i) - p_M(a_1) \right) \\
& + \left(\sum_{i=1}^K [\mathbf{H}_B]_{vi} \sum_{j=1}^{N_i} p_D(b_{i,j}) - \sum_{j=1}^N p_D(a_{1,j}) \right) \\
& + \left(\sum_{i=1}^K [\mathbf{H}_B]_{vi} \sum_{k=1}^K p_I(c_{i,k}) \right).
\end{aligned} \tag{4.11}$$

The ultimate question is, can this difference be made negative to achieve the cost savings offered by modularization? The equation consists of three parts: (a) The manufacturing cost difference (b) the design parameter cost difference and (c) the interface cost difference. It can therefore be expressed as the sum of these components as

$$P_{\text{modular}}(v) - P_{\text{integral}} = \Delta P_M + \Delta P_D + \Delta P_I. \tag{4.12}$$

Many variables affect the individual components of the cost difference equation:

- (a) **Manufacturing Cost Difference** ΔP_M : The cost function p_M is heavily technology dependent. In general, cost optimization may be possible by increasing modularity if it is cheaper to manufacture multiple smaller physical modules instead of manufacturing one large and complex physical module. This is likely often the case, because increasing the number of design parameters, the size and the complexity of a physical module tends to increase the difficulty of handling the physical module. Manufacturing costs may also be optimized with modularity if the architecture is clustered so that design parameters requiring expensive manufacturing technologies are separated from less expensive design parameters. This way only the physical modules with demanding design parameters may be manufactured using expensive technologies and cheaper technologies may be used for other physical modules which makes the manufacturing cost difference negative.
- (b) **Design Parameter Cost Difference** ΔP_D : The cost difference associated with design parameters depends heavily on product configuration. If the integral and modular products share the same design parameter set, the cost difference is zero because the design parameter costs of the integral and modular architectures cancel out. On the other hand, if some design parameters are unused in some product variants and consequently removed from a modular product, there can be large cost savings. The cost difference is nevertheless always negative or zero which favors a modular architecture.

- (c) **Interface Cost Difference** ΔP_I : The cost difference associated with interfaces is always positive and should therefore be minimized. The cost function $p_I(c_{i,k})$ is technology dependent, but it is also likely to be dependent on the complexity and number of interfaces of a module.

This model forms a basis for analyzing the cost-effects of varying product modularity, however it does not attempt to account for supply chain costs, final assembly costs, logistics costs, product lifetime costs and various other costs associated with bringing a product to market. It is nevertheless suitable for analyzing the modularity associated costs from the product design viewpoint. An example application of this model is provided in chapter 6.

5. ELECTRIC VEHICLE CHARGING TECHNOLOGY

Electric vehicle charging technology is cutting edge. It is characterized by a fast developing market where customer needs, standards, regulations and technologies are constantly evolving. An embedded system of an electric vehicle charging device contains a complex mix of technologies related to applications processing, power supplies, communication, networking, cloud technologies, user interfaces, safety features, wireless interfaces, vehicle communication among others. The implementation of many of these technologies are governed by standards of varying maturity level. An especially important part of any EV charger is the vehicle communication interface. At the moment, the market has not converged to any particular vehicle communication standard. Instead, there are different standards in use around the world. In addition, the existing standards are constantly being improved as more practical experience is gathered from real-world charging systems. The evolving nature of the EV charging market puts pressure on vehicle and charger manufacturers to keep up with the latest technologies available.

In this chapter, the main design aspects related to electric vehicle charging and embedded systems are introduced. While the introduced selection of design aspects is not a comprehensive list, it gives a good basis for defining the customer needs of an exemplary electric vehicle charging system later in the thesis.

5.1 Applications Processing

At the core of any embedded system is computing which always requires some kind of computing hardware. In practice, this means that an embedded system always contains a processor and related components such as power supplies, memories and various interfaces. The choice of processor is tightly linked to functionality within the embedded system. Some key selection criteria are processing power, amount of memory, power consumption and supported interfaces among others. These criteria narrow down the processor selection considerably.

Processor types can generally be divided into two categories: microcontrollers (MCU) with relatively little processing power but a lot of integrated peripherals and microprocessors (MPU) with high processing power but minimal integrated peripherals. A processor chip with a lot of integrated peripherals can also be called a System on Chip

(SoC) if it can be used on its own to implement a complete electronic system. [40] An electric vehicle charger intended for public charging often requires heavy applications processing due to user interfaces, connectivity and real-time charge control. Therefore, an MPU or SoC is most likely suitable as the main applications processor. Smaller low-power MCUs may be useful as auxiliary processors for low-level tasks which require, for example, functional separation from the rest of the system or hard real-time operation for safety critical tasks.

5.2 Charging Standards

The EV market is fragmented in terms of charging technology. Electric vehicles can be charged with either alternating current (AC) or direct current (DC) depending on vehicle support. AC charging is generally simpler since it uses charging electronics located on-board the vehicle whereas DC chargers bypass the on-board charger and use external charger hardware instead. This also means that DC chargers can use larger charging power because they are not limited by the on-board vehicle charger [8]. A second difference between charging technologies is the method of vehicle communication which can be implemented using various protocols [54]. A third difference is the choice of physical interface, ie. the charging plug used for the charging device. [8, 54] Although the market is currently fragmented, standardization bodies work to create national and international standards to ensure compatibility between vehicles and chargers. Standardization bodies related to electric vehicle charging include the International Electrotechnical Commission (IEC) internationally, Society of Automotive Engineers (SAE) in the USA and the Standardization Administration of China (SAC) in China. This thesis focuses on IEC and SAE standards due to their widespread use internationally.

In practice, charging technologies are developed by automotive manufacturers and other parties either on their own or as a part of an initiative such as CharIN. An example of a work in progress standard is the Megawatt Charging System (MCS) that is being developed by CharIN [31]. Development of the technical details is ongoing by the members of the initiative and once the technology is finalized, standardization bodies such as IEC and SAE publish standards for the technology. To this end, IEC and SAE have already prepared the standards IEC 63379 [39] and SAE J3271 [25] respectively.

The main charging standards in use today are summarized in table 5.1.

Table 5.1. Important IEC and SAE standards for EV charging. [43]

Standard	Region	Status	Contents
IEC 61851	International	Active	AC and DC charging, physical connection and communication
IEC 62196	International	Active	AC and DC charging, charging connector and communication
IEC 63379	International	Work in Progress	Megawatt Charging System, work in progress
SAE J1772	North America	Active	AC charging, charging connector and communication
SAE J3068	North America	Active	DC charging, charging connector and communication
SAE J3400	North America	Active	North American Charging Standard
SAE J3271	North America	Work in Progress	Megawatt Charging System, work in progress

According to SAE J1772, electric vehicle chargers are categorized into three levels [54, 17].

Level 1: Slow AC charging from 120V outlets. Typically used for home charging.

Level 2: Slow and fast AC charging from 240V outlets. Typically used for home and public charging.

Level 3: Fast DC charging. Typically used for public and fleet charging.

IEC 61851-1:2019, on the other hand, categorizes charging devices into four modes [34, 43].

Mode 1: AC charging from standard outlets. Implements no communication features. Typically used for home charging.

Mode 2: AC charging from standard outlets with a dedicated charging device. Implements communication between the charger and EV. Typically used for home charging.

Mode 3: AC charging from a charging device permanently connected to an AC supply. Implements communication features between the charger and EV. Typically used for public and fleet charging.

Mode 4: DC charging from a dedicated charging device. Implements communication features between the charger and EV. Typically used for public and fleet charging.

This thesis focuses on charging devices intended for public charging infrastructure for which SAE J1772 level 3 or IEC 61851 modes 3 and 4 are of interest. Such devices are based on electric vehicle charging equipment (EVSE) where an external charging system controls the charging process as opposed to a charger that is internal to the vehicle.

Because there are many modes of charging, there are also many different requirements for charging connectors. Examples of some commonly used charging connectors are shown in figure 5.1.



Figure 5.1. Examples of commonly used EV charging connectors.

IEC 62196-2:2022 defines three connector types for mode 1-3 charging [37].

Type 1: Connector for single phase AC charging. This connector type is identical to the connector used in SAE J1772 [54] and also known as the Yazaki connector.

Type 2: Connector for single phase or three phase AC charging. This connector type is identical to the connector used in SAE J3068 and is also known as the Mennekes connector. This connector is shown in figure 5.1.

Type 3: Connector for single phase or three phase AC charging. This connector is also known as the Scame connector.

In the type 1-3 connectors, communication between the charger and the electric vehicle uses the proximity pilot (PP) and control pilot (CP) signals present on the connector. Proximity pilot is used for cable connection detection and control pilot is used for post-connection communication between the charger and the electric vehicle [34]. This form of communication is also called basic communication [35].

IEC 62196-3:2022 defines four connector types for mode 4 charging [38].

Type AA: This connector is used for DC charging and is also known as the CHAdeMO connector. The type AA connector is used for chargers implementing system A as specified in IEC 61851-23:2023 [35]. System A uses Controller Area Network (CAN) for digital communication between the charger and the electric vehicle. This connector is shown in figure 5.1.

Type BB: This connector is used for DC charging mostly in China and is also known as the GB/T connector [54]. The type BB connector is used for chargers implementing system B as specified in IEC 61851-23:2023 [35]. System B also uses CAN for digital communication between the charger and the electric vehicle.

Type EE: This connector is used for combined AC and DC charging and it extends the type 1 connector defined in IEC 62196-2:2022 to add DC charging support. It is used for the combined charging system (CCS) and also known as the CCS1 connector. The type EE connector is used for chargers implementing system C as specified in IEC 61851-23:2023 [35]. In addition to basic communication, system C uses Internet protocol (IP) based power-line communication (PLC) for digital communication between the charger and the electric vehicle [36].

Type FF: This connector is used for combined AC and DC charging and it extends the type 2 connector defined in IEC 62196-2:2022 to add DC charging support. It is used for the combined charging system (CCS) and also known as the CCS2 connector. Similar to type EE, the type FF connector is used for system C chargers and it also uses the same communication technologies. This connector is shown in figure 5.1.

Another widely used charger type is the AC and DC capable North American charging standard (NACS) developed by Tesla, Inc. [50] which is commonly used on Tesla vehicles. Tesla has also opened the connector specification for use by other automotive manufacturers [50] and it is consequently standardized in SAE J3400 [26].

The large number of different charging technologies poses challenges for charging device manufacturers due to the need to support as many vehicle models as possible. Interface selection and implementation is therefore a crucial step in designing a charging system.

5.3 Networking and the Internet of Things

Many electric vehicle charging devices can be considered a part of the Internet of things (IoT) because IoT technologies are often used in networked embedded systems which are geographically distributed. Internet of things refers to networked devices which communicate with each other to perform a particular task [57]. IoT technologies can be used, for example, to monitor devices, update software, perform management and

maintenance tasks and collect analytics among other things. Cloud applications also go hand in hand with IoT. However, these functionalities require some form of network connectivity which requires hardware and software support from the embedded system. Some connectivity options are illustrated in figure 5.2.

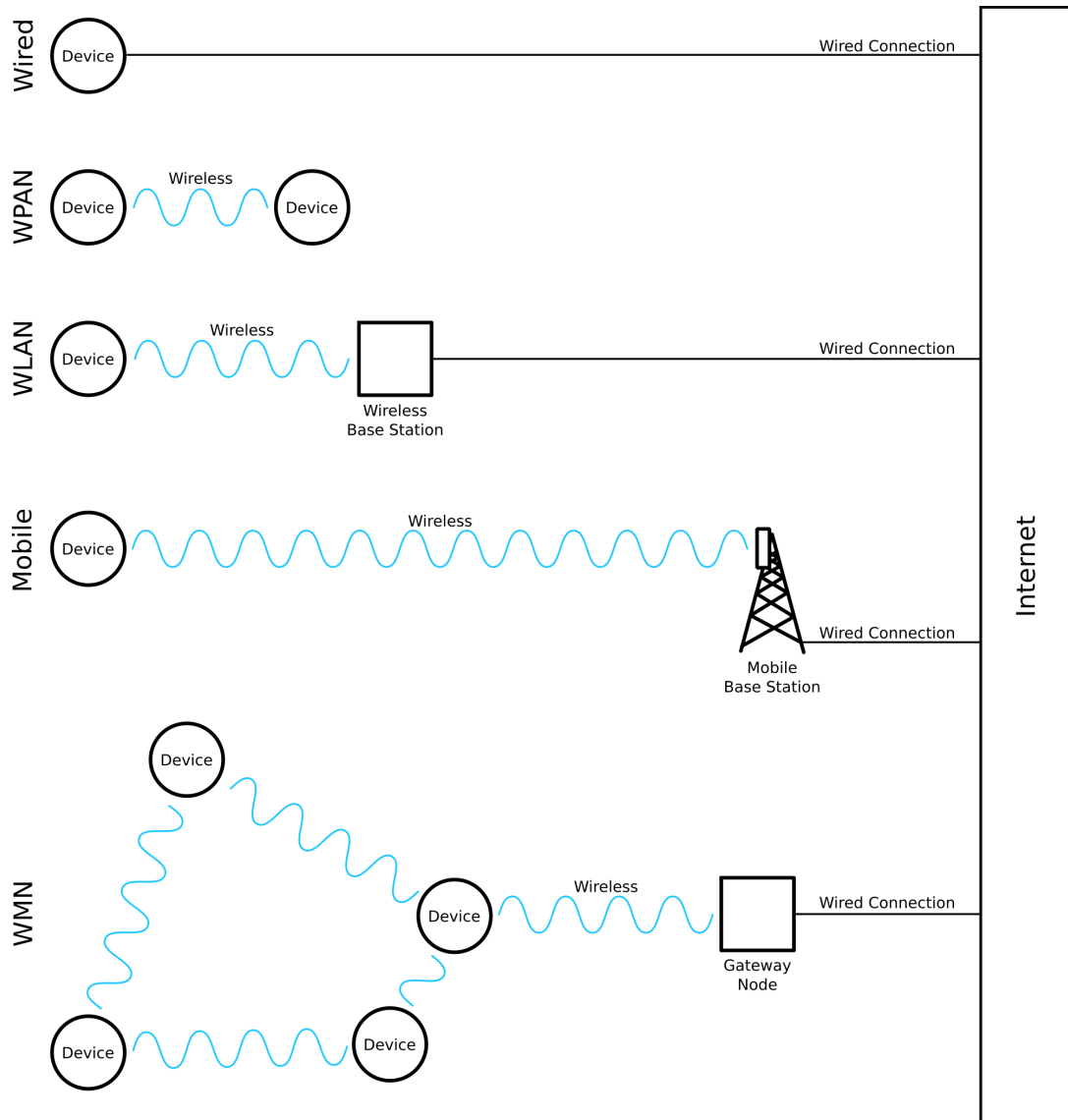


Figure 5.2. Different types of network connectivity.

The most traditional way to provide network connectivity is via a wired interface such as Ethernet or fiber-optics. A wired connection is especially suited for stationary systems which are rarely or never moved physically. However, wired connectivity may not always be available in remote or rural installation locations. Other options for connectivity are wireless radio interfaces such as Bluetooth, ZigBee, WiFi, LTE, 5G [12] or mesh networks [5].

Bluetooth is a wireless personal area network (WPAN) technology suitable for short range communications with a maximum range of 100 m. It uses a carrier frequency between

2.402 GHz and 2.480 GHz. Because the traditional versions of Bluetooth consume relatively large amounts of energy, Bluetooth low energy (BLE) has been introduced for low power systems such as IoT devices. The maximum data rate of Bluetooth version 5.0 is 2 Mbps. ZigBee is another WPAN technology suitable for IoT use cases with a maximum range of up to 100 m. It uses a 2.4 GHz, 868 MHz or 915 MHz carrier. The maximum data rate of ZigBee is 250 kbps. [12] Bluetooth and ZigBee do not offer Internet connectivity by default which means a gateway device is required if Internet connectivity is needed. These technologies are generally best suited for short range communication between devices.

WiFi is a wireless local area network (WLAN) which provides last mile Internet connectivity via a wireless base station. WiFi has a larger range and higher data rates compared to WPAN technologies. It uses a 2.4 GHz or 5 GHz carrier and modern versions can achieve data rates as high as 7 Gbps. [12] WiFi networks are suitable for installations where devices can be placed so that they are in fairly close proximity to a WiFi base station.

Another option for Internet connectivity are mobile networks such as LTE and 5G. These networks are characterized by good worldwide availability and large data transfer rates. LTE and 5G networks operate at various carrier frequencies and theoretically offer up to 1 Gbps and 20 Gbps data rates respectively. [12] Mobile networks are especially suited for devices in locations where no other network connectivity is available or for movable devices.

