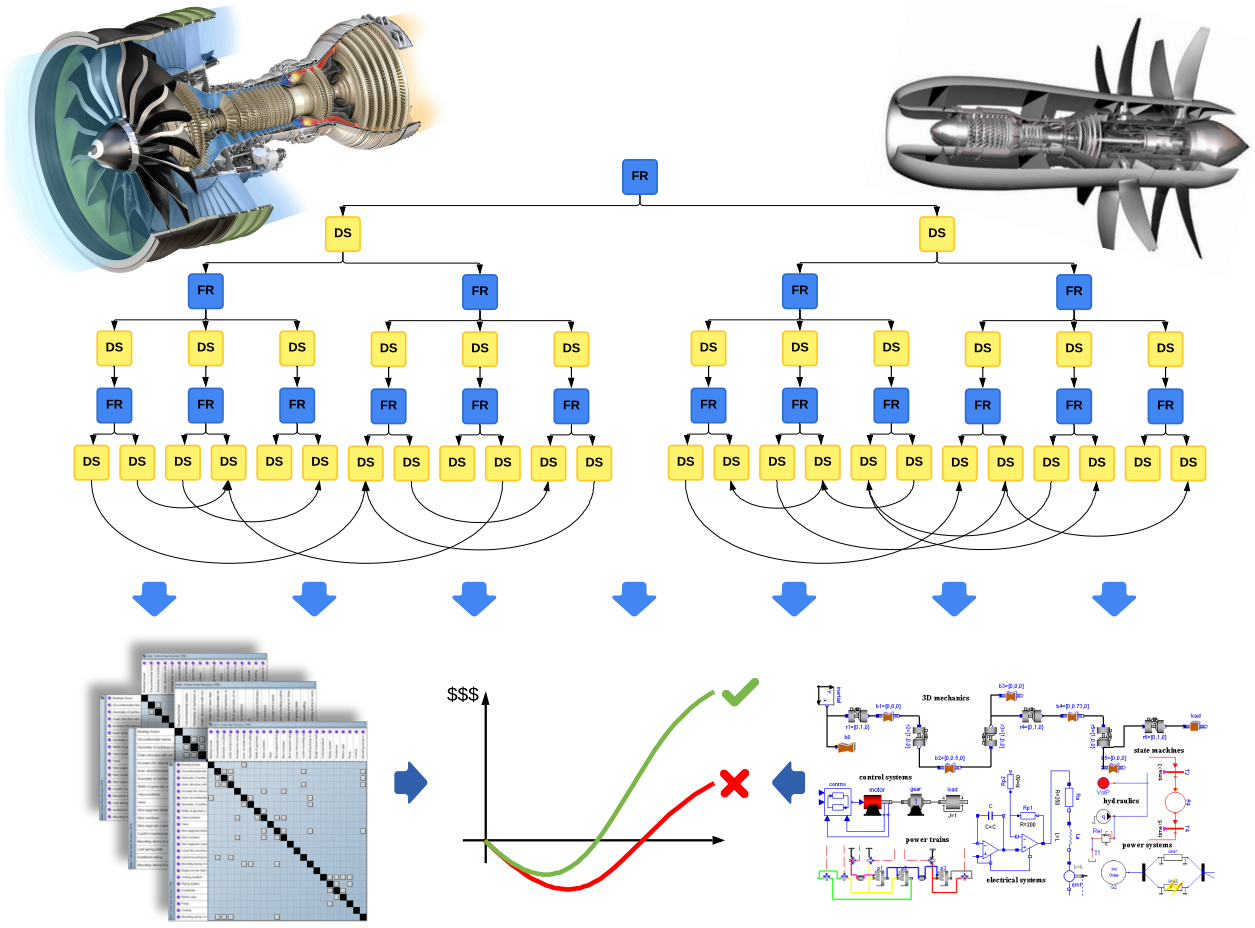
CHALMERS
UNIVERSITY OF TECHNOLOGY



Enriched Functional Modelling Software to Assess Aerospace System Architectures

Master's thesis in Product Development

OSKAR STRANDH THOLIN

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Enriched Functional Modelling Software to Assess Aerospace System Architectures

OSKAR STRANDH THOLIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Product Development
Systems Engineering Design
Chalmers University of Technology
Gothenburg, Sweden 2021

Enriched Functional Modelling Software to Assess Aerospace System Architectures
OSKAR STRANDH THOLIN

Supervisor and examiner:
Associate Professor Massimo Panarotto, Industrial and Materials Science

Master's Thesis 2021
Department of Industrial and Materials Science
Division of Product Development
Systems Engineering Design
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A visualisation of the possibilities of the enhanced software tool. Subimages from Alhamaly (2019), Page (2009) and "Modelica Libraries" (2000).

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Enriched Functional Modelling Software to Assess Automotive System Architectures
OSKAR STRANDH THOLIN
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

The aerospace manufacturing industry is transforming, bringing in electrical and autonomous technologies into both products and production processes at a faster pace than the past. Yet manufacturers are not exploring radically different variants or product architectures early in the product development process, which is when changes in design are most economical. A link is needed in the design process between the problem at hand and the design phase, where not just radically different architectures can be analysed and thus represented together, but also where properties unrepresentable by current CAD systems are possible to analyse.

To tackle these challenges and to strengthen the available set of tools and methods used before a geometry is available, this thesis has focused on the enhancement of an existing functional modelling software developed at the department of Industrial and Materials Sciences at Chalmers University of Technology. The tool uses the enhanced function-means (EF-M) method to represent product architectures side-by-side with alternatives. The problem was divided into four functionality levels, which were to be implemented sequentially into the tool. The four functionalities consisted among others of implementing interface visualisation and characterisation, extraction of the interfaces in design structure matrices (DSMs), the addition of a function-component library and the connection of the tool to an external, third-party simulation software.

The benefits of this development are the utilisation of historic data of the organisation in the EF-M models and the connection of the tool to established methods. The DSM models are well-defined in the literature and their extraction from the tool means that the modelled products can be analysed further in external software. Furthermore, the enhanced tool paves the way for architecture discovery by simulation, since many steps of the process can with relative ease be automated.

Keywords: product development, concept development, concept generation, concept evaluation, function modelling, non-functional requirements, EF-M, DSM, Modelica

Acknowledgements

Support, help and guidance have come from many during the thesis. It is an understatement to say that little could have been possible without this engagement from various actors. I would like to express my gratitude, thankfulness and appreciation to the following people and organisations, who represent the *many* in the first sentence of this paragraph.

First and foremost, to Associate Professor Massimo Panarotto of Chalmers University of Technology, for the supervision and expertise provided throughout the thesis, for acting as examiner and providing the thesis, for the time he invested to ensure quality of the thesis and for his dedication in supporting me throughout it.

To Dr Jakob Müller of Chalmers University of Technology, for technical supervision during the development, for insight into the EF-M tool and EF-M modelling as well as for helpful and friendly guidance.

To Dr Timos Kipouros of the University of Cambridge, for technical guidance regarding the connection to CAM software, for overseeing a functionality of batch importing DSMs into CAM specifically for this thesis, and for fruitful discussions regarding the improvements of in the thesis implemented DSM-based functionalities.

To Professor Ola Isaksson and people involved in the DIAS project, both of Chalmers University of Technology, for the participation in various meetings, for allowing me to present my progress and receive constructive feedback.

To the company GKN Aerospace Sweden AB from Trollhättan, Sweden, for providing up-to-date and relevant material and data as support of the thesis and for participation in various meetings.

I thank you sincerely.