Wireless mesh networks (WMN) provide connectivity via a mesh of wireless nodes where network traffic is routed from node to another until an Internet gateway node is reached. Compared to WiFi networks where the physical link always exists directly between the wireless client and the base station, in mesh networks the link is split into multiple smaller hops between mesh nodes. The strengths of mesh networks are the possibility to cover a large area and no line of sight (LOS) connectivity. Weaknesses of mesh networks include their low maturity level and the need for a wired connection for the Internet gateway node [5]. WMNs may be a viable option when network connectivity is required for multiple physically separate systems which are still in fairly close proximity to each other but too far apart from a centralized Internet gateway.

5.4 User Interfaces

A human-machine interface (HMI) or a user interface is an integral part of most embedded systems. An HMI can mean many things. Although a traditional keyboard and mouse is also an HMI, such an interface is rarely used in an embedded system in practice. It is instead much more common to include HMI elements such as graphical displays, buttons or switches and other purpose-designed interactive components. [57] Nowadays,

even a mobile application connected to an embedded system via a cloud might be considered a form of HMI of the embedded system itself.

An HMI requires support from the underlying embedded hardware. For example, a display requires, in addition to the display hardware itself, at least a power supply and a graphics bus such as MIPI DSI, LVDS or HDMI. Especially the choice of graphics bus depends on the capabilities of the system processor, because not all processors support all graphics buses. The choice of display resolution is also dependent on the capabilities of the processor, because a high resolution such as 4K demands more processing power compared to a lower resolution. Even simpler user interfaces such as buttons and switches require some support from the processor, such as a simple communication bus like I2C or SPI, or general purpose I/O pins.

5.5 Payment Solutions

Payment of goods is a fundamental part of any public electric vehicle charger. After all, there is always a cost associated with electricity and a customer charging their electric vehicle must be able to pay for the transferred electric energy. The two main forms of payment at charge points are contract-based payment and pay as you go (PAYG). Contract-based payment refers to payment using mobile applications or RFID cards provided by charge point operators (CPO). PAYG payment, on the other hand, is typically based on credit cards and does not require a contract between the customer and CPO. While the PAYG method would be the most preferable for many customers, contract-based payment is the most common in practice. [30] The issue with contract-based payment is that it makes ad hoc charging difficult for customers who do not already have a contract with a particular CPO and taking an application into use forces the customer into a contract with the CPO. Such issues hinder the adoption of electric vehicles in general and therefore contradict the green transformation aspirations of many countries. The European union has taken action to solve these issues by creating the alternative fuels infrastructure (AFIR) regulation which mandates that all new electric vehicle charging points must implement payment via widely used payment methods such as payment cards starting from the 13th of April 2024. [60]

Complying with payment related regulations also poses requirements on electric vehicle charging hardware. Supporting contactless payment and payment cards as mandated by the AFIR regulation requires specialized hardware designed to deal with such payment methods. Although the AFIR regulation does not mandate that existing installations are equipped with payment terminals, all new installations must include them. Payment terminals must therefore be integrated into all new public chargers either as a builtin feature or as a 3rd party module starting from the 13th of April 2024.

5.6 Modular Connectivity

Implementing a complex embedded system necessarily means implementing interfaces between different parts of the system such as between the processor and its peripherals. These interfaces can consist of simple I/O, but they can also be more complex parallel or serial interfaces such as USB, SPI, I2C or Ethernet among many others. [40] Choosing interfaces between pre-made components such as integrated circuits (IC) of a system is primarily governed by what is available on the market, unless manufacturing application specific integrated circuits (ASIC) is viable. Therefore, low-level interface selection is mostly a matter of selecting compatible ICs.

While the low-level interfaces between ICs of an embedded system usually cannot be chosen completely freely, clustering and modularity can be used as discussed in chapter 3 to abstract low-level interfaces behind higher-level ones. This means that a product may use an arbitrary interface, such as a serial bus, between its modules, while the modules themselves contain other interfaces for internal communication. In practice this also means that each module connecting via the common interface must include a processor which handles intra-module communication and acts as a gateway on the module boundary. Such abstraction is perhaps more familiar from software design where abstractions and modularity are routinely combined to hide complexity in order to keep software more manageable [72]. Nevertheless, similar concepts can also be applied in hardware system design to achieve similar goals.

5.7 Safety Requirements

Electric vehicle charging deals with high voltage, current and power which means safety of charging systems is a number one priority. The safety of such systems is mainly governed by electrical safety standards and regulation as well as charging system standards. Electrical safety standards apply to all electrical devices and are not discussed in this thesis. Safety requirements of charging standards, on the other hand, are specific to electric vehicle charging devices. Because this thesis is mostly concerned with the architecture of embedded systems and not implementation details of any particular device, the safety aspects of electric vehicle charging systems are not covered in detail. Nevertheless, the safety of such systems should also be taken into account early in the architecture design stage.

6. EMBEDDED SYSTEM ARCHITECTURE FOR AN EV CHARGING SYSTEM PLATFORM

In this chapter, the process outlined in 3.7 is used to design a product family and a product platform for a public electric vehicle charging system. The functionality of such devices identified in chapter 5 is used as a basis for the CNs of the product family.

6.1 Methods

Architectures were designed using axiomatic design and design structure system in conjunction with the novel cluster utilization and cost analysis methods. The design was done in Microsoft Excel, because it provides a concise way of representing data in tabular and matrix forms as well as support for mathematical operations on real numbers and matrices. For calculations, the matrix forms of equations introduced in chapters 3 and 4 were used. Calculations were automated using Microsoft Excel formulas.

6.2 Architecture Design

As discussed in chapter 5, a public electric vehicle charging device must fulfill many CNs, some of which are more important to the customer and others more important to the CPO. Based on the common charger functionality identified in chapter 5, the following CNs were selected for this example.

- CN1:** Control the electric vehicle charging process.
- CN2a:** Compatible with CCS2 charging.
- CN2b:** Compatible with CHAdeMO charging.
- CN3:** Support payment with credit and debit cards.
- CN4:** Use a 24V input power supply.
- CN5a:** Include a visual user interface.
- CN5b:** Use a mobile application as a user interface.
- CN6:** Support IoT applications.

Customer needs and constraints are listed in *Table 1* and *Table 2* of appendix A. For this example, some CNs were considered optional. These CNs are marked with a and b symbols. In the case of CN2, they relate to the following customer groups.

- Group 1:** This group requires that the device is compatible with CCS2 charging only. (CN2a)
- Group 2:** This group requires that the device is compatible with CHAdeMO charging only. (CN2b)
- Group 3:** This group requires that the device is compatible with CCS2 and CHAdeMO charging. (CN2a & CN2b)

For CN5, they relate to the following customer groups.

- Group 1:** This group requires that the device includes a visual display where the charging status is visible as well as support for a mobile application. (CN5a & CN5b)
- Group 2:** This group requires that the charging event is controllable using a mobile application only. (CN5b)

With the CNs identified, the next step was to translate the CNs into an FR-DP hierarchy which satisfies the CNs. To this end, the zigzagging process of axiomatic design was used. The complete FR-DP hierarchy is included in *Table 3* of appendix A. A visual representation of the FR hierarchy is shown in figure 6.1. The matching DP hierarchy is shown in figure 6.2.

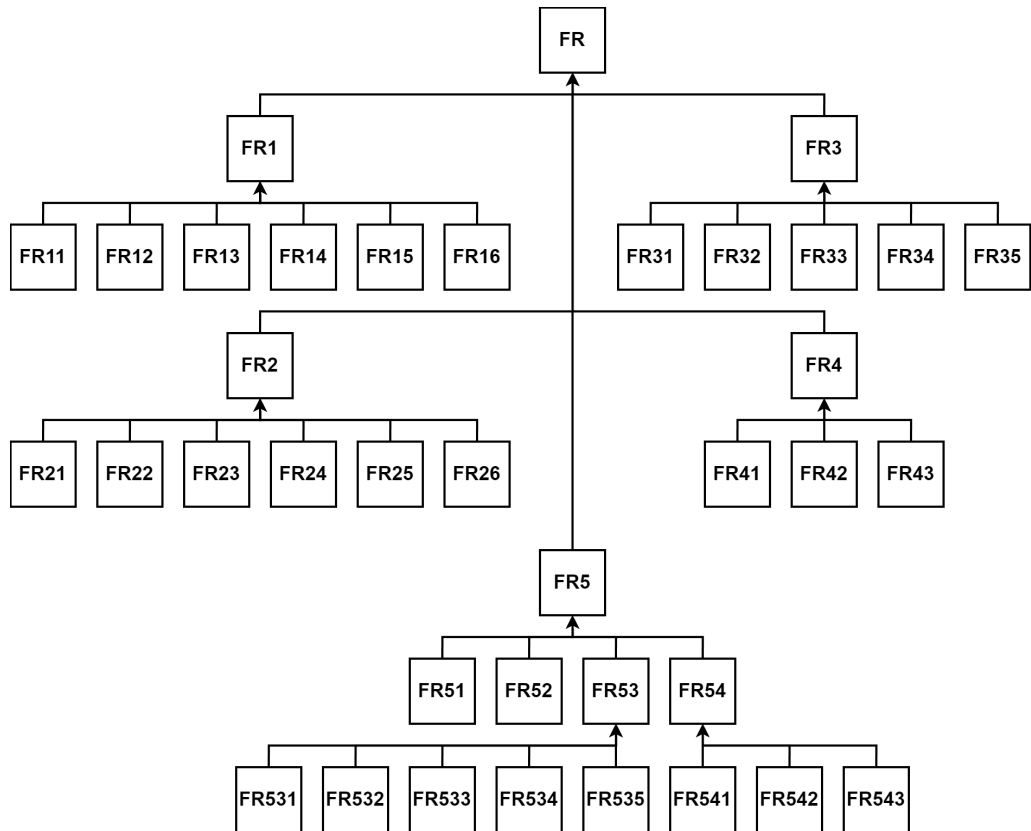


Figure 6.1. The FR hierarchy of the designed EV charging system architecture.

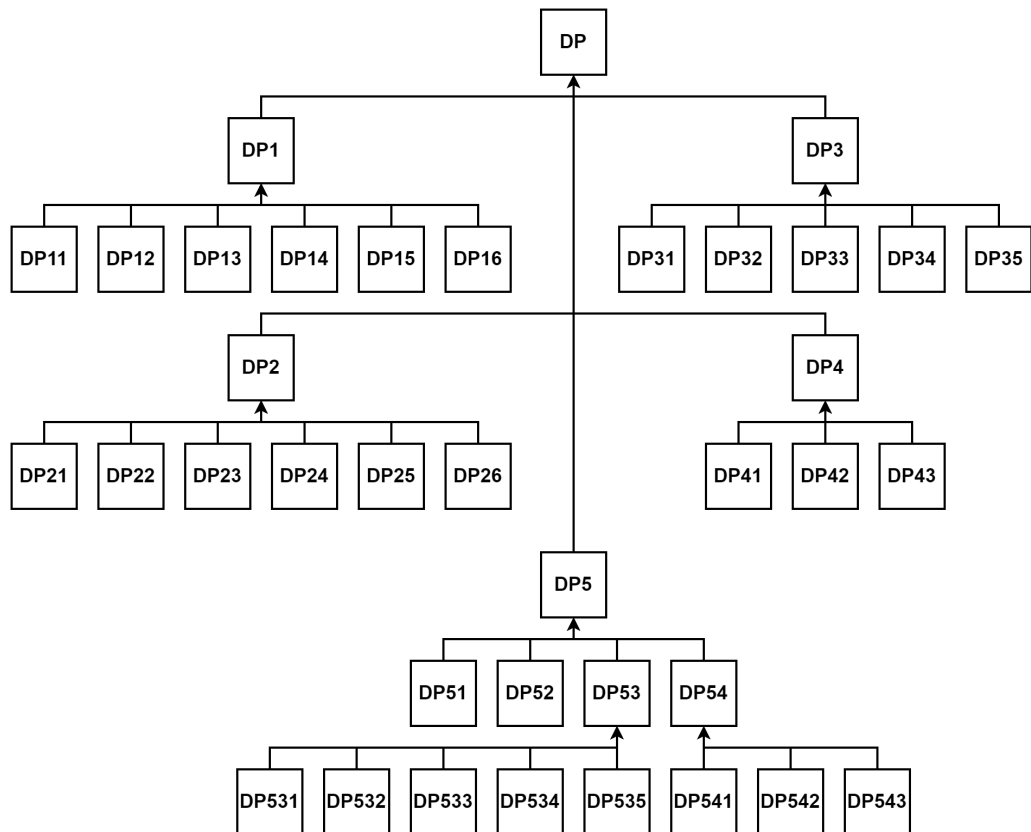


Figure 6.2. The DP hierarchy of the designed EV charging system architecture.

After the FR-DP hierarchy was created, the independence axiom was applied to check that the design was satisfactory, ie. that all design matrices were either uncoupled or decoupled. The results of this analysis are included in appendix B. All design matrices turned out to be triangular which means the design is decoupled, although some matrices required reordering of their rows and columns to arrive at triangular matrices. In the end, all matrices were reordered to be upper triangular to help in later analysis. After the design was deemed satisfactory in terms of the independence axiom, the composite design matrix **A** described in section 3.3 was constructed. This matrix is included in *Matrix 1* of appendix D. The matrix was constructed to take into account the functional dependencies on all hierarchical levels. In practice, this means that if FR_x depends on DP_x on a particular hierarchical level, then FR_x also depends on all DP_{xy} on the hierarchical level below DP_x . In the composite design matrix, the high-level functional dependencies can be seen as large blocks of X symbols on the off-diagonal positions of the matrix while the dependencies for leaf nodes are seen closer to the diagonal.

Next, a simple product family specification was formulated. The specification is included in *Table 1* of appendix C. The product family includes all possible variants which can be realized from the set of CNs. The total number of product variants is $2^3 = 8$, because three optional CNs were specified in appendix A and the rest are mandatory. A visual illustration of the product family is shown in figure 6.3. The product configuration matrix **C** introduced in section 3.3 was formulated based on the variant configuration table in appendix C by selecting the FRs corresponding to selected CNs for each variant on the rows of the matrix. The mapping between CNs and FRs was obtained from the FR-DP hierarchy table in appendix A. The final product configuration matrix is included in *Matrix 1* of appendix C.

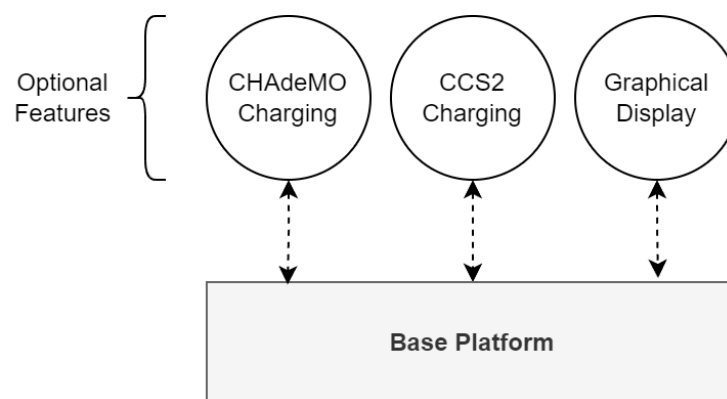


Figure 6.3. An illustration of the product family for the designed EV charging device.

The last step of the architecture design process was physical clustering. First, the composite design parameter DSM included in *Matrix 1* of appendix E was constructed. The DSM was constructed by taking the leaf-nodes of the DP hierarchy and assembling them on the rows and columns of the matrix. Different types of interactions between DPs were then identified by inspection. The interaction types identified in this design were:

- Data (D):** A data flow between DPs, for example via a serial data bus.
- Power (P):** An energy flow between DPs, for example via a power cable.
- Mechanical (M):** A mechanical interaction between DPs, for example a mounting interface.
- Diagonal (X):** Diagonal elements of the matrix were marked with X symbols to represent the interaction of a DP with itself.

Although there can be multiple types of interactions between two DPs, in this case only the most major interactions were recorded for each DP pair.

After the initial DP DSM was constructed, the architecture was optimized using the integration analysis methods introduced in chapter 3. The main optimization method was DSM clustering based on the previously recorded DP interactions, ie. the aim was to minimize interactions between clusters and consequently maximize interactions within clusters. The clustering was done by inspection. In order to get insight into the effects of different levels of physical modularity on cluster utilization and cost, five different clustering strategies were selected for analysis. These strategies are listed in table 6.1.

Table 6.1. *Physical clustering strategies.*

Option	Strategy	Results
A	DP hierarchy based modular	Appendix F
B	Constant cluster size based modular	Appendix G
C	DP interaction based modular	Appendix H
D	DP interaction based semi-modular	Appendix I
E	Completely integral	Appendix J

The calculations and results for each respective architecture option are included in the appendix indicated by the results column in table 6.1. The clustering strategies are ordered from roughly most modular to most integral, although option A is not completely modular in the sense that it does not use a one-to-one mapping between DPs and physical modules. Such a clustering was not analyzed because it would be impractical in the real world. The clustered DP DSMs are included in *Matrix 1* of each architecture appendix.

After the DP clustering was done, DSMs were constructed on the level of physical modules for each architecture option. These are included in *Matrix 2* of each architecture appendix. In the module DSMs, different types of physical interfaces were selected to satisfy the interactions between physical modules. In total, nine different physical interfaces were selected. These interface types were encoded with the letters A-I and their descriptions are included in table 6.2. The DSMs were constructed so that up to four interfaces can be recorded between each physical module.

Table 6.2. *Interface types between physical modules.*

Type	Description
A	CAN interface
B	UART interface
C	SPI interface
D	MIPI-DSI interface
E	5V power supply
F	3V3 power supply
G	PMIC power supply
H	24V power supply
I	Heatsink mount

The resulting five different physical architectures were analyzed and compared using the novel cluster utilization method introduced in section 3.3 and the novel cost analysis method introduced in chapter 4. In order to apply the cost analysis method, price data for DPs and interfaces was acquired from two different electronic component distributors' catalogs. This data is included in appendix L. Design parameter cost data is included in *Table 1* and interface cost data is included in *Table 2*. The price data was selected as an estimate only. While it does also give estimates of total product cost, the relative differences between costs using different architectures are the main point of interest. Therefore, the absolute accuracy of the cost estimates does not play a significant role in the analysis.

The design parameter and interface cost data can be used to estimate the DP cost and interface cost components of the total product cost equation introduced in chapter 4. The third cost component, the manufacturing cost, must be estimated based on other factors. In this analysis, the manufacturing cost was estimated at 30% of DP cost for the module which includes the multicore SoC, ie. DP51, and 20% of DP cost for other modules. This estimate is based on two assumptions.

1. The module which includes DP51 requires more intricate manufacturing technology because the multicore SoC is a complex part. Therefore, a higher manufacturing cost is justified for this module.
2. Increasing the number and complexity of DPs included in a module increases the manufacturing cost of the module because more labor and more intricate manufacturing technology is required. Therefore, the manufacturing cost of a module may be estimated as a linear function of total module DP cost.

Again, the absolute values of manufacturing costs do not play a significant role in this analysis, because the main point of interest is in analyzing the cost differences between different architectures.

6.3 Cluster Utilization Analysis

The novel cluster utilization analysis method introduced in section 3.3 was applied on all physical architecture options A-E. The results are included in the appendix of each respective architecture option. In practice, the analysis method was applied by calculating the product assembly matrix CA (*Matrix 3*) and applying the same reordering and clustering onto its columns as used in the respective DP DSM (*Matrix 1*). Because the entries of the design matrix A were originally entered as X symbols, they were interpreted as binary values for calculating CA , ie. each cell with an X symbol was interpreted as 1 and empty cells were interpreted as zeros. Next, the per cluster utilizations were calculated for each variant using equation 3.16 and these individual results were assembled into matrix form (*Matrix 4*) as shown in equation 3.18. The binary cluster utilization matrix (*Matrix 5*) was also calculated based on equation 3.19. Finally, the average cluster utilization vector (*Vector 1*) was calculated according to equation 3.20.

The calculated average cluster utilizations were collected into table 6.3 for comparison. Note that the cluster numbers do not correspond to the same clusters across all architecture options because each architecture is clustered differently.

Table 6.3. Average cluster utilization comparison.

Option	Cluster							
	1	2	3	4	5	5	7	8
A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
B	1.00	0.67	0.80	0.70	1.00	1.00	N/A	N/A
C	1.00	1.00	1.00	1.00	1.00	0.67	N/A	N/A
D	0.67	1.00	0.79	N/A	N/A	N/A	N/A	N/A
E	0.75	N/A	N/A	N/A	N/A	N/A	N/A	N/A

According to this comparison, option A is best in terms of cluster utilization. This result was expected, because in option A, DPs are clustered based on the DP hierarchy which also matches the FR hierarchy. This means that removing FRs from the design maps directly into removing DPs from the system and therefore no unused DPs remain and all modules are completely utilized. However, this does not necessarily mean that option A is optimal in all other aspects such as cost. Options A-E, on the other hand, all contain some non-unity cluster utilizations. This indicates that some physical modules in the system are not completely utilized across all product variants. This directly translates to increased DP cost, because at least some product variants contain unused DPs.

6.4 Modularity Cost Analysis

The novel cost analysis method introduced in chapter 4 was applied on all physical architecture options A-E. The results are included in the appendix of each respective option. While the cluster utilizations of different options can be used to get insight into how well a product family utilizes the DPs of a product platform, it is also important to consider how the cluster utilizations affect the total cost of product variants. While an architecture may be optimal in terms of cluster utilization, it may still not be optimal in terms of cost, because an extremely modular or poorly designed system may increase cost due to additional interfaces. The novel cost analysis method also takes the interface cost into account to give a more complete picture of the total cost effects of modularity.

The cost analysis method was applied by first arranging the individual DP costs into a vector (*Vector 2*) and applying the same reordering and clustering onto its rows as applied onto the DP DSM (*Matrix 1*). Where DP merging possibilities were identified while clustering, DP costs were merged by setting the cost of all but one of the merged DPs to zero. This is also visualized in the appendices. Next, the per-cluster DP costs were calculated by calculating the sums of DP costs included in each cluster according to equation 4.9. These costs were assembled into the vector \bar{p}_{DM} (*Vector 3*). Next, the

module manufacturing costs were estimated based on the module DP costs by assuming 30% of DP costs as the manufacturing cost for the module containing DP51 and 20% of DP costs for other modules as discussed earlier. The manufacturing costs were then assembled into the vector \bar{P}_{MM} (*Vector 3*) according to equation 4.8. Next, the interface costs were calculated based on the physical module DSM (*Matrix 2*). For each module-to-module interface cell, the total cost associated with the cell was calculated by adding up the costs of individual interfaces within that cell. A cluster interface cost matrix was constructed from these results (*Matrix 6*). The variant interface cost vector \bar{P}_{IM} (*Vector 3*) was then calculated by taking the sum of each row of the cluster interface cost matrix according to equation 4.10. With all of the individual cost components calculated, the sum vector $\bar{P}_{MM} + \bar{P}_{DM} + \bar{P}_{IM}$ (*Vector 3*) was calculated. Finally, the variant cost vector \bar{P}_{modular} (*Vector 4*) was calculated according to equation 4.7. All of the calculated vectors and matrices are included in appendices F-J for each respective architecture option.

The cost-effects of different architecture options were then compared according to equation 4.11. It is noteworthy that representing the cost effects of different architecture options as a function of modularity is not a trivial problem, because the scale used to measure modularity has to be defined first. It could be argued that, for example, the number of physical modules could be used as a scale to describe the level of modularity. However, such a scale is not necessarily a good choice, because two different architectures with an equal number of modules may have completely different module boundaries. In an extreme case, an architecture with many physical modules may be clustered so that interdependencies between physical modules effectively make the system completely integral. The number of physical modules is therefore not a good way to describe modularity. To avoid having to define a scale of modularity only for representing analysis results, the architectures A-E were roughly ordered based on intuition as E, D, C, B, A with E being the most integral and A being the most modular. Such an ordering is acceptable in this case because it only affects how the analysis results are visually presented and not the results themselves. When analyzing the effects of changing the level of modularity, it was natural to consider the cost difference when moving from the most integral architecture, ie. option E, to any other option. The results of this comparison are included in *Table 1* and *Figure 1-3* of appendix K. In addition to absolute cost differences, relative cost differences were calculated to give better insight into the effects of different levels of modularity. These relative results are included in figure 6.4 for all product variants and all architecture change scenarios.

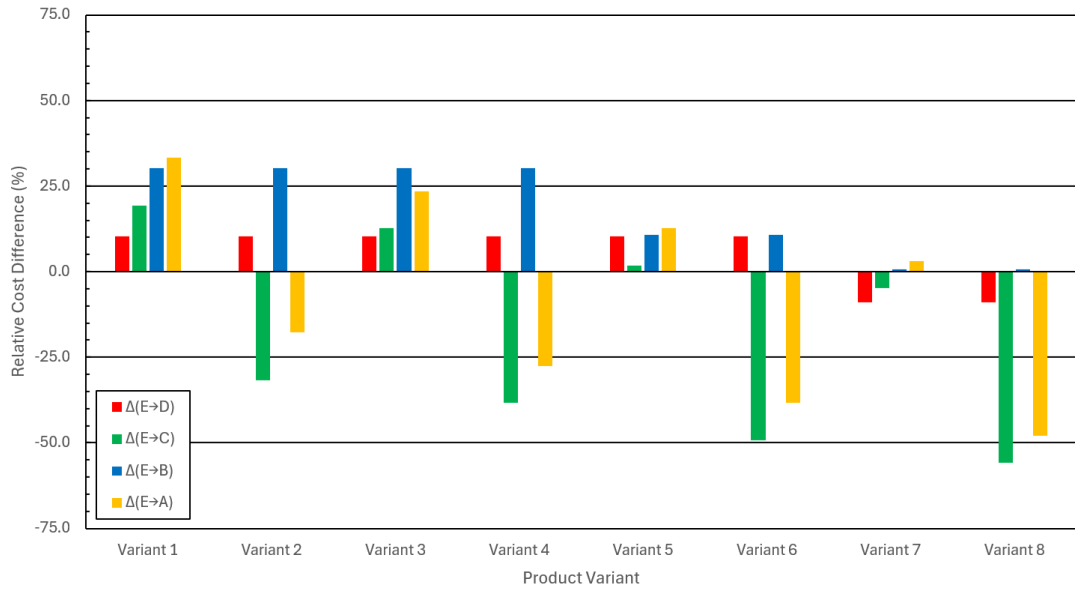


Figure 6.4. Relative cost differences per product variant with different architecture change scenarios.

From this figure, it is evident that the physical architecture of a product platform may have a considerable positive or negative effect on product cost. The cost-effects of modularity are also variant-dependent. While a particular architecture option may decrease the cost of one variant, it may increase the cost of another variant. For example, in figure 6.4, moving from architecture E to architecture D increases the cost of variants 1-6 and decreases the cost of variants 7-8. In some cases, a particular architecture option may even increase the cost of all variants such as when moving from architecture E to B in figure 6.4. Selecting the most optimal physical architecture therefore requires knowledge about the variants included in the product family. In addition, if some variants are considered more important than others, for example due to larger sales volumes, it may be acceptable to decrease the cost of the more important variants with the expense of making less important variants more costly.

To give an overview of the results, the minimum, maximum and average relative cost differences across all product variants were calculated. These results were collected into table 6.4.

Table 6.4. Relative cost differences with different architecture change scenarios.

Type	Relative Cost Difference			
	$\Delta(E \rightarrow D)$	$\Delta(E \rightarrow C)$	$\Delta(E \rightarrow B)$	$\Delta(E \rightarrow A)$
Minimum	-8.9%	-55.8%	+0.6%	-48.0%
Maximum	+10.4%	+19.3%	+30.2%	+33.2%
Average	+5.6%	-18.2%	+17.9%	-7.4%

According to the results, moving from architecture E to C lowers product costs by 18.2% on average. Moving from architecture option E to A, on the other hand, lowers product costs by 7.4% on average. The other analyzed change scenarios result in a cost increase on average.

In the product platform analyzed in this thesis, the physical display module is particularly expensive and variants where it is not needed show the most potential for cost optimization by removing it. Other physical modules are not as expensive, and therefore their effect on total variant cost is not as great. It can be concluded, in general, that extra attention should be paid to the clustering of the most expensive DPs, because they also offer the most potential for cost optimization. Clustering less costly DPs into their own modules, on the other hand, may increase total cost due to greater interface costs. These conclusions can also be made by analyzing the cost difference equation 4.11.

The accuracy of the cost estimates given by the cost analysis method used in this thesis is heavily dependent on the accuracy of the DP cost data. In general, the cost estimates should be made as accurate as possible with the information available at any given time during the design process. In the conceptual design stage, accurate cost information is usually not available and therefore designers must rely on DP cost estimates. As the design process progresses and final design choices are made, the results of the cost analysis method also converge to more accurate variant cost vectors.

6.5 Results

Based on the independence axiom, all architectural options A-E are satisfactory because all of them are based on the same satisfactory design characterized by triangular design matrices. However, the physical architectures are different in each case. An optimal physical architecture is characterized by loosely interconnected modules. Architecture option B can therefore be considered suboptimal because its module DSM (*Matrix 2*) indicates it contains a complex set of interfaces. The other architectures also contain many interfaces, but they are mostly related to processor communication or power supplies. In general, the interfaces in other options are better organized than in option B.

Based on the average cluster utilization results from table 6.3, architecture option A is the most optimal and C is the second most optimal. In these options, most DP clusters are completely utilized. Based on the cost analysis results from table 6.4, architecture option C is the most optimal and A is the second most optimal. On average, these options reduce the costs of product variants compared to the integral architecture option. Architecture options B and D, on the other hand, are not satisfactory because they exhibit poor cluster utilizations and they also increase costs. The integral architecture option E was used as the reference architecture. Because architectures A and C are more optimal based on the cluster utilization and cost analyses, it can be considered non-satisfactory as well.

There is a conflict between the options recommended by the cluster utilization analysis and the cost analysis. According to cluster utilization, option A should be selected, and according to cost, option C should be selected. While cluster utilization is also important in terms of material usage and consequently, for example, environmental concerns, cost is likely to have the largest role in the decision making in a business. Therefore, option C may be selected as the most optimal of the five physical architectures compared in this thesis. Because cluster utilization is also used as a parameter in the cost analysis method, the most cost-optimized architecture also optimizes cluster utilization to some degree. Option C is also one of the satisfactory options according to physical integration analysis. Therefore, it can be concluded that the integration analysis, cluster utilization analysis and cost analysis methods all give similar indicators for architecture selection. However, each of these indicators gives a different view into architecture selection, and joint application of them is therefore necessary.

7. EFFECTS, RISKS AND ADVANTAGES OF STRUCTURED DESIGN METHODS

To take a structured design method into use is a big leap for an R&D organization and such a transition needs solid justification. In this chapter, I summarize the effects, risks and advantages of structured design methods and modularity based on the findings of this thesis.

7.1 Evaluating Product Architectures

Increasing modularity offers many advantages in product design. However, the advantages of modularity are mostly obtained when an optimal level of modularity is used, and too little or too much modularity may even make a design suboptimal in various ways. Therefore, modularity of a product should not be considered to be of intrinsic value. Instead, modularity should be treated as a byproduct of applying a structured design method, such as axiomatic design, in order to achieve a satisfactory design. Whether the satisfactory design is integral, modular or something in between, is mostly a curiosity. What is important is that the design is suitable for its intended use case and intended production paradigm.

According to the findings of this thesis, a set of criteria for gauging whether a design is satisfactory may be formulated:

Criterion A: *(Based on ch. 3)* Does the design fulfill its set of functional requirements, formulated from customer needs?

Criterion B: *(Based on ch. 3)* Does the design obey the independence axiom, ie. is it uncoupled or decoupled?

Criterion C: *(Based on ch. 3)* Does the design obey the information axiom, ie. is its information content minimized?

Criterion D: *(Based on ch. 2)* Does the chosen level of physical modularity fit either mass production, mass customization or mass individualization?

Criterion E: *(Based on ch. 4)* Is the level of physical modularity optimal based on a cluster utilization analysis and modularity versus cost analysis?

7.2 Risks of Structured Design Methods

Applying a structured design method such as axiomatic design comes with its own set of risks. This section is a summary of the risks identified in this thesis.

7.2.1 Cost of Modularity

As discussed in chapter 4, too little or too much modularity can increase the cost of a product. It is therefore crucial to analyze the cost-effects of modularity already when designing the physical architecture of a product. To this end, the novel cost analysis method introduced in chapter 4 may be used.

7.2.2 Total Complexity

As discussed in chapter 2.2, a side-effect of increased physical modularity is necessarily an increase in interfaces and an increase in complexity. To avoid unnecessary complexity, a good balance between physical modularity, complexity and other factors summarized in this chapter should be pursued.

7.2.3 Organizational Fit

Applying the design methods introduced in chapter 2 is a matter of organizing the product design process within a design organization. Aligning the design methods and proposed product architecture to the design organization is therefore of crucial importance to keep R&D efficiency at a good level [53].

7.2.4 Lack of Expertise

While chapter 3 serves as a good introduction to axiomatic design, applying axiomatic design in practice requires a deep understanding of the fundamentals of the method to guarantee success. Consequently, a lack of expertise can lead to a misapplication of the method and cause a failure to satisfy customer needs. Training is therefore required if axiomatic design is taken into use in an R&D organization.

7.2.5 Amount of Work

Suh's axiomatic design has been accepted as one good solution by a large part of the research community, but its application in practical design problems is still quite labor intensive and understanding its nuances requires a lot of theoretical knowledge. In order to reduce the required amount of labor, computer aided design software is needed. An

example of such software is Acclaro DFSS [51], but software was not the topic of this thesis. Therefore, further research is needed on the availability and practicality of design software.