Oskar Strandh Tholin, Gothenburg, June 2021

Nomenclature

CAD	Computer-aided design.
CAM	Cambridge Advanced Modeller. Only the second version, CAM 2, is implied here.
CFD	Computational fluid dynamics.
CPM	Change prediction method.
DC	Direct current.
DMM	Domain mapping matrix
DS	Design solution.
DSM	Design structure matrix.
EF-M	Enhanced function-means.
F-M	Functions-means.
FB	Functional basis.
FEA	Finite element analysis.
FR	Functional requirement.
IC	The DSM convention *inputs in columns*.
icb	"is-constrained-by" connections in EF-M modelling.
IMS	The department of *Industrial and Materials Science* at Chalmers University of Technology.
IR	The DSM convention *inputs in rows*.
isb	"is-solved-by" connections in EF-M modelling.
iw	"interacts-with" conncections in EF-M modelling. "iw's" indicates plural.
rf	"requires-function" connections in EF-M modelling.
TRS	Turbine rear structure.

Contents

List of Figures

List of Tables

1
Introduction

Many industries are currently and rapidly transforming by bringing in electrical and autonomous technologies into both products and processes (Eckert et al., 2019). At the same time, many manufacturing companies are still refining and optimizing the same products year after year, without exploring radically new solutions or alternative architectures. This is a problem because future technologies may be noncompliant with the current product architecture. The risks and consequences of new technology introduction need to be fully understood before committing to them (Isaksson et al., 2016). Furthermore, the tools and methods available are expensive to set up, not very flexible and only explores a very limited area of the design space (Müller et al., 2020). When considering alternative architectures, a much larger spectrum needs to be covered compared to what is being done today.

Computer-aided design (CAD) is one of the tools used for designs pace exploration today. Companies try to explore a larger design space by applying extensive parametrisation and thereby creating a large number of variants. This approach is very well suited for running simulations and for understanding functional performances. However, this typically makes the development effort focus on only one concept or on one architecture (Isaksson et al., 2016) and it is dependant on an existing geometry. The current parametrisation efforts focus on generating a lot of variants of a single architecture which itself is not subject for variation. Furthermore, due to their limited information content, CAD solely is not considered sufficient for design space exploration (Müller et al., 2019).

CAD-models also only allow for a limited flexibility and cannot capture non-functional requirements (Müller et al., 2020), which can for example be complexity, flexibility, modularity or the ability to integrate new technologies. In industries that are radically transforming through electrification and automation – such as the transport manufacturing industry is today – new technologies are often introduced in products and innovations are common. Situations like this are making it more crucial to be able to assess non-functional requirements early in the development process. Studies have also shown that both the novelty and variety of generated concepts are stimulated when being constrained by non-functional requirements (Worinkeng et al., 2015), which further establishes the ability to assess them as crucial.

Product designers find it difficult to assess the potential complexity of a product when new functionalities and technologies are introduced, as well as when single components are used to fulfil multiple functions (Gonzalez Castro et al., 2020). Situations like these are frequently encountered during new technology introduction. The number of components in a product can be reduced through electrification

or introduction of software. A reduction of components can seemingly reduce the product's complexity, when in fact the complexity of the interactions between the components in such a case grows (Gonzalez Castro et al., 2020). This suggests an optimisation problem for designers, where the optimal trade-off between a product with a lot of components and a product with complex components needs to be found. Such decisions need to be made already during the concept phase, which is a phase that is characterised by changes in design and architecture being both easy and cheap, but at the same time by limited knowledge of the problem (Panarotto et al., 2018).

What can be seen to be missing here is the possibility for product designers to perform a first order assessment during the early stages of the product development process. During such an initial assessment, radically different architectures and alternatives can be analysed on multiple evaluation criteria, including – but not limited to – non-functional requirements. To do this, a tool that produces the information needed to make these decisions, analyses and judgements is missing.

The benefits of such development are the utilisation of already well-defined methods in the literature (e.g., DSM) and other existing software. This means that initially identified architectures can be exported and communicated with other simulation and analysis tools and the models can be enriched with additional and more detailed information. This can then be used to discover new architectures or to refine and classify the existing architecture. Furthermore, recent developments in architecture discovery by simulation could potentially be utilised.

1.1 Aim

The thesis will focus on the further enhancement of a functional modelling software, which has been developed at the department of Industrial and Materials Science at Chalmers University of Technology. The software tool is described in chapter 2 and is used to represent product architectures. The enhancement will consist of adding the following functionality: product interface visualisation; matrix representation of certain aspects of the model; a library of modelling elements; and the connection of the tool to an external simulation software. This is described in more detail in chapter 3.

1.2 Research Questions

The thesis will aim to answer the following two research questions:

> *In what way does the enhancement of the tool contribute to the early stages of the product development process?*
>
> *What key non-functional modelling criteria can be modelled with the enhanced tool compared to existing CAD systems?*

2
Literature Review

A process is "a continuous and regular action or succession of actions occurring or performed in a definite manner, and having a particular result or outcome" ("process, n." 2021). The *product development process* is the structured manner from the conception of a market opportunity to the distribution of a product. This process can employ everything from a small team to entire companies and it can span a year or the good part of a decade. A successful product development process brings together multiple functions of a company – such as marketing, design and manufacturing – in a way that quickly produces a product at low cost that meets the customer needs. To understand how this is possible and to provide a knowledge basis for the rest of the thesis, this chapter commences with an explanation of the product development process. (Ulrich & Eppinger, 2016, pp. 2, 5, 12)

The product development process consists of six phases and can be seen as the stage-gate process pictured in Figure 2.1. Each stage (□) is succeeded by a gate (◇), which can either act as a go/kill decision point for the stake holders or as a control point that the stage's objectives are fulfilled before continuing with the next stage (Cooper, 2011, pp. 147–148). The product development process differs from company to company and project to project since it heavily depends on the context and environment of its implementation. However, the processes generally contain the same steps as well as start and end similarly, but they instead differ in terms of the exact content of the steps as well as the amount of iterations and the distribution of how the resources are invested (Ulrich & Eppinger, 2016, pp. 22–23). What will be described here will be the general, also called the generic, product development process.

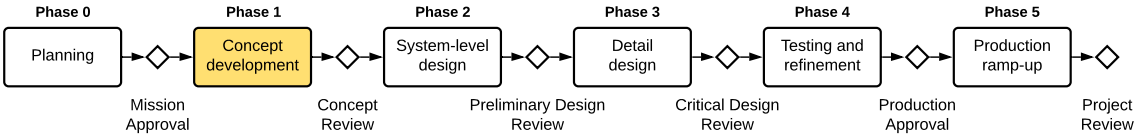


Figure 2.1: The generic product development process as a stage-gate process after Ulrich and Eppinger (2016, p. 22). The stages are denoted with rectangles (□) whereas the gates with diamonds (◇). The concept development phase has been highlighted with yellow as this is the phase treated in this thesis.