7.3 Advantages of Structured Design Methods

Applying a structured design method such as axiomatic design also offers many advantages. This section is a summary of the advantages identified in this thesis.

7.3.1 Project Management

Modularity makes design tasks more decoupled, gives designers more freedom within modules and simplifies complex products. This may ease the management of product design. [46] For example, the design of different modules of a system can be handled by separate small teams which are also managed separately. This may be easier than managing a large team designing the entire product.

7.3.2 Theoretical Proof

As discussed in chapter 2, using a structured method such as axiomatic design has been mathematically proven to yield well designed products. As long as a theoretically sound design method is applied correctly, the results of a design project are guaranteed to be satisfactory. This gives product designers confidence during the design process and improves the odds of success.

7.3.3 Product Quality

As discussed in chapter 3, axiomatic design can improve the quality of products due to its information axiom which tries to maximize the probability of satisfying functional requirements. This means that products designed with the information axiom in mind have a higher probability of operating as intended compared to products designed without the information axiom.

7.3.4 Product Maintenance

A physically modular system simplifies maintenance, because products can be maintained and repaired on the physical module level. For example, it may be more cost effective to repair or swap out a single broken physical module of a product instead of swapping out the entire product. [9] Implementing physical modularity in a product architecture is therefore useful in terms of product maintenance.

7.3.5 Product Configuration

As discussed in chapter 3.3, product architectures can be time-variant. For example, reconfiguring a product by adding or removing functional requirements causes time-variance. If existing functional requirements are removed from a product family architecture, reconfiguration is only a matter of removing physical modules provided that the physical architecture allows it. This means that implementing a modular product family architecture enables product configuration.

8. CONCLUSIONS

In this thesis, the principles of system architecting, modular system design and architecture optimization were researched. Traditionally, systems have been designed by applying practical expert knowledge acquired from successful projects. While such knowledge is a necessary condition of successful system design, basing a design solely on practical and empirical knowledge does not necessarily guarantee the satisfaction of the needs of all stakeholders, such as all customers and business parties. In general, the problem of creating system designs based on the needs of various stakeholders is called multiple-criteria decision making. There exists a vast spectrum of methods to solve different kinds of MCDM problems. Some of these methods are based on heuristic practices which have been formulated as design frameworks, while others are based on more rigorous theoretical bases. Heuristic methods include Pugh controlled convergence and analytic hierarchy process which are based on comparison of concepts. More structured methods include the Taguchi method, design structure system, axiomatic design and modular function deployment.

According to the findings of this thesis, architectures can be categorized in two ways: Based on modularity and based on the chosen production paradigm. These two ways are also related to each other, because modular architectures are inherently most suited for mass customization or mass individualization, whereas integral architectures are most suited for mass production. As a consequence, defining the required production paradigm and product variant portfolio is of critical importance before the physical architecture of a system is designed. Optimizing the architecture based on the required product variants is only possible in early design stages. Defining product variants after integration analysis, on the other hand, may result in a significant production cost increase.

While any of the introduced design methods can be used to design modular architectures, axiomatic design and design structure system were selected for further analysis in this thesis to outline a complete method for designing systems beginning all the way from customer needs and resulting in a product family architecture. Two novel methods for the optimization of physical architectures were also developed based on the theoretical basis provided by axiomatic design. These methods can be used to optimize the utilization of design parameters in physical architectures and to optimize the cost-effects of modularity. An exemplary product family architecture for a public electric vehicle charging system

was developed using the introduced design methods. In order to base the design on realistic customer needs, common properties and functionality of such systems were first identified. Some of the most important customer needs were support for applications processing, networking and standardized charging protocols. The two optimization methods developed in the thesis were used to optimize the physical architecture of the platform. The methods provided clear indicators on how optimal each architecture option was. Based on these indicators, selecting the best option was straightforward.

Even though design problems validating axiomatic design are numerous, some researchers are still skeptical. The method has been criticized, for example, because it may impose a preference structure on the designer. A limitation of the developed cost-analysis method, on the other hand, is that it only takes into account the immediate production costs of products and factors such as supply chain costs, final assembly costs, logistics costs or product lifetime costs are dismissed. Structured design methods, in general, may also be difficult to apply in practice especially without a good level of theoretical understanding of their application. In general, even though structured design methods and modularity offer many advantages, implementing them in an R&D organization requires careful planning in order to guarantee success.

REFERENCES

- [1] International Energy Agency. *CO₂ emissions in 2023*. 2024. URL: <https://iea.blob.core.windows.net/assets/33e2badc-b839-4c18-84ce-f6387b3c008f/CO2Emissionsin2023.pdf>.
- [2] International Energy Agency. *Global EV Outlook 2023*. 2023. URL: <https://iea.blob.core.windows.net/assets/dacf14d2-eabc-498a-8263-9f97fd5dc327/GEVO2023.pdf>.
- [3] M. A. Mellinger et. al. "Anxiety vs reality – Sufficiency of battery electric vehicle range in Switzerland and Finland". In: *Transportation research. Part D, Transport and environment* 65 (2018), pp. 101–115.
- [4] C. Renzi et. al. "Selecting alternatives in the conceptual design phase: an application of Fuzzy-AHP and Pugh's Controlled Convergence". In: *International Journal on Interactive Design and Manufacturing* 9.1 (2013), pp. 1–17.
- [5] D. Benyamina et. al. "Wireless Mesh Networks Design — A Survey". In: *IEEE Communications Surveys & Tutorials* 14.2 (2012), pp. 299–310.
- [6] D. D. Frey et. al. "The Pugh Controlled Convergence method: model-based evaluation and implications for design theory". In: *Research in Engineering Design* 20.1 (2009), pp. 41–58.
- [7] D. Tang et. al. "Design as Integration of Axiomatic Design and Design Structure Matrix". In: *Robotics and Computer-Integrated Manufacturing* 25.3 (2008), pp. 610–619.
- [8] G. Mouli et. al. "Implementation of Dynamic Charging and V2G using Chademo and CCS/Combo DC charging standard". In: *Proceedings of the 2016 IEEE Transportation Electrification Conference and Expo (ITEC)* (2016), pp. 1–6.
- [9] G. Yicong et. al. "Product Modular Design Incorporating Preventive Maintenance Issues". In: *Chinese Journal of Mechanical Engineering* 29.2 (2016), pp. 406–420.
- [10] H. ElMaraghy et. al. "Product Variety Management". In: *CIRP Annals* 62.2 (2013), pp. 629–652.
- [11] H. Skirde et. al. "Measuring the Cost Effects of Modular Product Architectures - A Conceptual Approach". In: *International Journal of Innovation and Technology Management* 13.4 (2016).
- [12] J. Ding et. al. "IoT Connectivity Technologies and Applications: Survey". In: *IEEE Access* 8 (2020), pp. 67646–67673.
- [13] J. Hackl et. al. "Impact of Modularity Decisions on a Firm's Economic Objectives". In: *Journal of Mechanical Design* 142.4 (2020).

- [14] J. Hou et. al. "Does industrial green transformation successfully facilitate a decrease in carbon intensity in China? An environmental regulation perspective". In: *Journal of Cleaner Production* 184 (2018), pp. 1060–1071.
- [15] M. Bounou et. al. "Improving the quality of an optimal power flow solution by Taguchi method". In: *Electrical Power & Energy Systems* 17.2 (1995), pp. 113–118.
- [16] O. Rincon-Guevara et. al. "Taxonomy for Classifying Products on the Customization Spectrum". In: *Proceedings of the 2018 International Conference on Production and Operations Management Society* (2018).
- [17] P. Reid et. al. "Electric Vehicle Conductive Charge Couplers". In: *Proceedings of the 2014 IEEE 60th Holm Conference on Electrical Contacts (Holm)* (2014), pp. 1–7.
- [18] R. Andersen et. al. "Investigating the applicability of modular function deployment in the process industry". In: *Proceedings of the 54th CIRP Conference on Manufacturing Systems* (2021), pp. 659–664.
- [19] R. Collin et. al. "Advanced Electric Vehicle Fast-Charging Technologies". In: *Energies* 12.10 (2019), pp. 1839–1865.
- [20] R. Duray et. al. "Approaches to mass customization: configurations and empirical validation". In: *Journal of Operations Management* 18.6 (2000), pp. 605–625.
- [21] S. Wolf et. al. "The European Green Deal - More Than Climate Neutrality". In: *Intereconomics* 56.2 (2021), pp. 99–107.
- [22] T. D. Brunoe et. al. "Modular Design Method for Reconfigurable Manufacturing Systems". In: *Proceedings of the 54th CIRP Conference on Manufacturing Systems* (2021), pp. 1275–1279.
- [23] T. Hohnen et. al. "Cost-Effects of Product Modularity - An Approach to Describe Manufacturing Costs as a Function of Modularity". In: *Proceedings of the 23rd CIRP Design Conference* (2013), pp. 745–754.
- [24] A. Alptekinoğlu, C. Corbett. "Mass Customization vs. Mass Production: Variety and Price Competition". In: *Manufacturing & Service Operations Management* 10.2 (2008), pp. 204–217.
- [25] Society of Automotive Engineers. *SAE J3271, Megawatt Charging System for Electric Vehicles*.
- [26] Society of Automotive Engineers. *SAE J3400, North American Charging System (NACS) for Electric Vehicles*.
- [27] E. M. Benavides. *Advanced engineering design, An integrated approach*. Woodhead Publishing Limited, 2012.
- [28] D. A. Broniatowski. "Do design decisions depend on "dictators"?" In: *Research in Engineering Design* 29.1 (2018), pp. 67–85.
- [29] D. M. Buede. *The Engineering Design of Systems: Models and Methods*. John Wiley & Sons, 2016.

- [30] K. Chamberlain, S. Al-Majeed. "Standardisation of UK Electric Vehicle Charging Protocol, Payment and Charge Point Connection". In: *World Electric Vehicle Journal* 12.2 (2021), pp. 63–95.
- [31] CharIN. *CharIN Whitepaper, Megawatt Charging System (MCS)*. 2022. URL: https://www.charin.global/media/pages/technology/knowledge-base/c708ba3361-1670238823/whitepaper_megawatt_charging_system_1.0.pdf (visited on 09/21/2024).
- [32] X. Cheng, C. Chen. "Applying Independence Axiom and Design Structure Matrix to Product Module Division". In: *Proceedings of the 2010 International Conference on Mechanic Automation and Control Engineering* (2010).
- [33] D. Chick. "The Time Value of Project Change". In: *Cost Engineering* 41.6 (1999), pp. 27–31.
- [34] International Electrotechnical Commission. *IEC 61851-1:2019, Electric vehicle conductive charging system. Part 1: General requirements*. 2019.
- [35] International Electrotechnical Commission. *IEC 61851-23:2023, Electric vehicle conductive charging system - Part 23: DC electric vehicle supply equipment*. 2023.
- [36] International Electrotechnical Commission. *IEC 61851-24:2014, Electric vehicle conductive charging system - Part 24: Digital communication between a d.c. EV charging station and an electric vehicle for control of d.c. charging*. 2014.
- [37] International Electrotechnical Commission. *IEC 62196-2:2022, Plugs, socket-outlets, vehicle connectors and vehicle inlets - Conductive charging of electric vehicles - Part 2: Dimensional compatibility requirements for AC pin and contact-tube accessories*. 2022.
- [38] International Electrotechnical Commission. *IEC 62196-3:2022, Plugs, socket-outlets, vehicle connectors and vehicle inlets - Conductive charging of electric vehicles - Part 3: Dimensional compatibility requirements for DC and AC/DC pin and contact-tube vehicle couplers*. 2022.
- [39] International Electrotechnical Commission. *IEC 63379, Vehicle connector, vehicle inlet and cable assembly for Megawatt DC charging*.
- [40] L. Das. *Embedded Systems: An Integrated Approach*. Pearson, 2012.
- [41] J. S. Dyer. "Remarks on the Analytic Hierarchy Process". In: *Management Science* 26.3 (1990), pp. 249–258.
- [42] S. Eppinger, T. Browning. *Design Structure Matrix Methods and Applications, 1st edition*. MIT Press, 2012.
- [43] Farnell. *EV charging standards: Ensuring compatibility and safety in the charging process*. URL: <https://fi.farnell.com/ev-charging-standards-ensuring-compatibility-and-safety-in-the-charging-process-trc-ar> (visited on 07/12/2024).
- [44] G. A. Hazelrigg. "Letter to the editor: The Pugh controlled convergence method: model-based evaluation and implications for design theory". In: *Research in Engineering Design* 21.3 (2010), pp. 143–144.

- [45] W. Ho. “Integrated analytic hierarchy process and its applications – A literature review”. In: *European Journal of Operational Research* 186.1 (2008), pp. 211–228.
- [46] K. Hölttä-Otto, O. de Weck. “Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints”. In: *Concurrent Engineering: Research and Applications* 15.2 (2007), pp. 113–126.
- [47] E. Hong, G. Park. “Decomposition Process of Engineering Systems Using Axiomatic Design and Design Structure Matrix”. In: *Proceedings of the Fifth International Conference on Axiomatic Design* (2009).
- [48] E. Hong, G. Park. “Modular Design Method Using the Independence Axiom and Design Structure Matrix in the Conceptual and Detailed Design Stage”. In: *Proceedings of the Sixth International Conference on Axiomatic Design* (2011).
- [49] S. J. Hu. “Evolving Paradigms of Manufacturing: From Mass Production to Mass Customization and Personalization”. In: *Procedia CIRP* 7 (2013), pp. 3–8.
- [50] Tesla Inc. *Opening the North American Charging Standard*. 2022. URL: <https://www.tesla.com/blog/opening-north-american-charging-standard> (visited on 07/11/2024).
- [51] Functional Specs Inc. *Acclaro DFSS™ Features*. URL: <https://www.axiomaticdesign.com/software/acclaro-dfss/> (visited on 09/14/2024).
- [52] J. Jiao, M. Tseng. “Fundamentals of product family architecture”. In: *Journal of Manufacturing Technology Management* 11.7 (2000), pp. 469–483.
- [53] T. Juuti. *Design Management of Products with Variability and Commonality*. Tampereen Teknillinen Yliopisto, 2008.
- [54] G. Kumar, S. Mikkili. “Advancements in EV international standards: Charging, safety and grid integration with challenges and impacts”. In: *International Journal of Green Energy* 21.12 (2024), pp. 2672–2698.
- [55] M. W. Lange, A. Imsdahl. “Modular Function Deployment: Using Module Drivers to Impart Strategies to a Product Architecture”. In: *Advances in Product Family and Product Platform Design, Methods & Applications* (2014), pp. 91–118.
- [56] E. Lavi, Y. Reich. “Cross-disciplinary system value overview towards value-oriented design”. In: *Research in Engineering Design* 35.1 (2023), pp. 1–20.
- [57] P. Marwedel. *Embedded System Design - Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things - 4th Edition*. Springer, 2021.
- [58] A. T. Olewnik, K. Lewis. “On Validating Engineering Design Decision Support Tools”. In: *Concurrent Engineering* 13.2 (2005), pp. 85–166.
- [59] G. Park. *Analytic Methods for Design Practice, 1st edition*. Springer, 2007.
- [60] The European Parliament, The Council of the European Union. *Regulation (EU) 2023/1804 of the European Parliament and of the Council of 13 September 2023 on the deployment of alternative fuels infrastructure*. 2023.

- [61] S. Pashaei, J. Olhager. "The impact of global operations on product architecture: an exploratory study". In: *International Journal of Operations & Production Management* 37.10 (2017), pp. 1304–1326.
- [62] T. L. Saaty, L. G. Vargas. *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*. Springer, 2012.
- [63] C. Shannon. "A Mathematican Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.
- [64] A. Stevenson, ed. *Oxford Dictionary of English*. Oxford University Press, 2010.
- [65] D. Steward. "The Design Structure System: A Method for Managing the Design of Complex Systems". In: *IEEE Transactions on Engineering Management* 28.3 (1981), pp. 71–74.
- [66] E. Stiassnie, M. Shpitalni. "Axiomatic Modular System Design for Service-Oriented Products". In: *Proceedings of The Sixth International Conference on Axiomatic Design* (2011).
- [67] N. P. Suh. "Axiomatic Design Theory for Systems". In: *Research in Engineering Design* 10.4 (1998), pp. 189–209.
- [68] N. P. Suh. "Designing-in of Quality Through Axiomatic Design". In: *IEEE Transactions on Reliability* 44.2 (1995).
- [69] N. P. Suh. *The Principles of Design*. Oxford University Press, 1990.
- [70] J. Tiihonen, A. Felfernig. "An introduction to personalization and mass customization". In: *Journal of Intelligent Information Systems* 49.1 (2016), pp. 1–7.
- [71] M. Tseng, J. Jiao. "Design for Mass Customization". In: *CIRP Annals* 45.1 (1996), pp. 153–156.
- [72] H. Zhu. *Software Design Methodology: From Principles to Architectural Styles*. Elsevier, 2005.

Title	FR-DP Hierarchy for an EV Charger
Revision	A
Created By	Eero Talus
Date	22/09/2024

Table 1: Identified Customer Needs

CN	Description	Notes
CN1	Control the electric vehicle charging process.	
CN2a	Compatible with CCS2 charging.	Optional
CN2b	Compatible with CHAdeMO charging.	Optional
CN3	Support payment with credit and debit cards.	
CN4	Use a 24V input power supply.	
CN5a	Include a visual user interface.	Optional
CN5b	Use a mobile application as a user interface.	
CN6	Support IoT applications.	

Table 2: Identified Constraints

CS	Description	Notes
CS1	Keep operating temperatures low.	

Table 3: FR-DP Hierarchy

Satisfies	FR	Description	DP	Description
CN1, CN2a, CN4	FR1	Communicate using the CCS2 protocol	DP1	CCS2 PHY
	FR11	Handle communication state machine	DP11	CPU
	FR12	Use low-level communication	DP12	Digital IO
	FR13	Use high-level communication	DP13	PLC PHY
	FR14	Use a common physical interface	DP14	4-pin Molex Micro-Fit connector
	FR15	Use a 24V to 5V power supply	DP15	Buck converter
CN1, CN2b, CN4	FR2	Communicate using the CHAdeMO protocol	DP2	2-pin Molex Micro-Fit connector
	FR21	Handle communication state machine	DP21	CHAdEMO PHY
	FR22	Use low-level communication	DP22	CPU
	FR23	Use high-level communication	DP23	Digital IO
	FR24	Use a common physical interface	DP24	CAN PHY
	FR25	Use a 24V to 5V power supply	DP25	8-pin Molex Micro-Fit connector
CN3, CN4	FR26	Use a common input power connector	DP26	Buck converter
	FR3	Accept payment using a payment card	DP3	2-pin Molex Micro-Fit connector
	FR31	Accept non-contact payments	DP31	Payment terminal
	FR32	Handle payment processing	DP32	RFID interface
	FR33	Have a connection to the internet	DP33	Secure CPU
	FR34	Use a 24V to 5V power supply	DP34	Ethernet hardware
CN5a, CN4	FR35	Use a common input power connector	DP35	Buck converter
	FR4	Display graphical information	DP4	2-pin Molex Micro-Fit connector
	FR41	Display high resolution graphics	DP41	Display
	FR42	Mount on an enclosure	DP42	1920x1080 LCD panel unit with 24V power input
	FR43	Be resistant to scratches and hits	DP43	Mechanical mounting frame Tempered cover glass

	FR5	Run control and application software	DP5	Control system
CN1, CN6	FR51 FR52 FR53	Run complex simultaneous processes Have a connection to the internet Use a 24V to 5V, 3.3V, 1.8V and 0.75V power supply	DP51 DP52 DP53	Multicore SoC Ethernet hardware Multiple regulators
CN4	FR531 FR532 FR533 FR534 FR535	Convert 24V to 5V Convert 5V to 3.3V Convert 5V to 1.8V Convert 5V to 0.75V Use a common input power connector	DP531 DP532 DP533 DP534 DP535	Buck converter PMIC buck converter PMIC buck converter PMIC buck converter 2-pin Molex Micro-Fit connector
CS1	FR54 FR541 FR542 FR543	Cool the CPU Dissipate heat Transfer heat from the SoC to the heatsink Mount on top of the SoC	DP54 DP541 DP542 DP543	Passive cooling solution Metal heatsink Thermal interface material CPU mounting solution

Title	Design Matrices for an EV Charger
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Hierarchy level 1 Design Matrix.

Decoupled design

	DP1	DP2	DP3	DP4	DP5
FR1	X				X
FR2		X			X
FR3			X		X
FR4				X	X
FR5					X

Matrix 2: Hierarchy level 2 Design Matrix for FR1.

Decoupled design

	DP11	DP12	DP13	DP14	DP15	DP16
FR11	X	X	X			X
FR12		X	X			
FR13			X	X		
FR14				X		
FR15					X	X
FR16						X

Matrix 3: Hierarchy level 2 Design Matrix for FR2.

Decoupled design

	DP21	DP22	DP23	DP24	DP25	DP26
FR21	X	X	X			X
FR22		X		X		
FR23			X	X		
FR24				X		
FR25					X	X
FR26						X

Matrix 4: Hierarchy level 2 Design Matrix for FR3.

Decoupled design

	DP31	DP32	DP33	DP34	DP35
FR31	X			X	X
FR32		X	X	X	
FR33			X	X	
FR34				X	X
FR35					X

Matrix 5: Hierarchy level 2 Design Matrix for FR5.

Decoupled design

	DP41	DP42	DP43
FR41	X		
FR42	X	X	
FR43	X		X

Reorder

	DP43	DP42	DP41
FR43	X		X
FR42		X	X
FR41			X

Matrix 6: Hierarchy level 2 Design Matrix for FR5.

Decoupled design

	DP51	DP52	DP53	DP54
FR51	X		X	X
FR52		X	X	
FR53			X	
FR54				X

Matrix 7: Hierarchy level 3 Design Matrix for FR53.

Decoupled design

	DP531	DP532	DP533	DP534	DP535
FR531	X				X
FR532	X	X			
FR533	X		X		
FR534	X			X	
FR535					X

Reorder

	DP532	DP534	DP533	DP531	DP535
FR532	X				X
FR534		X			X
FR533			X	X	
FR531				X	X
FR535					X

Matrix 8: Hierarchy level 2 Design Matrix for FR54.

Decoupled design

	DP541	DP542	DP543
FR541	X		
FR542	X	X	X
FR543			X

Reorder

	DP542	DP543	DP541
FR542	X	X	X
FR543		X	
FR541			X

Title	Composite DP DSM for an EV Charger
Revision	A
Created By	Eero Talus
Date	22/09/2024

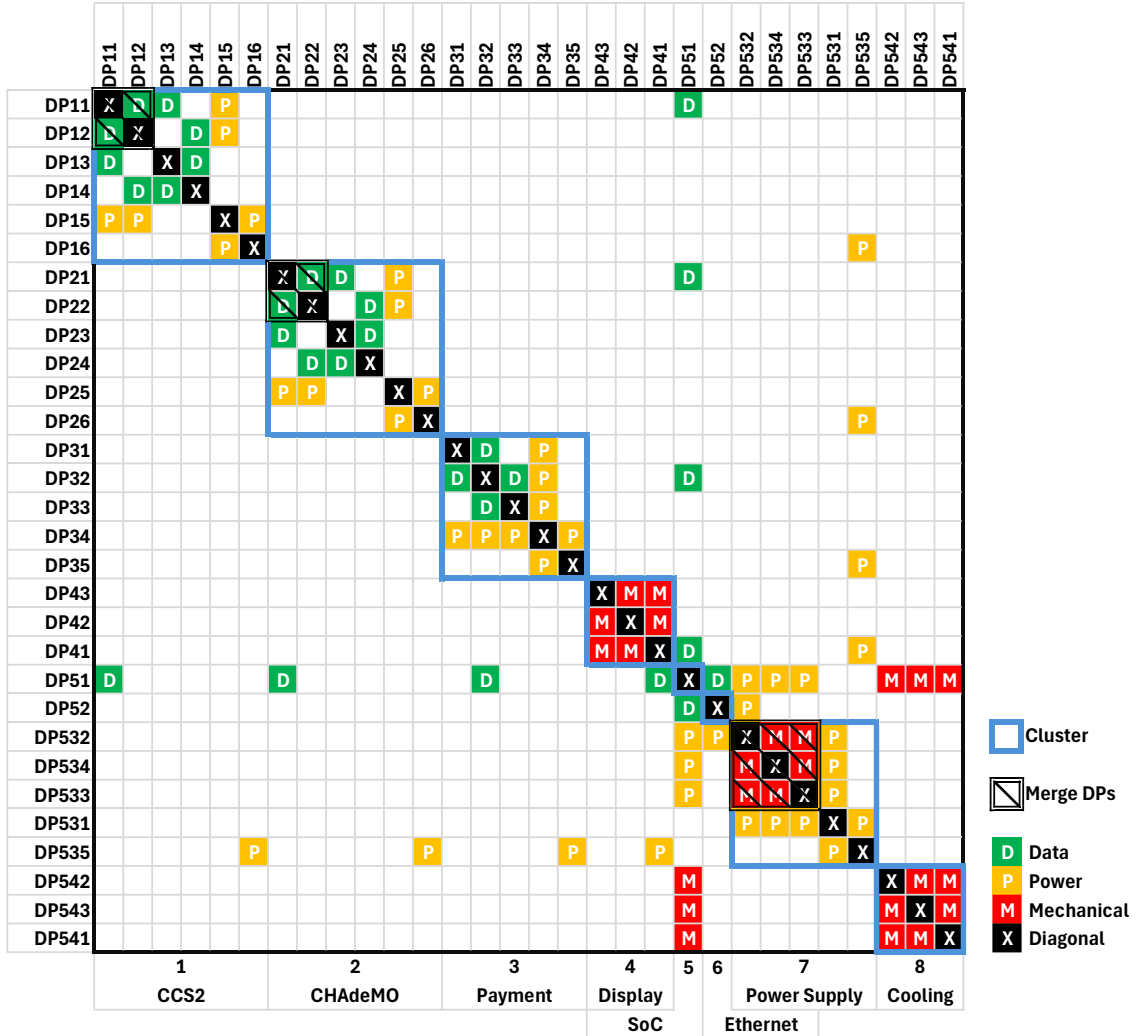
Matrix 1: Composite Design Parameter DSM.

	DP11	DP12	DP13	DP14	DP15	DP16	DP21	DP22	DP23	DP24	DP25	DP26	DP31	DP32	DP33	DP34	DP35	DP43	DP42	DP41	DP51	DP52	DP532	DP534	DP533	DP531	DP535	DP542	DP543	DP541		
DP11	X	D	D		P																D											
DP12	D	X		D	P																											
DP13	D		X	D																												
DP14		D	D	X																												
DP15	P	P			X	P																										
DP16					P	X																								P		
DP21							X	D	D		P											D										
DP22							D	X		D	P																					
DP23							D		X	D																						
DP24								D	D	X																						
DP25							P	P			X	P																				
DP26											P	X																			P	
DP31													X	D		P																
DP32													D	X	D	P						D										
DP33													D	X	P																	
DP34													P	P	P	X	P															
DP35																P	X														P	
DP43																		X	M	M												
DP42																		M	X	M												
DP41																		M	M	X	D										P	
DP51	D						D														D	X	D	P	P	P				M	M	M
DP52																						D	X	P								
DP532																						P	P	X	M	M	P					
DP534																						P	M	X	M	P						
DP533																						P	M	M	X	P						
DP531																							P	P	P	X	P					
DP535						P																					P	X				
DP542																						M							X	M	M	
DP543																						M							M	X	M	
DP541																						M							M	M	X	

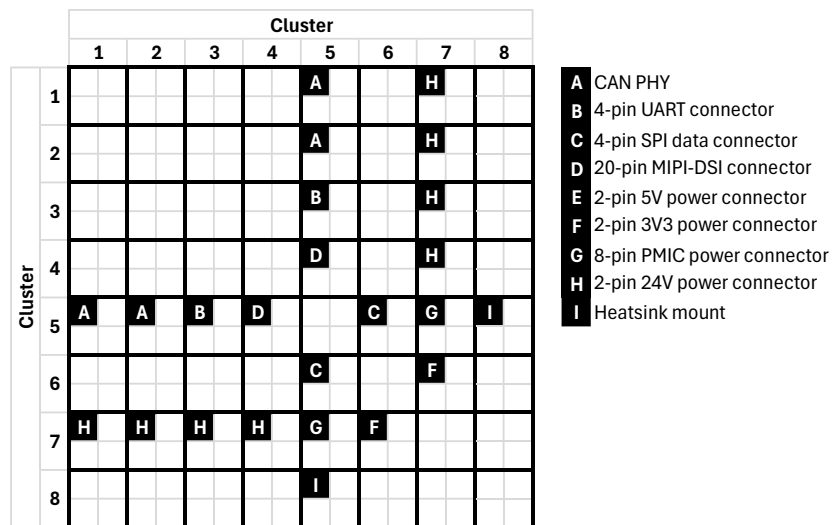
D Data
P Power
M Mechanical
X Diagonal

Title	Naive Clustering Based on DP Hierarchy (Option A)
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Clustered Design Parameter DSM.



Matrix 2: Module DSM.



Vector 2: Individual DP costs.

Cluster	DP	Cost	Status
1	DP11	2.81	Merged
	DP12	0.00	
	DP13	20.67	
	DP14	2.57	
	DP15	3.85	
	DP16	1.82	
2	DP21	2.81	Merged
	DP22	0.00	
	DP23	0.43	
	DP24	3.00	
	DP25	3.85	
	DP26	1.82	
3	DP31	4.86	
	DP32	2.81	
	DP33	5.20	
	DP34	3.85	
	DP35	1.82	
4	DP43	19.50	
	DP42	19.50	
	DP41	52.00	
5	DP51	13.50	
6	DP52	5.20	
7	DP532	3.20	Merged
	DP534	0.00	
	DP533	0.00	
	DP531	3.85	
	DP535	1.82	
8	DP542	0.71	
	DP543	0.00	
	DP541	2.14	

Vector 3: P_{DM} , P_{MM}^* and P_{IM} and their sum vector.

	P_{DM}	P_{MM}^*	P_{IM}	Σ
Cluster 1	31.71	6.34	7.14	45.20
Cluster 2	11.91	2.38	7.14	21.43
Cluster 3	18.54	3.71	6.71	28.96
Cluster 4	91.00	18.20	3.07	112.27
Cluster 5	13.50	4.05	22.40	39.95
Cluster 6	5.20	1.04	6.71	12.95
Cluster 7	8.86	1.77	18.68	29.31
Cluster 8	2.85	0.57	0.00	3.42

* Manufacturing cost is estimated at 30% of DP costs for the cluster containing the SoC and 20% for others.

Vector 4: Variant cost vector.

	P_{modular}
Variant 1	293.50
Variant 2	181.22
Variant 3	272.06
Variant 4	159.79
Variant 5	248.30
Variant 6	136.02
Variant 7	226.86
Variant 8	114.59

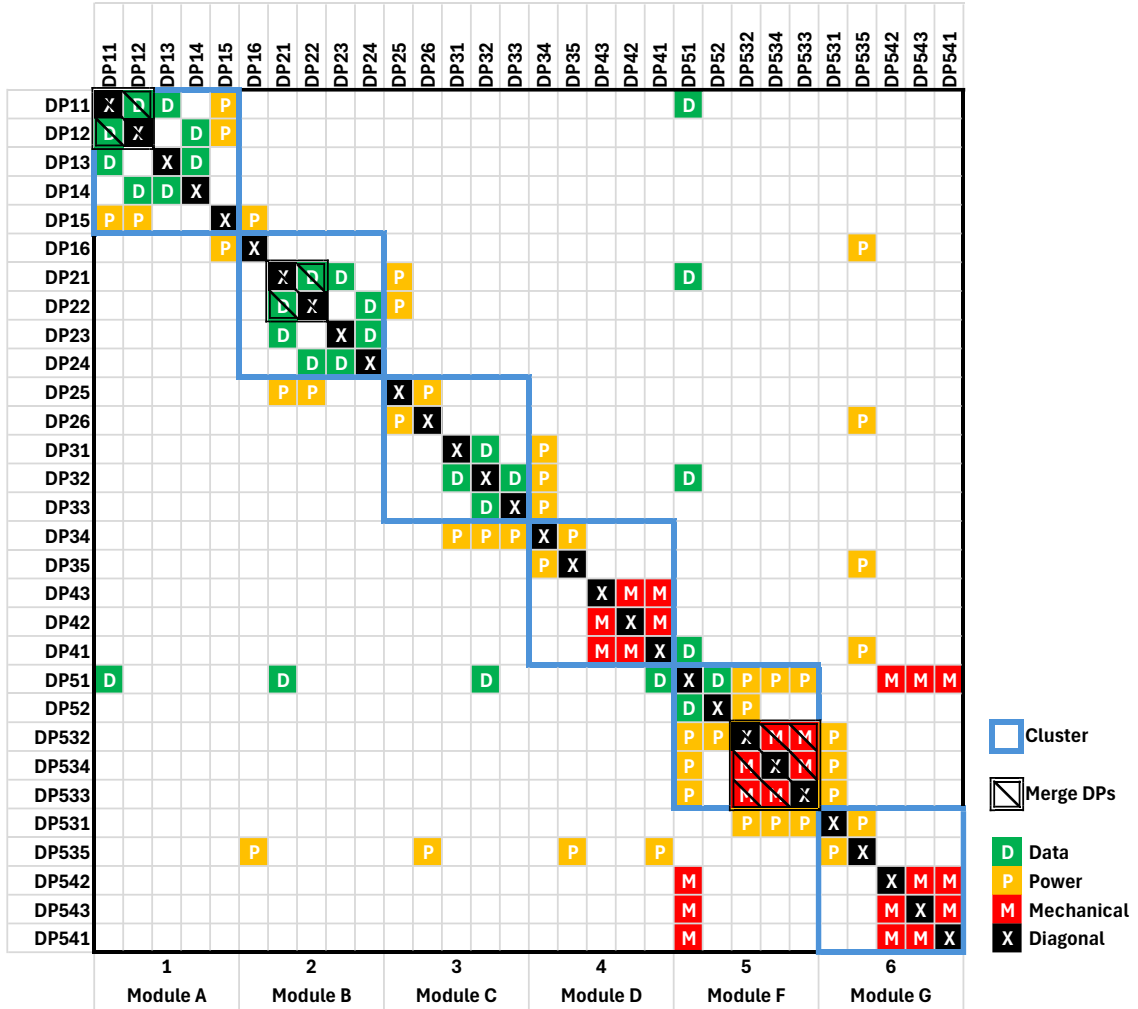
Matrix 6: Total cluster interface cost matrix.