The process begins with the *planning phase* (phase 0), which is labelled phase zero because it lies before the approval and start of the project. During this step an opportunity for a product is found and investigated. A strategy for commercialising a product that solves this problem is developed and presented for a stake holder for

approval. Once the project idea has a green light, the product development project begins with the *concept development* phase (phase 1). Here, the customer needs and target specifications of the product are defined, and the space of possible alternatives – the design space – is explored by generating multiple concepts. The concepts are then evaluated and tested based on the customer needs and target specifications and at least one is chosen for further development. After this, the *system-level design* (phase 2) takes place, whereby the product architecture and some central components are designed. The product is decomposed into components and subsystems, and the design embodiment of the selected concept/concepts is carried out. The components are then completely specified for manufacturing and assembly in the *detail design* (phase 3). The geometry and tolerances are decided, the material is selected, the standard and unique parts are identified, and the production cost is calculated. After that extensive prototyping takes place during the *testing and refinement* phase (phase 4). Early prototypes of just equal geometry and material (alpha prototypes), and more mature prototypes using the actual and intended production processes (beta prototypes), are used to validate the product's functionality, performance and reliability. The final phase in the product development process is the *production ramp-up* (phase 5), during which the production is prepared and initiated. Also, during this step, the product is finally launched on the market and becomes available to the intended customers. The step concludes with a project review, during which the project is assessed and lessons learnt are collected for the sake of advancing the development process for future projects. (Ulrich & Eppinger, 2016, pp. 13–16)

This thesis is mostly concerned with the concept development phase of the product development process, which is why this has been highlighted in Figure 2.1. This phase has been expanded into individual steps in Figure 2.2. The thesis will in particular treat the *generate product concepts* and the *select product concept(s)* phases of the concept development, since these lie closest to the problem statement of chapter 1. During the select product concepts phase, the concepts generated in the previous phase are evaluated and one or more concepts are "selected for further investigation,

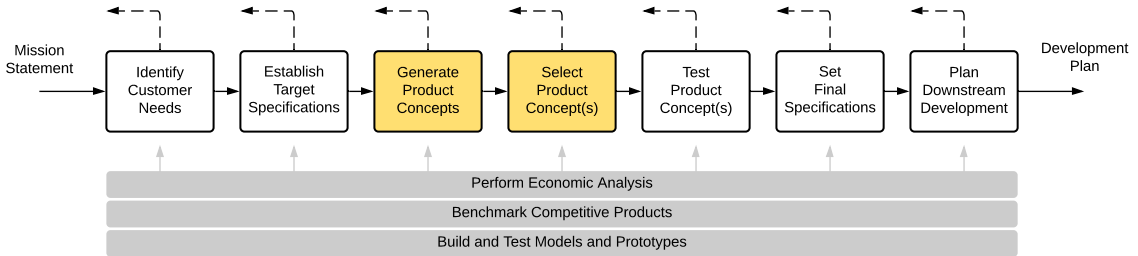


Figure 2.2: The concept development phase of the product development process expanded in its different activities after Ulrich and Eppinger (2016, p. 16). The dashed arrows indicate that the process is iterative and that one might have to redo or reconsider the material from previous steps. The grey segments below the expanded concept development phase illustrate activities and tasks that are to be continually performed throughout the process. The concept generation and concept evaluation phases will be more closely treated in the rest of the chapter and have therefore been highlighted in yellow.

testing, or development" (Ulrich & Eppinger, 2016, p. 146). These two phases will hereafter be referred to as the *concept generation* phase and the *concept evaluation* phase respectively.

Two theoretical concepts need to be defined here, namely concept and architecture. A product *concept* is defined as a rough description of a product in terms of geometry, key technologies and ways of function. The concept is in this way a detailed explanation of how the customer needs, which have been translated into target specifications, are to be fulfilled. The product *architecture* specifies the mapping of the functional to the physical elements, where the functional elements are the functions required for the product's performance and the physical elements are the product's components and building blocks. The functional and physical elements are later in the thesis called functional requirements and design solutions respectively. (Ulrich & Eppinger, 2016, pp. 118, 186–187)

The number of concepts varies throughout the concept development process as illustrated in Figure 2.3. It commences with the concept generation where a huge

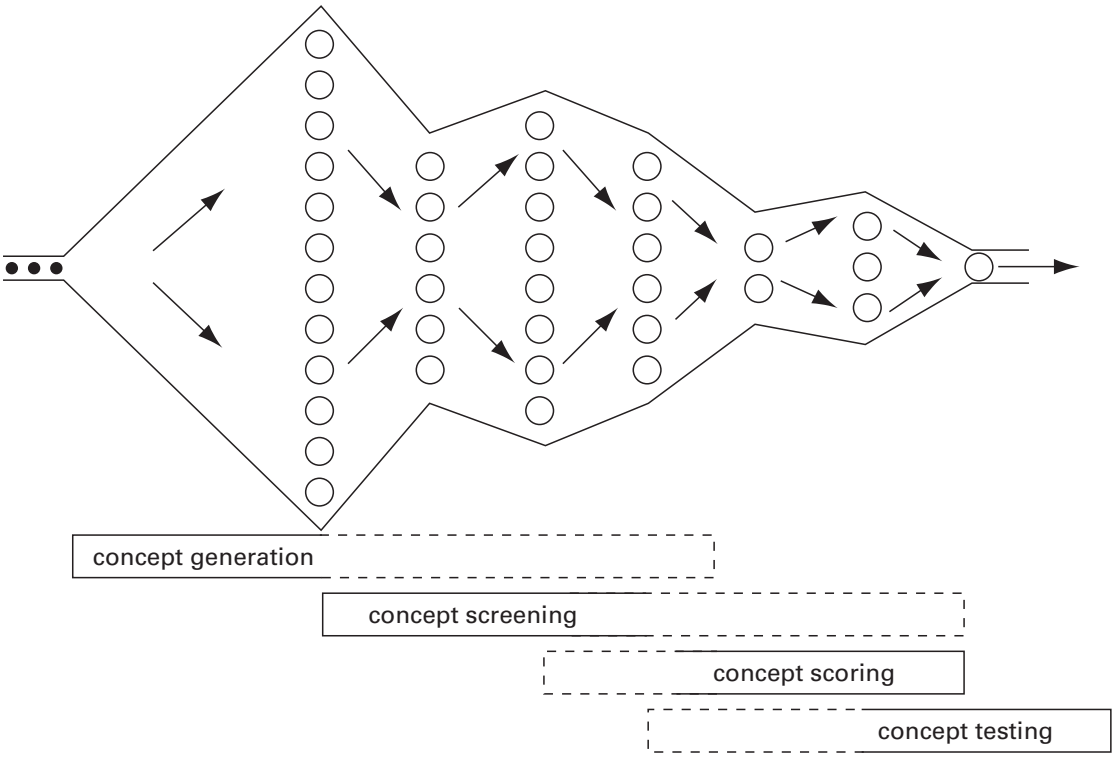


Figure 2.3: The number of concepts under consideration during the concept generation and evaluation of the concept development phase is shown here as the distance between the lines. The concepts, which are here shown as circles, increase rapidly during the concept generation. These are then reduced and refined during the concept evaluation. The number of concepts can temporarily increase during the evaluation phase, as new alternatives are found by combining and changing less suitable concept. Two methods are mainly used here: concept screening and concept scoring. (Ulrich & Eppinger, 2016, p. 150)

number of concepts are produced. During the ensuing concept evaluation, the concepts are gradually reduced and refined at the same time, by changing and combining concepts to cancel-out disadvantages. These changes and combinations, as well as the fact that the team members' insight into the problem increases, can lead to bursts of additional concept creation as shown in the figure. (Ulrich & Eppinger, 2016, pp. 150–151)

The concept generation and concept evaluation phases the concept development process have been the main research areas of the thesis. The state of the art of these phases are elaborated on in the coming sections of this chapter, more specifically in section 2.1 and 2.2 respectively. Finally, a reflection on the current state of the art of the literature review is presented in section 2.3.

2.1 Concept Generation

> *The concept generation process begins with a set of customer needs and target specifications and results in a set of product concepts from which the team will make a final selection.*
>
> Ulrich and Eppinger (2016, p. 140)

The concept generation phase is an integral part of the concept development process. It succeeds the phases where customer needs are identified and target specifications are established. The phase takes these customer needs and target specifications and outputs a set of concepts. The phase is characterised by being inexpensive and time efficient relative the other phases of developing a product, which means that it neither makes much sense to avoid investing resources that ensure quality in the result of this phase nor to rush this step. Ultimately, the entire design space should be explored in the concept generation phase, which minimises the risk of not considering the best concept and of being dramatically outperformed by solutions found by competitors. (Ulrich & Eppinger, 2016, pp. 118–119)

A five-step method concept generation method can be used to support this phase. The method, which is presented in Figure 2.4, can be applied on almost every product. The steps are to be seen as a basis from where product developers can refine their own methods. The linear approach that the method is here presented in can be

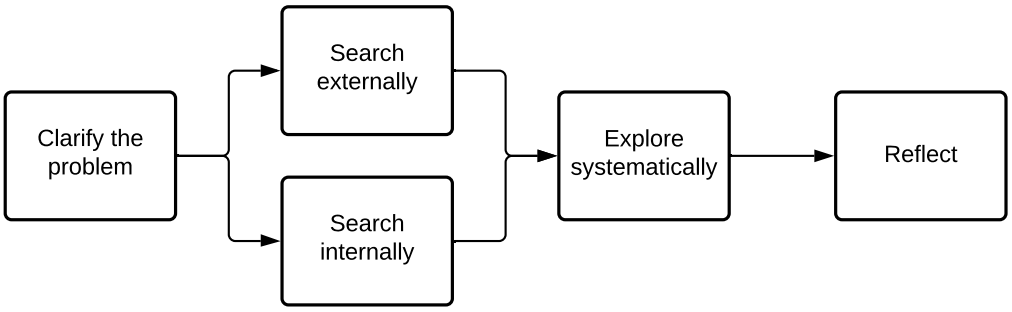


Figure 2.4: A five-step concept generation method. (Ulrich & Eppinger, 2016, p. 120)

misleading and the reader should note that concept generation is an iterative process.

The first step, *clarify the problem*, is about understanding the problem and to reduce its complexity. A good way to do this is to apply problem decomposition on the most critical subproblems, where one complex problem is broken down into multiple, more simple problems. A good way to do this is to use functional modelling. A complex problem or product can be seen as a black box, where one can only see the inputs and outputs, but not how they are transformed in the black box. One can apply functional decomposition on this black box to try to work out how the inputs are transformed to the outputs and what functions are needed to support this invisible mechanism inside the black box. The team can then focus on finding solutions for the subfunction. (Ulrich & Eppinger, 2016, pp. 120–124)

The next two steps, which are *search externally* and *search internally*, can happen in parallel or in any order as well as throughout the entire concept generation. The external search is about finding existing solutions that can be helpful in solving either the main problem or the subproblems. Following the engineering paradigm of not reinventing the wheel, the external search helps the development team use the fact that it is less resourceful to implement existing solutions than to develop new ones. Multiple sources can be used when searching externally, such as interviewing lead users, consulting experts, searching in patents and literature, and benchmarking the competitors. The internal search, on the other hand, is about finding new solutions from the knowledge that the team members already have. Equated with brainstorming, the internal search should follow five rules to be as effective as possible: suspend judgment, generate a lot of ideas, welcome ideas that may seem infeasible, make plenty of sketches, and build sketch models. (Ulrich & Eppinger, 2016, pp. 124–131)

When arriving at the fourth step, the product development team has typically generated more concepts than can individually be thoroughly considered, and which the team now should *explore systematically*. For example, if there were ten subproblems, for which the team had found five solutions each, the total number of generated concepts would be 9 765 625 ($= 5^{10}$). Going through all the concepts would not only be rather impossible but also highly wasteful since many of the generated concepts do not make much sense. Two approaches are used to limit the number of concepts and to assess them, namely the concept classification tree and the concept combination table. The classification tree is used to divide the concepts into independent categories. This can be used to remove all solutions that are based on an inferior approach, such as removing all concepts of a drill that are powered by nuclear fusion. Also, different independent approaches can be identified using the tree, and the developing team can also be privy to the fact that different branches of the tree have been unequally research and that some are requiring more attention. The combination table is then used to selectively guide the consideration of individual concepts. One can say that the tree is used to rapidly reduce the concepts under consideration and the table to consider the concepts that are left. (Ulrich & Eppinger, 2016, pp. 131–139)

The fifth and final step is to *reflect* on the solutions and the process. The step is used to help future projects as well as upcoming iterations to be performed better. The team should among others assess whether there exist alternative branches or approaches that can or could be explored further, whether all external sources have

been exhausted and whether the entire design space has been explored. This step is also supposed to be done throughout the process. (Ulrich & Eppinger, 2016, pp. 139–140)

The following subsections will explain two crucial concepts. First, *axiomatic design* will be explained in section 2.1.1, which is a design theory needed to be understood to decide what a good design is. Thereafter, a concept generation method and representation called *enhanced function-means* (EF-M), which is used throughout the thesis and which is based on the principles of axiomatic design, is explained in section 2.1.2. In this section, the software tool used in the thesis will also be explained as it is mainly an EF-M modelling tool.

2.1.1 Axiomatic Design

Axiomatic design is a design theory developed by Suh in the 1990's and is one of the most widely spread design theories (Wang et al., 2020, p. 36). Axiomatic design heavily relies on two axioms. An axiom is a "proposition that commends itself to general acceptance" or "a well-established or universally-conceded principle" ("axiom, n." 2021) and, in other words, is stated without proof. By assuming that the axioms of axiomatic design are true, several logical theorems following from their definitions are then true, for instance the information axiom leads to unquestioned engineering practiced like *minimise parts* and *minimise tolerances* (Benavides, 2012, p. 20). The two axioms of axiomatic design are (Suh, 1990, p. 47):

Axiom 1 *The Independence Axiom*
Maintain the independence of functional requirements.

Axiom 2 *The Information Axiom*
Minimise the information content of the design.

The independence axiom means that the functional requirements should be independent of each other. This means that functional requirements should be defined without consideration of other functional requirements. If two or more functional requirements are depending on one another, they are redundant and can be reduced to a single functional requirement. Functional requirements that are not independent are called "coupled". Ideally, or an optimal design, is given by the minimum number of functional requirements needed to describe the design problem at hand. Also, the functional requirement should be defined in such a way, that adjusting a functional requirement's corresponding design solution does not affect the other functional requirements (Suh, 1990, p. 48). This means that a design solution can only be assigned to one functional requirement, since the functional requirements will be coupled otherwise. If a functional requirement has more than one design solution, the design is either redundant or the functional requirements are coupled (Suh, 1990, p. 58). This means that a functional requirement can only have one solution and a solution can only be assigned to one functional requirement. (Suh, 1990, pp. 37–39, 47–48)

The information axiom deals with the information content of the design and tells us to minimise it. Reducing the information of a design reduces the design's complexity, since basing a decision on less information or reducing redundant information will make it easier to make an informed decision, if we assume that the necessary information is present. In reality, the designer starts with the independence axiom when generating concepts and then applies the information axiom to screen the ones proposed (Suh, 1990, p. 67). (Suh, 1990, pp. 64–65)

Axiomatic design classifies the design process into the four domains: customer domain, functional domain, physical domain, and process domain. The order of the domains is important, since a domain defines what is needed in the next domain at the same time it represents the solution to the previous domain. (Benavides, 2012, pp. 79–80)

The design problem starts in the customer domain, which defines the customer and user needs of the product or system. In the functional domain the problem is characterised into a solution neutral environment using functional requirements and constraints. Typically, the functional requirements are represented in a hierarchical functional structure. The functional domain now represents what a product needs to do or be, and this is mapped in the physical domain to solutions of the functional requirements. One can say that the functional domain and the physical domain represent the "what" the "how" respectively. The process domain is the manufacturing and production domain and defines how to produce the product defined in the physical domain. (Benavides, 2012, pp. 103–105)

2.1.2 Enhanced Function-Means (EF-M)

Functional structures can be modelled and represented in many ways (Hubka & Eder, 1988, p. 76). One of these is function-means (F-M) modelling, which originates from about the late 1970's. It uses a hierarchical tree structure to model the architecture of a product, by modelling the product's functions together with the functions' intended solutions. F-M modelling was further enhanced into the enhanced function-means (EF-M) model by Schachinger and Johannesson (2000) through the introduction of constraints. (Müller et al., 2019)

In the coming paragraphs, the EF-M modelling approach will be described, and the reader can use Figure 2.5 as a reference example of an EF-M tree as support. Constraints will also be explained here, but then not treated further since they lie outside the scope of the thesis. The definition of EF-M modelling provided below will be the concept generation method of choice throughout the thesis. The definition is mainly taken from Müller et al. (2019) but also from Müller et al. (2020), Johannesson and Claesson (2005), Schachinger and Johannesson (2000) and Levandowski et al. (2014).