		Cluster							
		1	2	3	4	5	6	7	8
Cluster	1	0.00	0.00	0.00	0.00	4.44	0.00	2.71	0.00
	2	0.00	0.00	0.00	0.00	4.44	0.00	2.71	0.00
	3	0.00	0.00	0.00	0.00	4.01	0.00	2.71	0.00
	4	0.00	0.00	0.00	0.00	0.37	0.00	2.71	0.00
	5	4.44	4.44	4.01	0.37	0.00	4.01	5.15	0.00
	6	0.00	0.00	0.00	0.00	4.01	0.00	2.71	0.00
	7	2.71	2.71	2.71	2.71	5.15	2.71	0.00	0.00
	8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

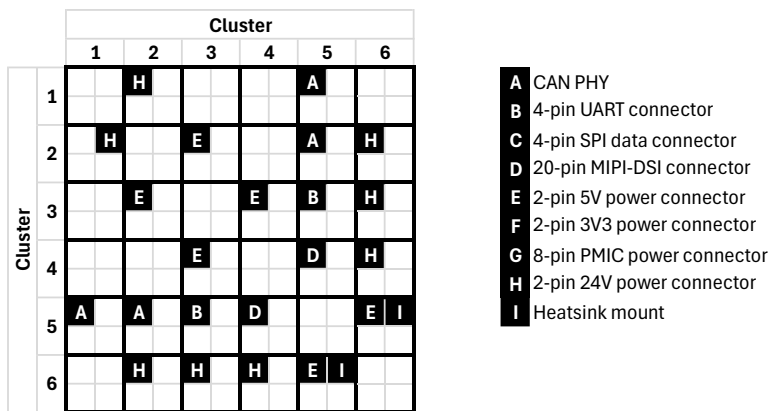
* Total cluster interface cost is the sum of the costs of individual interfaces defined in Matrix 2.

Title	Naive Clustering Based on a Constant Cluster Size (Option B)
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Clustered Design Parameter DSM.



Matrix 2: Module DSM.



Matrix 3: Clustered product assembly matrix CA.

	DP11	DP12	DP13	DP14	DP15	DP16	DP21	DP22	DP23	DP24	DP25	DP26	DP31	DP32	DP33	DP34	DP35	DP43	DP42	DP41	DP51	DP52	DP532	DP534	DP533	DP531	DP535	DP542	DP543	DP541
Variant 1	1	2	2	3	2	2	1	2	2	3	2	2	1	1	2	4	3	1	1	3	21	21	23	23	23	26	24	22	23	23
Variant 2	1	2	2	3	2	2	1	2	2	3	2	2	1	1	2	4	3	0	0	0	18	18	20	20	20	23	21	19	20	20
Variant 3	1	2	2	3	2	2	0	0	0	0	0	0	1	1	2	4	3	1	1	3	15	15	17	17	17	20	18	16	17	17
Variant 4	1	2	2	3	2	2	0	0	0	0	0	0	1	1	2	4	3	0	0	0	12	12	14	14	14	17	15	13	14	14
Variant 5	0	0	0	0	0	0	1	2	2	3	2	2	1	1	2	4	3	1	1	3	15	15	17	17	17	20	18	16	17	17
Variant 6	0	0	0	0	0	0	1	2	2	3	2	2	1	1	2	4	3	0	0	0	12	12	14	14	14	17	15	13	14	14
Variant 7	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	4	3	1	1	3	9	9	11	11	11	14	12	10	11	11
Variant 8	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	4	3	0	0	0	6	6	8	8	8	11	9	7	8	8

Matrix 4: Cluster utilization matrix H.

	Cluster					
	1	2	3	4	5	6
Variant 1	1.00	1.00	1.00	1.00	1.00	1.00
Variant 2	1.00	1.00	1.00	0.40	1.00	1.00
Variant 3	1.00	0.20	0.60	1.00	1.00	1.00
Variant 4	1.00	0.20	0.60	0.40	1.00	1.00
Variant 5	0.00	0.80	1.00	1.00	1.00	1.00
Variant 6	0.00	0.80	1.00	0.40	1.00	1.00
Variant 7	0.00	0.00	0.60	1.00	1.00	1.00
Variant 8	0.00	0.00	0.60	0.40	1.00	1.00

Matrix 5: Binary cluster utilization matrix H_b.

	Cluster					
	1	2	3	4	5	6
Variant 1	1	1	1	1	1	1
Variant 2	1	1	1	1	1	1
Variant 3	1	1	1	1	1	1
Variant 4	1	1	1	1	1	1
Variant 5	0	1	1	1	1	1
Variant 6	0	1	1	1	1	1
Variant 7	0	0	1	1	1	1
Variant 8	0	0	1	1	1	1

Vector 1: Average cluster utilization vector.

	Cluster					
	1	2	3	4	5	6
Average excl. unused	1.00	0.67	0.80	0.70	1.00	1.00

Vector 3: Individual DP costs.

Cluster	DP	Cost	Status
1	DP11	2.81	Merged
	DP12	0.00	
	DP13	20.67	
	DP14	2.57	
	DP15	3.85	
2	DP16	1.82	
	DP21	2.81	Merged
	DP22	0.00	
	DP23	0.43	
DP24	3.00		
3	DP25	3.85	
	DP26	1.82	
	DP31	4.86	
	DP32	2.81	
	DP33	5.20	
4	DP34	3.85	
	DP35	1.82	
	DP43	19.50	
	DP42	19.50	
	DP41	52.00	
5	DP51	13.50	
	DP52	5.20	
	DP532	3.20	Merged
	DP534	0.00	
DP533	0.00		
6	DP531	3.85	
	DP535	1.82	
	DP542	0.71	
	DP543	0.00	
DP541	2.14		

Vector 3: P_{DM}, P_{MM} and P_{IM} and their sum vector.

	P _{DM}	P _{MM} *	P _{IM}	Σ
Cluster 1	29.90	5.98	7.14	43.02
Cluster 2	8.06	1.61	12.56	22.23
Cluster 3	18.54	3.71	12.12	34.37
Cluster 4	96.66	19.33	5.78	121.78
Cluster 5	21.89	6.57	15.95	44.41
Cluster 6	8.52	1.70	10.83	21.05

* Manufacturing cost is estimated at 30% of DP costs for the cluster containing the SoC and 20% for others.

Vector 4: Variant cost vector.

	P _{modular}
Variant 1	286.86
Variant 2	286.86
Variant 3	286.86
Variant 4	286.86
Variant 5	243.84
Variant 6	243.84
Variant 7	221.60
Variant 8	221.60

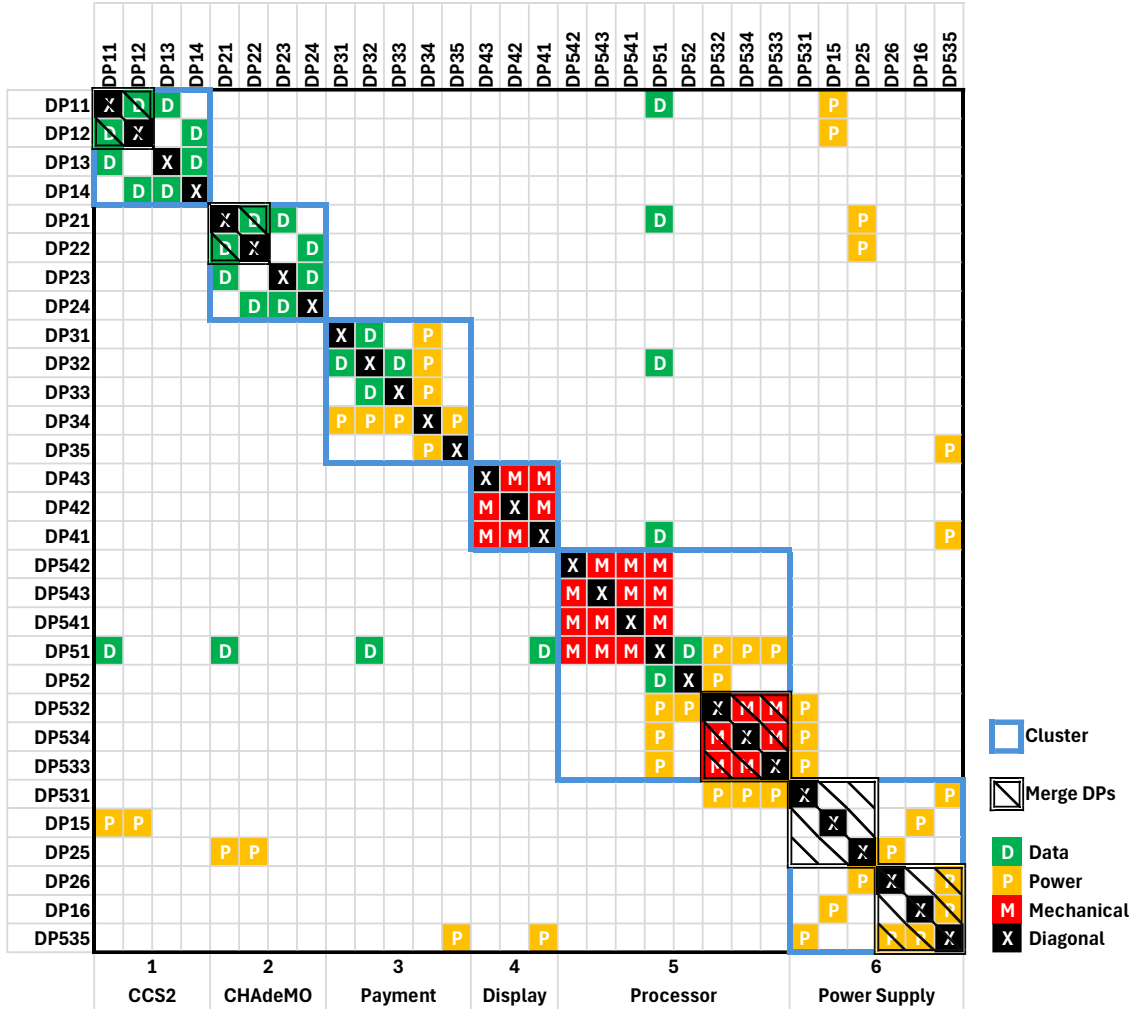
Matrix 6: Total cluster interface cost matrix.

		Cluster					
		1	2	3	4	5	6
Cluster	1	0.00	2.71	0.00	0.00	4.44	0.00
	2	2.71	0.00	2.71	0.00	4.44	2.71
	3	0.00	2.71	0.00	2.71	4.01	2.71
	4	0.00	0.00	2.71	0.00	0.37	2.71
	5	4.44	4.44	4.01	0.37	0.00	2.71
	6	0.00	2.71	2.71	2.71	2.71	0.00

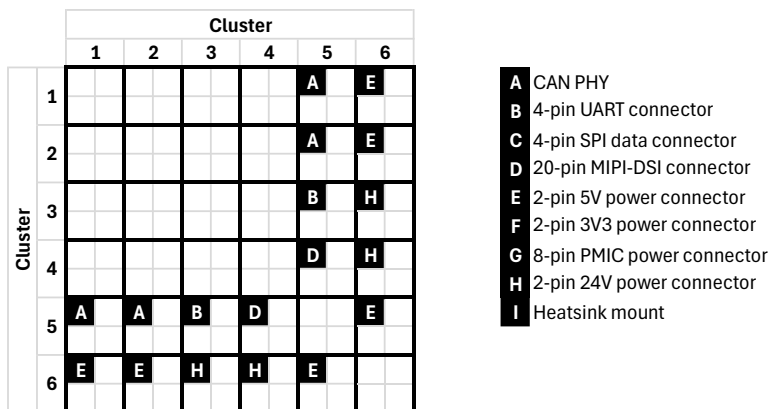
* Total cluster interface cost is the sum of the costs of individual interfaces defined in Matrix 2.

Title	Modular Clustering Based on DP Interactions (Option C)
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Clustered Design Parameter DSM.



Matrix 2: Module DSM.



Vector 2: Individual DP costs.

Cluster	DP	Cost	Status	
1	DP11	2.81	Merged	
	DP12	0.00		
	DP13	20.67		
	DP14	2.57		
2	DP21	2.81	Merged	
	DP22	0.00		
	DP23	0.43		
	DP24	3.00		
3	DP31	4.86		
	DP32	2.81		
	DP33	5.20		
	DP34	3.85		
	DP35	1.82		
4	DP43	19.50		
	DP42	19.50		
	DP41	52.00		
5	DP542	0.71		
	DP543	0.00		
	DP541	2.14		
	DP51	13.50		
	DP52	5.20		
	DP532	3.20		Merged
	DP534	0.00		
DP533	0.00			
6	DP531	3.85	Merged	
	DP15	0.00		
	DP25	0.00		
	DP26	1.82	Merged	
	DP16	0.00		
DP535	0.00			

Vector 3: P_{DM}, P_{MM} and P_{IM} and their sum vector.

	P _{DM}	P _{MM} *	P _{IM}	Σ
Cluster 1	26.05	5.21	7.14	38.40
Cluster 2	6.25	1.25	7.14	14.64
Cluster 3	18.54	3.71	6.71	28.96
Cluster 4	91.00	18.20	3.07	112.27
Cluster 5	24.75	7.42	15.95	48.12
Cluster 6	5.66	1.13	13.53	20.33

* Manufacturing cost is estimated at 30% of DP costs for the cluster containing the SoC and 20% for others.

Vector 4: Variant cost vector.

	P _{modular}
Variant 1	262.72
Variant 2	150.45
Variant 3	248.08
Variant 4	135.81
Variant 5	224.32
Variant 6	112.04
Variant 7	209.68
Variant 8	97.41

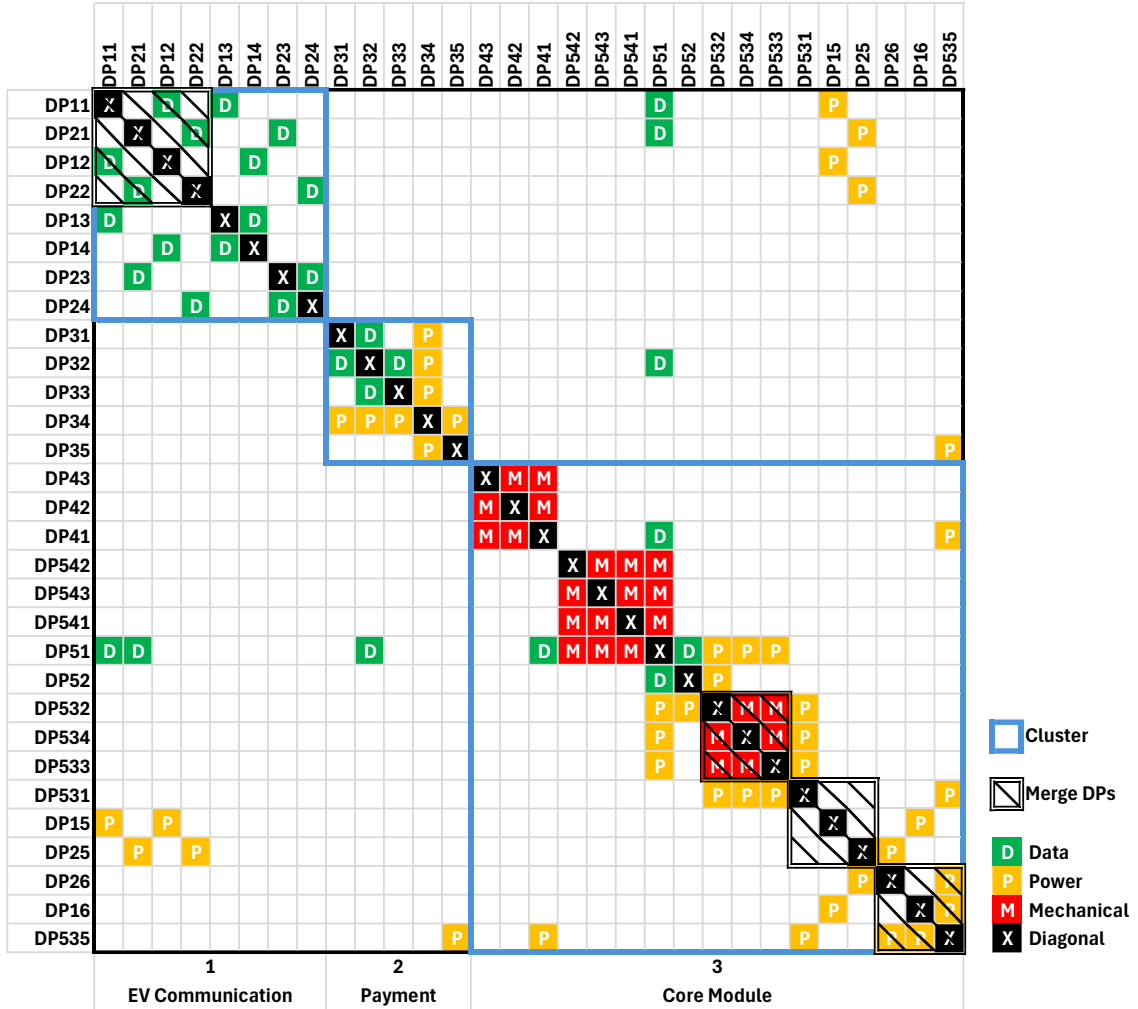
Matrix 6: Total cluster interface cost matrix.