In EF-M modelling, the functions are represented as functional requirements (FRs). Each functional requirement is solved by a design solution (DS), which represents a technical or technological solution to the functional requirement. To increase specificity, the design solutions can be divided into further functional requirements and design solutions as needed. Functional decomposition is made possible this way. Typically, a product architecture is represented by starting with a top-level functional

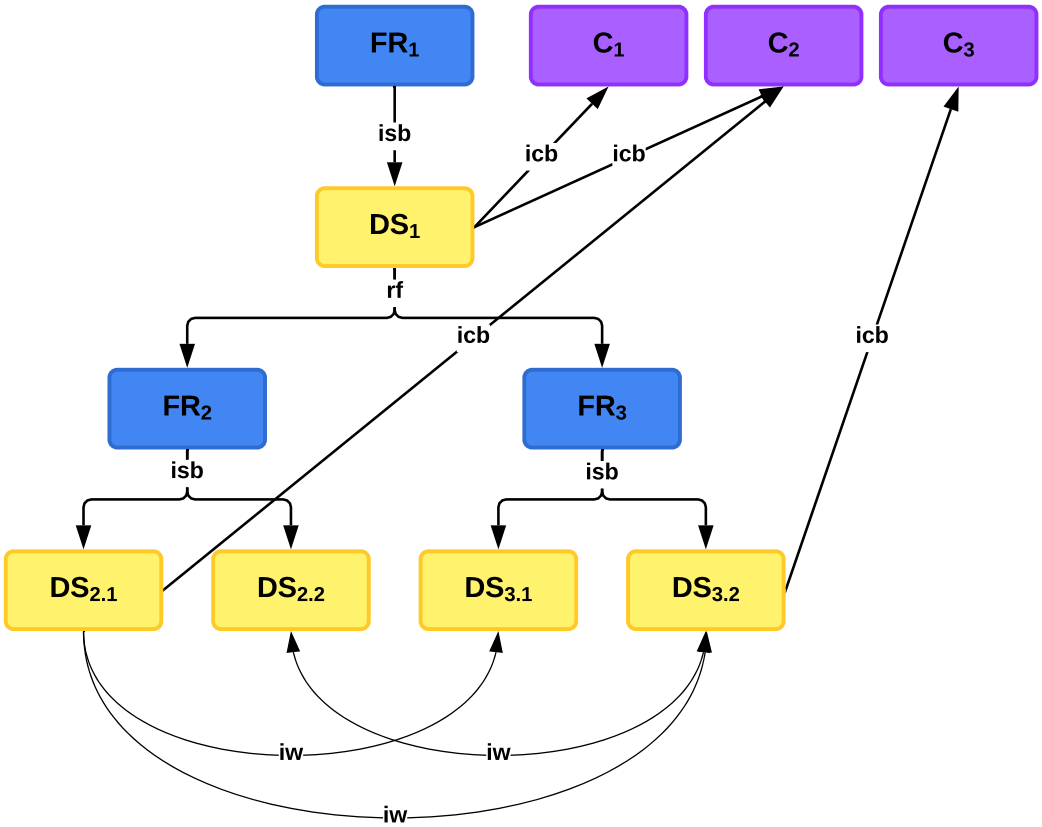


Figure 2.5: An example EF-M tree with frequently occurring elements modelled. Four concepts are modelled at the same time, by having two functional requirements with each two design solutions. These concepts have been expanded into separate EF-M in Figure 2.6. The colour scheme of blue functional requirements (FRs), yellow design solutions (DSs) and purple constraints (C) will be used throughout the thesis.

requirement, which is described as the main function or functions of what is modelled, and which is solved by a top-level design solution, which typically is the product that is modelled. Then, below that, the EF-M tree is functionally decomposed into multiple branches, all the way down to the required level of detail. An example can be a washing machine, which may have its top-level functional requirement and design solution as "FR: wash clothes" and "DS: washing machine" respectively. These would be decomposed into further functional requirements – for instance "FR: loosen dirt", "FR: lead away dirt", etc. – and corresponding design solutions.

A design solution can be parent of multiple functional requirements, but each functional requirement can only be solved by one design solution according to axiom 1 of axiomatic design (see section 2.1.1). If multiple design solutions in fact are assigned to a functional requirement in an EF-M tree – as can be seen in two cases in Figure 2.5 – then these design solutions are to be seen as alternative solutions. This means that, by selecting one of the design solutions as the solution to the functional requirement, one concept is yielded, whereas by selecting another, another concept

is yielded. Since there are two functional requirements with two design solutions each in the EF-M tree in Figure 2.5, the total number of concepts represented in the tree is four ($2 \cdot 2 = 4$). To emphasise what is meant, these four concepts have been modelled in separate EF-M trees in Figure 2.6.

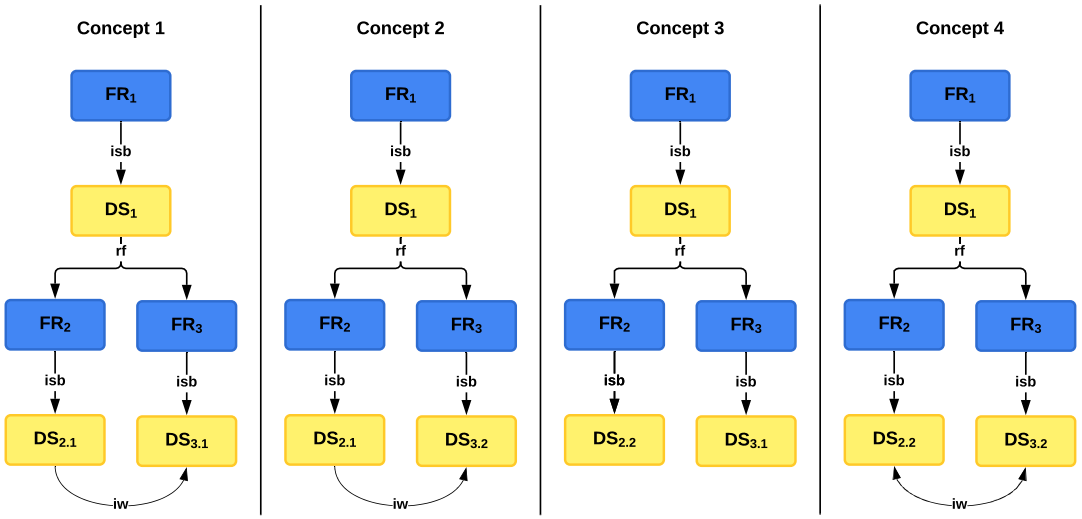


Figure 2.6: The concepts represented in Figure 2.5 modelled in separate EF-M trees. To generate all possible concepts, one design solution at a time is selected for both FR_2 and FR_3 until all permutations have been created. Note that the iw connections are exactly the same as in Figure 2.5. The constraints have been left out and the concept naming is entirely arbitrary.