		Cluster					
		1	2	3	4	5	6
Cluster	1	0.00	0.00	0.00	0.00	4.44	2.71
	2	0.00	0.00	0.00	0.00	4.44	2.71
	3	0.00	0.00	0.00	0.00	4.01	2.71
	4	0.00	0.00	0.00	0.00	0.37	2.71
	5	4.44	4.44	4.01	0.37	0.00	2.71
	6	2.71	2.71	2.71	2.71	2.71	0.00

* Total cluster interface cost is the sum of the costs of individual interfaces defined in Matrix 2.

Title	Semi-Modular Clustering Based on DP Interactions (Option D)
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Clustered Design Parameter DSM.



Matrix 2: Module DSM.

		Cluster		
		1	2	3
Cluster	1			A E
	2			B H
	3	A E	B H	

- A** CAN PHY
- B** 4-pin UART connector
- C** 4-pin SPI data connector
- D** 20-pin MIPI-DSI connector
- E** 2-pin 5V power connector
- F** 2-pin 3V3 power connector
- G** 8-pin PMIC power connector
- H** 2-pin 24V power connector
- I** Heatsink mount

Matrix 3: Clustered product assembly matrix CA.

	DP11	DP21	DP12	DP22	DP13	DP14	DP23	DP24	DP31	DP32	DP33	DP34	DP35	DP43	DP42	DP41	DP542	DP543	DP541	DP51	DP52	DP532	DP534	DP533	DP531	DP15	DP25	DP26	DP16	DP535
Variant 1	1	1	2	2	2	3	2	3	1	1	2	4	3	1	1	3	22	23	23	21	21	23	23	23	26	2	2	2	2	24
Variant 2	1	1	2	2	2	3	2	3	1	1	2	4	3	0	0	0	19	20	20	18	18	20	20	23	2	2	2	2	21	
Variant 3	1	0	2	0	2	3	0	0	1	1	2	4	3	1	1	3	16	17	17	15	15	17	17	17	20	2	0	0	2	18
Variant 4	1	0	2	0	2	3	0	0	1	1	2	4	3	0	0	0	13	14	14	12	12	14	14	14	17	2	0	0	2	15
Variant 5	0	1	0	2	0	0	2	3	1	1	2	4	3	1	1	3	16	17	17	15	15	17	17	17	20	0	2	2	0	18
Variant 6	0	1	0	2	0	0	2	3	1	1	2	4	3	0	0	0	13	14	14	12	12	14	14	14	17	0	2	2	0	15
Variant 7	0	0	0	0	0	0	0	0	1	1	2	4	3	1	1	3	10	11	11	9	9	11	11	14	0	0	0	0	12	
Variant 8	0	0	0	0	0	0	0	0	1	1	2	4	3	0	0	0	7	8	8	6	6	8	8	8	11	0	0	0	0	9

Matrix 4: Cluster utilization matrix H.

	Cluster		
	1	2	3
Variant 1	1.00	1.00	1.00
Variant 2	1.00	1.00	0.82
Variant 3	0.50	1.00	0.88
Variant 4	0.50	1.00	0.71
Variant 5	0.50	1.00	0.88
Variant 6	0.50	1.00	0.71
Variant 7	0.00	1.00	0.76
Variant 8	0.00	1.00	0.59

Matrix 5: Binary cluster utilization matrix H_b.

	Cluster		
	1	2	3
Variant 1	1	1	1
Variant 2	1	1	1
Variant 3	1	1	1
Variant 4	1	1	1
Variant 5	1	1	1
Variant 6	1	1	1
Variant 7	0	1	1
Variant 8	0	1	1

Vector 1: Average cluster utilization vector.

	Cluster		
	1	2	3
Average excl. unused	0.67	1.00	0.79

Vector 2: Individual DP costs.

Cluster	1	DP11	2.81	Merged		
		DP21	0.00			
		DP12	0.00			
		DP22	0.00			
		DP13	20.67			
		DP14	2.57			
		DP23	0.43			
		DP24	3.00			
	2	DP31	4.86			
		DP32	2.81			
		DP33	5.20			
		DP34	3.85			
		DP35	1.82			
		3	DP43	19.50		
			DP42	19.50		
	DP41		52.00			
	DP542		0.71			
	DP543		0.00			
	DP541		2.14			
	DP51		13.50			
	DP52		5.20			
	DP532		3.20	Merged		
	DP534		0.00			
	DP533		0.00	Merged		
DP531	3.85					
DP15	0.00	Merged				
DP25	0.00					
DP26	1.82	Merged				
DP16	0.00					
DP535	0.00					

Vector 3: P_{DM}, P_{MM} and P_{IM} and their sum vector.

	P _{DM}	P _{MM} *	P _{IM}	Σ
Cluster 1	29.48	5.90	7.14	42.52
Cluster 2	18.54	3.71	6.71	28.96
Cluster 3	121.41	36.42	13.86	171.69

* Manufacturing cost is estimated at 30% of DP costs for the cluster containing the SoC and 20% for others.

Matrix 6: Total cluster interface cost matrix.

		Cluster		
		1	2	3
Cluster	1	0.00	0.00	7.14
	2	0.00	0.00	6.71
	3	7.14	6.71	0.00

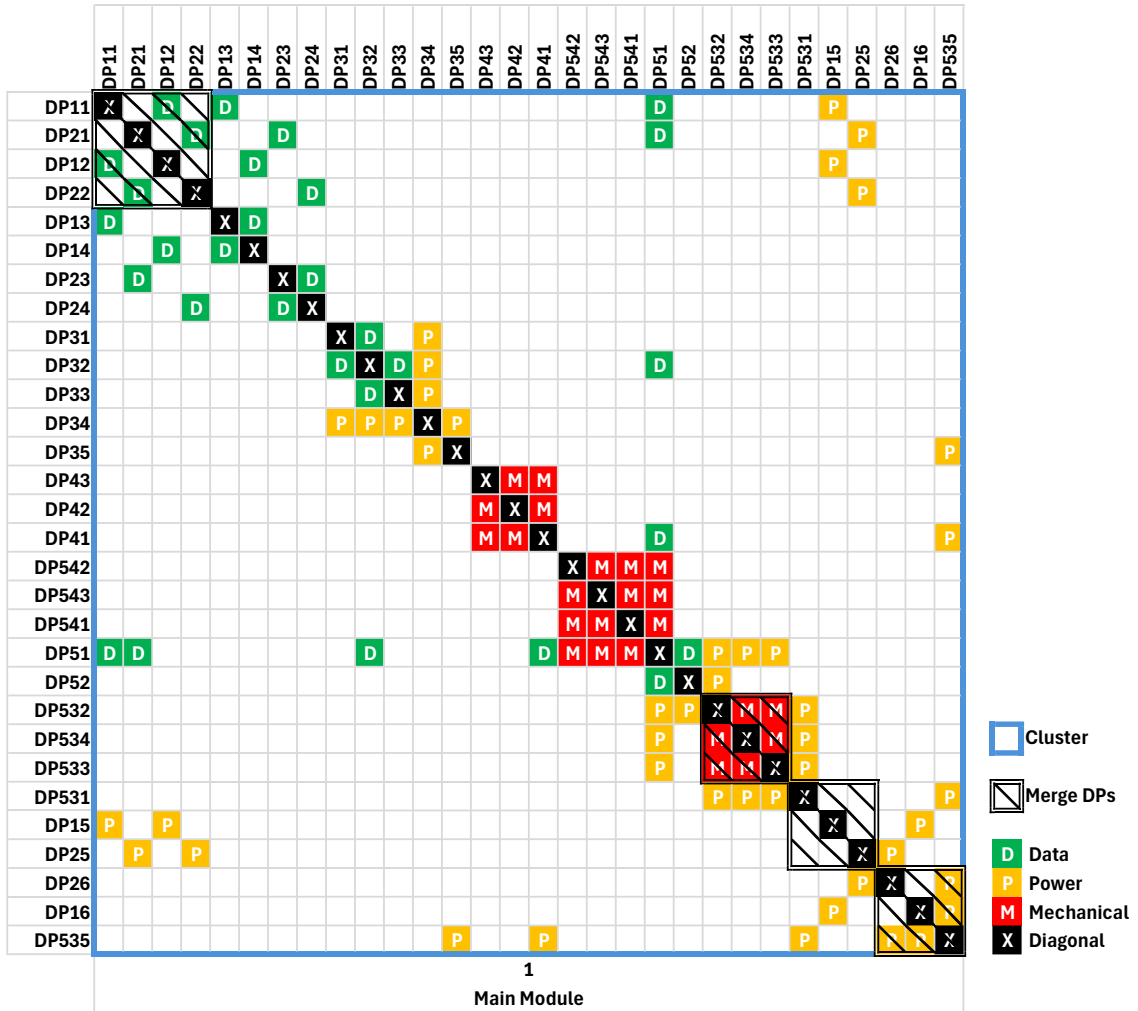
* Total cluster interface cost is the sum of the costs of individual interfaces defined in Matrix 2.

Vector 4: Variant cost vector.

	P _{modular}
Variant 1	243.17
Variant 2	243.17
Variant 3	243.17
Variant 4	243.17
Variant 5	243.17
Variant 6	243.17
Variant 7	200.64
Variant 8	200.64

Title	Completely Integral Clustering (Option E)
Revision	A
Created By	Eero Talus
Date	22/09/2024

Matrix 1: Clustered Design Parameter DSM.



Matrix 2: Module DSM.

	Cl.
Cl.	1

- A** CAN PHY
- B** 4-pin UART connector
- C** 4-pin SPI data connector
- D** 20-pin MIPI-DSI connector
- E** 2-pin 5V power connector
- F** 2-pin 3V3 power connector
- G** 8-pin PMIC power connector
- H** 2-pin 24V power connector
- I** Heatsink mount

Matrix 3: Clustered product assembly matrix CA.

	DP11	DP21	DP12	DP22	DP13	DP14	DP23	DP24	DP31	DP32	DP33	DP34	DP35	DP43	DP42	DP41	DP542	DP543	DP541	DP51	DP52	DP532	DP534	DP533	DP531	DP15	DP25	DP26	DP16	DP535	
Variant 1	1	1	2	2	2	3	2	3	1	1	2	4	3	1	1	3	22	23	23	21	21	23	23	23	23	26	2	2	2	2	24
Variant 2	1	1	2	2	2	3	2	3	1	1	2	4	3	0	0	0	19	20	20	18	18	20	20	20	23	2	2	2	2	21	
Variant 3	1	0	2	0	2	3	0	0	1	1	2	4	3	1	1	3	16	17	17	15	15	17	17	17	20	2	0	0	2	18	
Variant 4	1	0	2	0	2	3	0	0	1	1	2	4	3	0	0	0	13	14	14	12	12	14	14	14	17	2	0	0	2	15	
Variant 5	0	1	0	2	0	0	2	3	1	1	2	4	3	1	1	3	16	17	17	15	15	17	17	17	20	0	2	2	0	18	
Variant 6	0	1	0	2	0	0	2	3	1	1	2	4	3	0	0	0	13	14	14	12	12	14	14	14	17	0	2	2	0	15	
Variant 7	0	0	0	0	0	0	0	0	1	1	2	4	3	1	1	3	10	11	11	9	9	11	11	11	14	0	0	0	0	12	
Variant 8	0	0	0	0	0	0	0	0	1	1	2	4	3	0	0	0	7	8	8	6	6	8	8	8	11	0	0	0	0	9	

Matrix 4: Cluster utilization matrix H.

	Cluster 1
Variant 1	1.00
Variant 2	0.90
Variant 3	0.80
Variant 4	0.70
Variant 5	0.80
Variant 6	0.70
Variant 7	0.60
Variant 8	0.50

Matrix 5: Binary cluster utilization matrix H_B.

	Cluster 1
Variant 1	1
Variant 2	1
Variant 3	1
Variant 4	1
Variant 5	1
Variant 6	1
Variant 7	1
Variant 8	1

Vector 1: Average cluster utilization vector.

	Cluster 1
Average excl. unused	0.75

Vector 2: Individual DP costs.

Vector 3: P_{DM} , P_{MM}^* and P_{IM} and their sum vector.

Vector 4: Variant cost vector.

Cluster 1	DP11	2.81	Merged	
	DP21	0.00		
	DP12	0.00		
	DP22	0.00		
	DP13	20.67		
	DP14	2.57		
	DP23	0.43		
	DP24	3.00		
	DP31	4.86		
	DP32	2.81		
	DP33	5.20		
	DP34	3.85		
	DP35	1.82		
	DP43	19.50		
	DP42	19.50		
	DP41	52.00		
	DP542	0.71		
	DP543	0.00		
	DP541	2.14		
	DP51	13.50		
	DP52	5.20		
	DP532	3.20		Merged
	DP534	0.00		
	DP533	0.00		Merged
	DP531	3.85		
	DP15	0.00		
	DP25	0.00	Merged	
	DP26	1.82		
	DP16	0.00		
	DP535	0.00		

	P_{DM}	P_{MM}^*	P_{IM}	Σ
Cluster 1	169.43	50.83	0.00	220.26

* Manufacturing cost is estimated at 30% of DP costs for the cluster containing the SoC and 20% for others.

Matrix 6: Total cluster interface cost matrix.

		Cl.
		1
C_i	1	0.00

* Total cluster interface cost is the sum of the costs of individual interfaces defined in Matrix 2.

	P_{modular}
Variant 1	220.26
Variant 2	220.26
Variant 3	220.26
Variant 4	220.26
Variant 5	220.26
Variant 6	220.26
Variant 7	220.26
Variant 8	220.26

Title	Architecture Cost Comparison
Revision	A
Created By	Eero Talus
Date	22/09/2024

Table 1: Variant cost vector comparison between architectures.

Architecture	Product Cost												
	Integral				→ Modular								
	E	D	$\Delta(E \rightarrow D)$	$\Delta\%$	C	$\Delta(E \rightarrow C)$	$\Delta\%$	B	$\Delta(E \rightarrow B)$	$\Delta\%$	A	$\Delta(E \rightarrow A)$	$\Delta\%$
Variant 1	220.26	243.17	22.91	+10.4	262.72	42.46	+19.3	286.86	66.59	+30.2	293.50	73.24	+33.2
Variant 2	220.26	243.17	22.91	+10.4	150.45	-69.81	-31.7	286.86	66.59	+30.2	181.22	-39.04	-17.7
Variant 3	220.26	243.17	22.91	+10.4	248.08	27.82	+12.6	286.86	66.59	+30.2	272.06	51.80	+23.5
Variant 4	220.26	243.17	22.91	+10.4	135.81	-84.45	-38.3	286.86	66.59	+30.2	159.79	-60.47	-27.5
Variant 5	220.26	243.17	22.91	+10.4	224.32	4.06	+1.8	243.84	23.58	+10.7	248.30	28.03	+12.7
Variant 6	220.26	243.17	22.91	+10.4	112.04	-108.22	-49.1	243.84	23.58	+10.7	136.02	-84.24	-38.2
Variant 7	220.26	200.64	-19.62	-8.9	209.68	-10.58	-4.8	221.60	1.34	+0.6	226.86	6.60	+3.0
Variant 8	220.26	200.64	-19.62	-8.9	97.41	-122.85	-55.8	221.60	1.34	+0.6	114.59	-105.67	-48.0
Minimum	220.26	200.64	-19.62	-8.9	97.41	-122.85	-55.8	221.60	1.34	+0.6	114.59	-105.67	-48.0
Maximum	220.26	243.17	22.91	+10.4	262.72	42.46	+19.3	286.86	66.59	+30.2	293.50	73.24	+33.2
Average	220.26	232.54	12.28	+5.6	180.06	-40.20	-18.2	259.79	39.53	+17.9	204.04	-16.22	-7.4

Figure 1: Absolute cost differences with different architecture change scenarios.

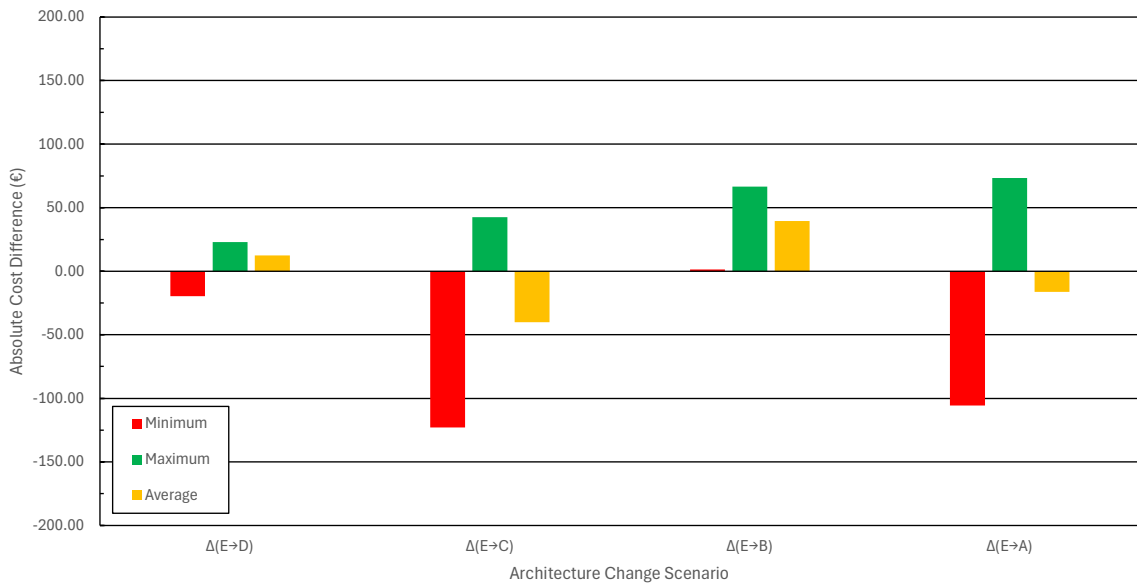


Figure 2: Relative cost differences with different architecture change scenarios.

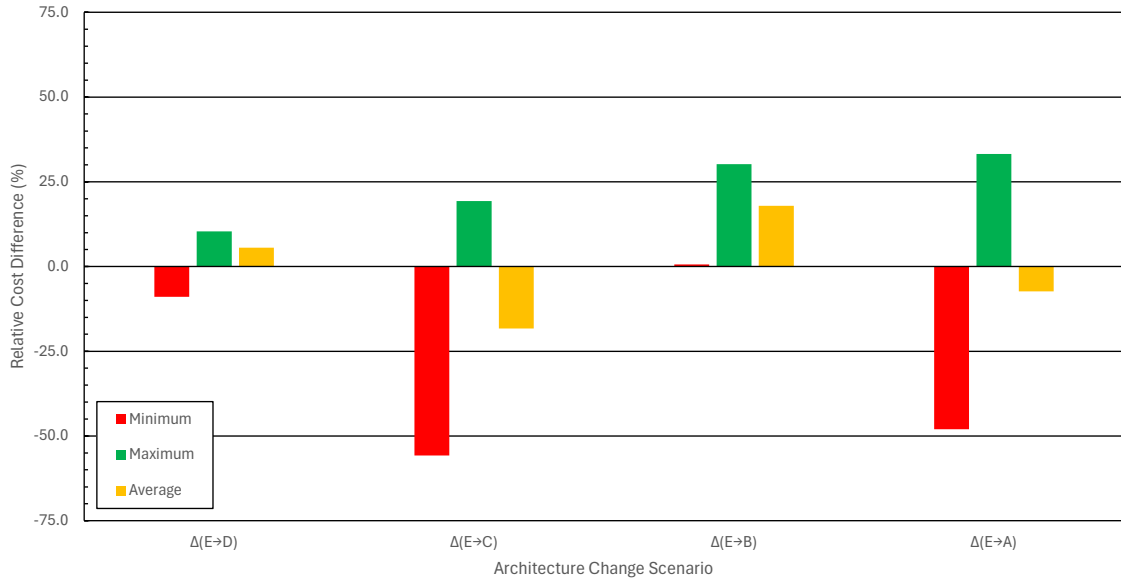
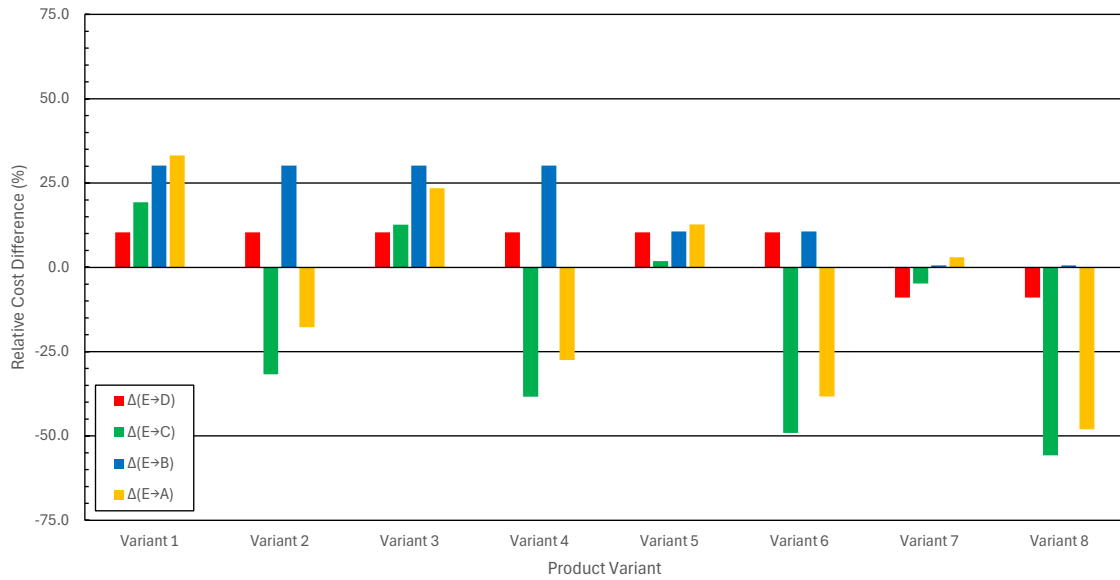


Figure 3: Relative cost differences per variant with different architecture change scenarios.



Title	Design Parameter and Interface Cost Data
Revision	A
Created By	Eero Talus
Date	22/09/2024

Notes

The selected components are for cost estimation only and they include only the largest cost contributors. Component selections were done based on manufacturer specifications, but compatibility of the components was not thoroughly checked. Cost data was obtained from the online catalogs of distributors between 22/08/2024 and 06/09/2024. Where data was not available, costs were estimated. In addition, the cost of supportive components required for the correct operation of the main cost contributors was estimated at 30% of main component cost.

DigiKey: <https://www.digikey.com>

Codico: <https://www.codico.com>

Table 1: Design Parameter Costs.

DP	MPN	Unit Price @ 1000 Qty	Distributor	Notes
DP11	CPU	2.8116 €	N/A	
	STM32L431CBT6	2.1628 €	DigiKey	
	Support components	0.6488 €	N/A	Est. 30% of main component cost.
DP12	Digital IO	0.0000 €	N/A	Built-in processor IO.
	Support components	0.0000 €	N/A	
DP13	PLC PHY	20.6700 €	N/A	
	RED-BEET-E 2.0 SPI (EVSE) Module	15.9000 €	Codico	
	Support components	4.7700 €	N/A	Est. 30% of main component cost.
DP14	4-pin Molex Micro-Fit connector	2.5688 €	N/A	
	Molex 0430450410	1.9760 €	DigiKey	
	Support components	0.5928 €	N/A	Est. 30% of main component cost.
DP15	Buck converter	3.8453 €	N/A	
	Texas Instruments TPS563300DRLR	0.3485 €	DigiKey	
	Würth Elektronik 74439346068 Inductor	2.0755 €	DigiKey	
	2x TDK C3216X5R1V226M160AC Output Capacitor	0.2402 €	DigiKey	
	2x TDK CGA5L1X7R1H106K160AC Input Capacitor	0.2938 €	DigiKey	
	Support components	0.8874 €	N/A	Est. 30% of main component cost.
DP16	2-pin Molex Micro-Fit connector	1.8188 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Support components	0.4197 €	N/A	Est. 30% of main component cost.
DP21	CPU	2.8116 €	N/A	
	STM32L431CBT6	2.1628 €	DigiKey	
	Support components	0.6488 €	N/A	Est. 30% of main component cost.
DP22	Digital IO	0.0000 €	N/A	Built-in processor IO.
	Support components	0.0000 €	N/A	
DP23	CAN PHY	0.4306 €	N/A	
	Microchip ATA6561-GAQW-N	0.3312 €	DigiKey	
	Support components	0.0994 €	N/A	Est. 30% of main component cost.
DP24	8-pin Molex Micro-Fit connector	3.0035 €	N/A	
	Molex 0430450809	2.3104 €	DigiKey	
	Support components	0.6931 €	N/A	Est. 30% of main component cost.
DP25	Buck converter	3.8453 €	N/A	
	Texas Instruments TPS563300DRLR	0.3485 €	DigiKey	
	Würth Elektronik 74439346068 Inductor	2.0755 €	DigiKey	
	2x TDK C3216X5R1V226M160AC Output Capacitor	0.2402 €	DigiKey	
	2x TDK CGA5L1X7R1H106K160AC Input Capacitor	0.2938 €	DigiKey	
	Support components	0.8874 €	N/A	Est. 30% of main component cost.
DP26	2-pin Molex Micro-Fit connector	1.8188 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Support components	0.4197 €	N/A	Est. 30% of main component cost.
DP31	RFID interface	4.8599 €	N/A	
	ST Microelectronics ST25R3916B-AQWT	3.7384 €	DigiKey	
	Support components	1.1215 €	N/A	Est. 30% of main component cost.
DP32	Secure CPU	2.8116 €	N/A	
	STM32L431CBT6	2.1628 €	DigiKey	
	Support components	0.6488 €	N/A	Est. 30% of main component cost.

DP33	Ethernet hardware	5.2000 €	N/A	
	Microchip KSZ8851SNL-TR	4.0000 €	DigiKey	
	Support components	1.2000 €	N/A	Est. 30% of main component cost.
DP34	Buck converter	3.8453 €	N/A	
	Texas Instruments TPS563300DRLR	0.3485 €	DigiKey	
	Würth Elektronik 74439346068 Inductor	2.0755 €	DigiKey	
	2x TDK C3216X5R1V226M160AC Output Capacitor	0.2402 €	DigiKey	
	2x TDK CGA5L1X7R1H106K160AC Input Capacitor	0.2938 €	DigiKey	
	Support components	0.8874 €	N/A	Est. 30% of main component cost.
DP35	2-pin Molex Micro-Fit connector	1.8188 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Support components	0.4197 €	N/A	Est. 30% of main component cost.
DP43	Tempered cover glass	19.5000 €	N/A	
	Glass	15.0000 €	N/A	Estimated cost.
	Support components	4.5000 €	N/A	Est. 30% of main component cost.
DP42	Mechanical mounting frame	19.5000 €	N/A	
	Frame	15.0000 €	N/A	Estimated cost.
	Support components	4.5000 €	N/A	Est. 30% of main component cost.
DP41	1920x1080 LCD panel unit with 24V power input	52.0000 €	N/A	
	LCD	40.0000 €	N/A	Estimated cost.
	Support components	12.0000 €	N/A	Est. 30% of main component cost.
DP51	Multicore SoC	13.4980 €	N/A	
	ST Microelectronics STM32MP157FAD1	10.3830 €	DigiKey	
	Support components	3.1149 €	N/A	Est. 30% of main component cost.
DP52	Ethernet hardware	5.2000 €	N/A	
	Microchip KSZ8851SNL-TR	4.0000 €	DigiKey	
	Support components	1.2000 €	N/A	Est. 30% of main component cost.
DP532	PMIC buck converter	3.1959 €	N/A	
	ST Microelectronics STPMIC1APQR	2.4584 €	DigiKey	
	Support components	0.7375 €	N/A	Est. 30% of main component cost.
DP534	PMIC buck converter	3.1959 €	N/A	
	ST Microelectronics STPMIC1APQR	2.4584 €	DigiKey	
	Support components	0.7375 €	N/A	Est. 30% of main component cost.
DP533	PMIC buck converter	3.1959 €	N/A	
	ST Microelectronics STPMIC1APQR	2.4584 €	DigiKey	
	Support components	0.7375 €	N/A	Est. 30% of main component cost.
DP531	Buck converter	3.8453 €	N/A	
	Texas Instruments TPS563300DRLR	0.3485 €	DigiKey	
	Würth Elektronik 74439346068 Inductor	2.0755 €	DigiKey	
	2x TDK C3216X5R1V226M160AC Output Capacitor	0.2402 €	DigiKey	
	2x TDK CGA5L1X7R1H106K160AC Input Capacitor	0.2938 €	DigiKey	
	Support components	0.8874 €	N/A	Est. 30% of main component cost.
DP535	2-pin Molex Micro-Fit connector	1.8188 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Support components	0.4197 €	N/A	Est. 30% of main component cost.
DP542	Thermal interface material	0.7111 €	N/A	
	Chip Quick TC3-1G	0.5470 €	DigiKey	Est. 10 uses per 5.47€ syringe.
	Support components	0.1641 €	N/A	Est. 30% of main component cost.
DP543	CPU mounting solution	0.0000 €	N/A	
	N/A	0.0000 €	N/A	Integrated in heatsink.
	Support components	0.0000 €	N/A	Est. 30% of main component cost.
DP541	Metal heatsink	2.1421 €	N/A	
	CUI Devices HSB41-303014P	1.6478 €	DigiKey	
	Support components	0.4943 €	N/A	Est. 30% of main component cost.

Table 2: Interface costs.

Interface	MPN	Unit Price @ 1000 Qty	Distributor	Notes
A CAN PHY		4.4366 €	N/A	
	Microchip ATA6561-GAQW-N	0.3312 €	DigiKey	
	Molex 0430450410	1.9760 €	DigiKey	
	Molex 2147551042	1.1055 €	DigiKey	Cost x0.5 to split between modules.
	Support components	1.0238 €	N/A	Est. 30% of main component cost.
B 4-pin UART connector		4.0060 €	N/A	
	Molex 0430450410	1.9760 €	DigiKey	
	Molex 2147551042	1.1055 €	DigiKey	Cost x0.5 to split between modules.
	Support components	0.9245 €	N/A	Est. 30% of main component cost.
C 4-pin SPI data connector		4.0060 €	N/A	
	Molex 0430450410	1.9760 €	DigiKey	
	Molex 2147551042	1.1055 €	DigiKey	Cost x0.5 to split between modules.
	Support components	0.9245 €	N/A	Est. 30% of main component cost.
D 20-pin MIPI-DSI connector		0.3672 €	N/A	
	Amphenol SFV20R-4STE1HLF	0.2824 €	DigiKey	
	Support components	0.0847 €	N/A	Est. 30% of main component cost.
E 2-pin 5V power connector		2.7063 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Molex 2147551022	0.6827 €	DigiKey	Cost x0.5 to split between modules.
	Support components	0.6245 €	N/A	Est. 30% of main component cost.
F 2-pin 3V3 power connector		2.7063 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Molex 2147551022	0.6827 €	DigiKey	Cost x0.5 to split between modules.
	Support components	0.6245 €	N/A	Est. 30% of main component cost.
G 8-pin PMIC power connector		5.1466 €	N/A	
	Molex 0430450809	2.3104 €	DigiKey	
	Molex 2147551082	1.6485 €	Digikey	Cost x0.5 to split between modules.
	Support components	1.1877 €	N/A	Est. 30% of main component cost.
H 2-pin 24V power connector		2.7063 €	N/A	
	Molex 0430450210	1.3991 €	DigiKey	
	Molex 2147551022	0.6827 €	DigiKey	Cost x0.5 to split between modules.
	Support components	0.6245 €	N/A	Est. 30% of main component cost.
I Heatsink mount		0.0000 €	N/A	
	Mounting geometry	0.0000 €	N/A	Integrated in heatsink.
	Support components	0.0000 €	N/A	Est. 30% of main component cost.