The main enhancement that EF-M brought compared to F-M is that the non-functional requirements are separated from the functional requirements into constraints (C). The constraints constrain the design space by describing "what a design solution must, or must not, be or do" (Müller et al., 2019, p. 504). They limit the design parameters and the behaviour of design solutions, and an example can be the requirement to fulfil a specific torque or a modularity constraint such as "easily maintainable". Thus, for a design solution to be considered a viable option it must not just solve the functional requirement it is assigned to, but it must also fulfil one or multiple constraints that may or may not be assigned to it.

Different parts of an EF-M tree can be encapsulated by dividing the tree into configurable components (CC). Configurable components are not allowed to share any elements with other configurable components but can be divided into further configurable components. They are visualised in the EF-M tree by surrounding the including functional requirements and design solutions with a rectangle. By doing this, combinable design solutions can be identified and highlighted. The interactions between the configurable components can also be analysed by looking at the iw connections (explained below). Configurable components help support the initiation of platform design, set based design and modularisation at an early stage in the product development process. Configurable components will be treated to a limited extent in this thesis but, however, can occur in various figures.

To indicate connections between the elements of the EF-M tree, certain arrows and lines of multiple types are used. Four types of arrows can be seen in Figure 2.5 – namely isb, rf, rcb and iw – and these are explained in Table 2.1. In the literature, people tend to use the ones relevant for the current situations, and many come up with their own types in order to model a certain situation or satisfy a particular need (for example, Johannesson and Claesson (2005) provide a list of thirteen different types). The arrows and lines are denoted by an EF-M specific denotation which consists of a lowercase abbreviation of one or multiple verbs followed by a preposition. The abbreviations are set up so that when one starts with the element that the arrow or line originates from, follows this by the spelled-out abbreviation, and then ends this with the element which is the destination of the arrow. Now a description of the two element's relationship is yielded. For example, the connection "$\mathrm{FR}_1 - isb \rightarrow \mathrm{DS}_1$", which can be seen in Figure 2.5, is supposed to be read out loud as "FR_1 is solved by DS_1".

Table 2.1: The EF-M specific abbreviations hitherto seen (in for instance Figure 2.5) in their spelled-out form and particular usage.

Abbreviation	**Spelled out**	**Usage**
isb	is solved by	from FR to DS
rf	requires function	from DS to FR
icb	is constrained by	from DS to C
iw	interacts with	from DS to DS

The iw connections specifies an interaction between two design solutions. Other names for these connections found in the literature can for instance be "interactions", "dependencies" and "interfaces", which are also exactly what iw connections try to model. However, the literature evolving around EF-M seem to have settled on the term "iw". The industry has a need of managing interfaces between components and design solutions at an early stage in the design process. The iw connections are EF-M's way to manage and model interfaces. The interactions represented with iw's can be of many types and have different directions. For instance, Pimmler and Eppinger (1994) use the categories (or types, or domains) spatial, energy, information and materials. The direction, which is denoted by the arrows' respective arrowhead/-heads, is for mechanical connections generally bidirectional (two-way), since if one component has a mechanical connection to a second component, the second component obviously also has the same connection with the first component. Other terms for mechanical connections can for instance be spatial and physical. The type, or types, of interaction represented by iw connections are generally not given in the literature, yet both single directional and bidirectional arrows occur in the EF-M trees. Three examples are figure 2 in Levandowski et al. (2014), figure 3 in Isaksson et al. (2016) and figure 2 Müller et al. (2019). This can be seen as a disadvantage of the current state of EF-M and is likely an effect of the fact that iw connections are not well-defined.

For the rest of the thesis, the isb and rf abbreviations and arrows will be heavily used as they connect functional requirements and design solutions, and these elements occur every EF-M tree. However, they will rarely be denoted outside of this chapter. Since constraints will not be treated in this thesis, as previously mention, the icb

abbreviation and arrows will also not be used further. The iw abbreviation and arrows will however be heavily used, since these connections are the main focus throughout most of the thesis, which is also why they were elaborated on above. It will also be rare for the types of the arrows and lines to be denoted in figures as for example in Figure 2.5. The reader will have to rely on the placement of the connectors and which elements they do connect to distinguish their type. One helping factor can be that isb and rf connectors are typically drawn as straight lines, which either travel in a horizontal or vertical direction, but never in both, and they are missing arrowheads. The iw connections, on the other hand, are typically drawn as upside-down arcs with arrowheads.

2.1.2.1 Design Solutions Possible for iw

The iw connections are not well defined in the literature. For instance, the literature does not specify to which design solutions iw connections are possible more concretely than that they appear between design solutions. When developing a tool where iw's are treated, it is however important to clearly define this, since the user should, ideally, just be presented those alternatives that are possible. Even if this is ignored and the user is presented every design solution in the iw interface, this issue still needs to be investigated to see when potential errors might occur and when (and if) error messages need to be raised.

Only restriction on design solution selection during iw creation has been found in the literature. Schachinger and Johannesson (2000) limit iw connections to only be created between design solutions that have the same parent and are on the same hierarchical level in the tree. This can, however, be seen as an unnecessary restriction, since the ability to create iw's in an EF-M tree that has different levels of detail in different branches is severely restricted. For example, let us say that there is a design solution DS_1, which is a leaf node in a very detailed branch of an EF-M tree. And in a less detailed branch of the same tree, there is a design solution DS_2, which is also a leaf node, but not as far down the hierarchical levels as DS_1. Then it is not possible for the user to create an iw between DS_1 and DS_2 if the Schachinger-Johannesson restriction is implemented. This can of course be solved by selecting a parent design solution (DS_3) of DS_1, which is possible for iw creation based upon the Schachinger-Johannesson restriction, and then creating an iw between DS_2 and DS_3. But then, DS_3 and all its children are seen to interact with DS_2, even though this might not be the case. Maybe it was just the component DS_1 that interacts with DS_2. Not only does this confuse second reader of the EF-M tree, but this will propagate trough the DSM extraction into any analyses that are run. Due to the lack of similar statements in the literature (especially in the most recent), this restriction will be ignored in the implementation of iw modelling in this thesis.

Due to the lack of conventions and clear definitions of iw connections in the literature, this thesis will allow all design solutions for iw creation except a minor few, which meet some key criteria. The design solutions that fall on these key criteria are deemed to lead to illogical iw modelling if they should be allowed. The first criterion is that there exists little sense in allowing iw connections between design solutions that are children to the same functional requirement and thus alternatives relative to

one another. This is because alternative design solutions do not exist in the same product instance and this way they cannot possibly interact. The second criterion is that iw connections between design solutions in the same branch, which in other words are in a parent-child relationship, do also not seem reasonable. This is because the EF-M tree itself indicates these interactions already. If a design solution is split into functional requirements, which are then solved by further design solutions, it is clear that some form of connection exist between the parent design solution and the children design solutions. If these iw's are allowed, they should henceforth always be active and they would then be redundant.

Hence, for this thesis the design solutions that are considered to be relevant for iw creation are all design solutions that are in the same a project and that are not

- a parent of
- a child of
- an alternative relative to

a reference design solution.

2.1.2.2 An EF-M Modelling Software Tool

A functional modelling software tool has been developed at the department of Industrial and Materials Science (IMS) at Chalmers University of Technology which implements EF-M modelling. This software, which will in the thesis be called either *the EF-M tool* or simply *the tool*, provides a good ground to build on and to include new needed functionalities from. This section will explain the state of the tool at the start of the thesis.

The EF-M tool is a web-based tool, meaning that it has the advantage of not requiring an installation since the user can run the software directly from the web-browser. This fact also allows for easy distribution of the tool, since the owner of the tool can just provide it from a server if distribution would be wanted and users can globally access it from a web-address just like any other web-page available on the internet.

The tool features a log-in system, a project management view, an EF-M modelling view and a concept view. These are accessed in the order given, i.e., the user logs in, creates a project, creates an EF-M tree and from there generates concepts. The log-in system is self-explanatory, and houses features such as account creation and logging in. When logged in, the user faces the project management view, where a form for project creation and a list of previously created project is presented. When creating a project, the EF-M modelling view is entered where a top-level design solution has automatically been created with the same name as the project. This is more reasonable rather than naming the project after a top-level functional requirement because it would be more intuitive to the users to have a list of projects that are all named after the respective products they represent and not after the functions they are solving. Of course, this statement does not account for the fact that users might actually want to start their EF-M tree with a functional requirement, or to the fact that a project name does not have to be linked to the name of one of the modelling elements. However, this is not a huge problem since there are ways around this, for instance, by creating a project with a desired name and thereafter renaming the top-level element.

It is in the EF-M modelling view where EF-M trees are created, edited, moved and deleted. In this tree view, functional requirements, design solutions and constraints can be created and put in relation to each other. The user can toggle between two modelling directions, namely *top-down* and *left-right*, where the first is default. A screenshot of the tool running in a web-browser is shown in Figure 2.7.

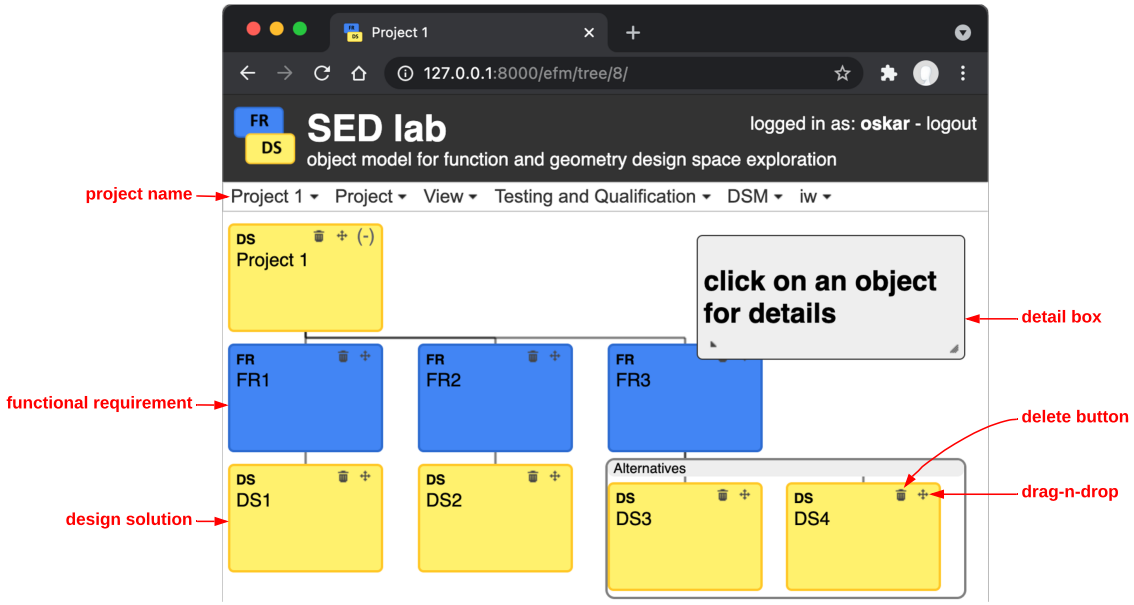


Figure 2.7: A screenshot of the tree view in the tool, which is running on a local server in the Google Chrome web-browser on a Mac. Particular elements of interest are pointed out in red.

The creation and editing of the EF-M elements happens from the so called "detail box". The detail box is always visible in the right-hand side of the view of tree view. When the user clicks on an object such as a functional requirement or a design solution, the details about that object appears in the detail box as shown in Figure 2.8. In this view, the user can do things like changing the name of the object or adding an optional and potentially much longer description. The deletion of elements happen on the elements themselves by clicking on a trash can symbol, and moving them is either done by clicking and dragging them to their new parent or by deleting them and recreating them at the proper location.

As described in section 2.1.2, many different alternative designs can be simultaneous represented in an EF-M tree. The tool offers a concept view where all the different alternatives can be generated and viewed. This is done as also described in section 2.1.2: by, one at a time, mapping one design solution to each functional requirement when multiple design solutions are children of a single functional requirement. When entering the concept view for the first time or when resetting the concepts, all concepts are automatically generated and presented in a list in the concepts view.

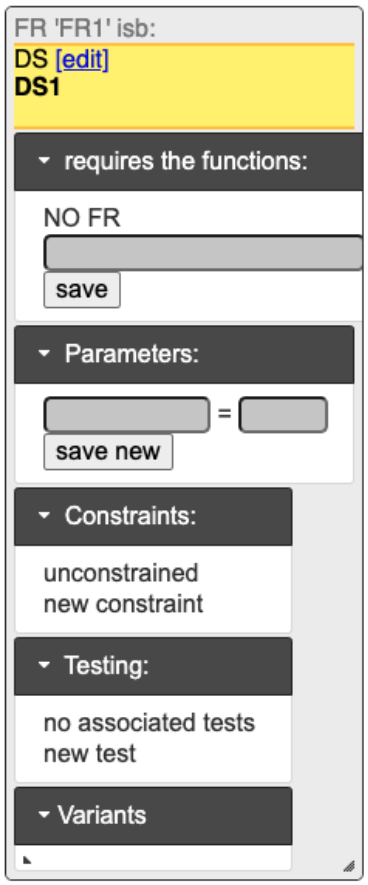


Figure 2.8: The detail box in the EF-M tool, which is used to show and edit design solutions and functional requirements.

2.2 Concept Evaluation

> *Concept [evaluation] is the process of evaluating concepts with respect to customer needs and other criteria, comparing the relative strengths and weaknesses of the concepts, and selecting one or more concepts for further investigation or development.*
>
> Ulrich and Eppinger (2016, p. 161)

The step after concept generation is concept evaluation. Here the generated concepts from the previous phase (see section 2.1) are to be compared and evaluated. This lays the ground for the concept selection of one or multiple concepts for further development. Customer requirements, among other criteria, are here used as points of reference to rank the concepts relative each other. The goal is to reduce a large set of concepts, but new alternatives can be created by combining two less qualified concepts to one new and thus temporarily increasing the number of currently considered alternatives. However, through either elimination or combination of concepts and while concurrently refining the remaining alternatives, the number of considered concepts are reduced until one or a few of the most dominant and viable alternatives are selected for further development. (Ulrich & Eppinger, 2016, pp. 146–147)

A two-step method is used to support these endeavours, where the first step is *concept screening* and the second *concept scoring*. Both are matrix-based methods, where decision matrices are used to guide the developing team in rating, ranking and selecting between the alternatives. It is important that the entire team has insight in all decisions in creating the matrices and that as many iterations as needed for arriving at a reasonable result are used. The matrices function as visual guides for the team as they guide their attention to the evaluation criteria such as the customer needs. Both methods are to be done in six steps: (1) prepare the matrix; (2) rate and (3) rank the concepts; (4) combine and improve the concepts; (5) select one or more concepts; and (6) reflect on the results and the process. Concept screening is a coarse sieve for reducing alternatives by removing the least viable concepts. The subsequent concept scoring works as a finer sieve aimed at more carefully choosing between the remaining alternatives. The screening and scoring methods will be explained in the coming sections, but are also shown in Figure 2.3. (Ulrich & Eppinger, 2016, pp. 151–152)

Concept screening, also called *Pugh concept selection*, is used to quickly reduce the number of alternatives up for consideration. A matrix on the same structure as the example provided in Table 2.2 is set up. All criteria to be used for evaluation are mapped to one row each. Here it is important that all criteria are of similar importance since no weighting is used here. A reference solution is appointed by the group, which can either be one of the concepts or an existing solution (e.g., the best

Table 2.2: An example of a concept screening matrix filled with example data. Concept C has been used as a reference, thus marked with "ref.". The concepts are rated better than (+), equal to (0) or worse than (−) the reference. Note that concept C for this reason has all zeros. The net score is calculated as the number of pluses minus the number of minuses. The concepts are ranked based on the net score. Note that concept B and C are both ranked second. As can be seen in the last row of the matrix, the decision was to proceed with the best valued concept (D), not to continue with the worst ranked concept (A) and to combine concept B and C into a new concept called BC for further analysis.

	Concept			
Selection critera	**A**	**B**	**C (ref.)**	**D**
Critera 1	−	0	0	0
Critera 2	−	−	0	+
Critera 3	+	0	0	0
Critera 4	0	0	0	+
Critera 5	0	+	0	−
Sum +'s	1	1	0	2
Sum 0's	2	3	5	2
Sum −'s	2	1	0	1
Net score	−1	0	0	1
Rank	3	2	2	1
Continue	No	Combine	Combine	Yes

benchmarked competitive solution). Each concept as well as the reference are given one column and, in each cell, the respective concept's rating based on the respective criteria is entered as either being better than (+), equal to (0) or worse than (−) the reference. Below this section of the matrix, the pluses, zeros and minuses are summarized, whereafter the net score (number of pluses minus number of minuses) is calculated. Based on this, the concepts can be ranked relative each other and a decision whether to continue or not with each individual alternative can be made. Here, the team can also decide to combine two or more less suitable concepts into one in order to keep the concepts' good qualities while cancelling-out the bad. In the example in Table 2.2, this has been done with concept B and C, which have been combined into concept BC. The process can be done again with a different reference if the team feels the need for this. (Ulrich & Eppinger, 2016, pp. 152–155)

Concept scoring is used to choose between a reduced set of concepts. The method is very similar to concept screening, but the selection criteria are here weighted instead, which accounts for their individual importance. This means that the concepts can be ranked based on additional criteria, since they now do not have to be of similar importance. A finer scale is also used, typically a five-point scale (1 = much worse; 2 = worse; 3 = equal; 4 = better; 5 = much better) instead of the three-point scale used for screening (+, 0, −). An example of a concept scoring matrix, which is building upon the one used in Table 2.2, is presented in Table 2.3. When preparing the matrix, the team assigns a value, either from a range or as a percentage, for

Table 2.3: An example of a concept scoring matrix filled with example data. The concepts used are the ones from the example in Table 2.2, where concept B and C has been combined into concept BC. Concept A has been used as a reference, thus marked with "ref.". The concepts are rated on a scale from 1 to 5 (1 = much worse; 2 = worse; 3 = equal; 4 = better; 5 = much better) and the weighted score is given by the rating multiplied with the weight of the selection criteria. The concepts are the ranked based on the total score, which is the sum of the weighted scores. Here can be seen that concept A scored best, and has therefore been selected for continued development.

		Concept			
		A (ref.)		**BC**	
Selection criteria	**Weight**	**Rating**	**Weighted score**	**Rating**	**Weighted score**
Criteria 1	25%	3	0.75	1	0.25
Criteria 2	10%	3	0.30	5	0.50
Criteria 3	35%	3	1.05	2	0.70
Criteria 4	5%	3	0.15	2	0.10
Criteria 5	5%	3	0.15	1	0.05
Criteria 6	20%	3	0.60	3	0.60
	Total score		3.00		2.20
	Rank		1		2
	Continue		Yes		No

each selection criteria indicating their individual importance and weight. When selecting a reference, it is usually best to select a new reference for each evaluation criteria if multiple concepts are compared. This is because if one does not select a concept that averages all criteria, all other concepts will lie on either the better than side or on the worse than side of the reference. This makes the scale go from a five-point scale to a three-point scale (either just $\{1, 2, 3\}$ or just $\{3, 4, 5\}$ can be used). If just a few concepts are considered, like the two in the example in Table 2.3, this is not a problem. The concepts are assigned a rating based on how well they fulfil the respective criteria. This rating is then multiplied with the weight of their respective criteria to generate the weighted score. The concepts' sum of their weighted score is then what constitutes the basis for the ranking. (Ulrich & Eppinger, 2016, pp. 156–159)

In the following subsections, two concepts that are used in this thesis to support the concept evaluation is presented. The first is *design structure matrix* (DSM), which is used to represent product interfaces on a matrix form and which will be described in section 2.2.1. The second concept is *non-functional requirements*, which are very important for concept evaluation early in the design process as mentioned in chapter 1. Non-functional requirements will be compared with functional requirements in section 2.2.2.

2.2.1 Design Structure Matrix (DSM)

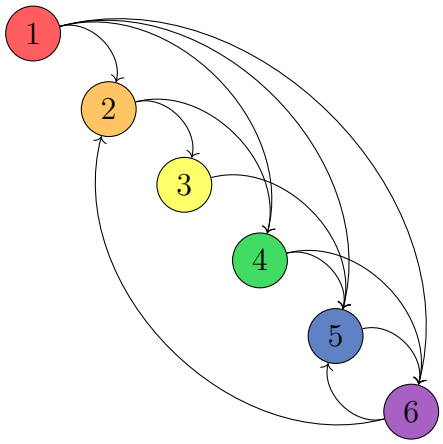
The design structure matrix (DSM) is a modelling tool used to represent system elements together with their relationships. The term DSM was coined by Don Steward in the 1970's (Eppinger & Browning, 2012, p. 12), but similar methods has been in use in various US aerospace companies since the 1950's or 1960's (Eppinger & Browning, 2012, p. 20). A DSM is always a square matrix, but there are also support for rectangular matrices in the literature which are then called domain mapping matrices (DMMs). In other words, the DSM represents the interactions and the relationships between the same set of elements, which means that the matrix will be square and have empty diagonal elements. Rectangular matrices are used to map one form or type of elements to another. These can of course be square if the number of elements in the rows and columns are equal, but this is not guaranteed. (Browning, 2016)

The cells of the DSM represent the relationships between the elements. The relationships, which can for instance be dependencies, interfaces or interactions, are denoted by the contents of the cells. One separates binary DSMs from numerical DSMs by identifying whether the relationships are marked in the cells with a symbol or a number. Often, other types of symbols or colour-coding are used to highlight patterns and other types of information. (Browning, 2016)

There are two conventions on how to read a DSM, namely the *inputs in columns* (IC) and *inputs in rows* (IR) convention. For DSMs following the IC convention, an element has its inputs in its columns and its outputs in its rows. That is, reading along the columns of an element, the reader is informed from which other elements the element has incoming relationships with. This is exemplified in Figure 2.9. A

	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6
Element 1	1	×		×	×	×
Element 2		2	×	×		
Element 3			3		×	
Element 4				4	×	×
Element 5					5	×
Element 6		×			×	6

(a)



(b)

Figure 2.9: An example DSM following the IC convention in (a), with its corresponding node diagram representation in (b). This example has been taken from Browning (2016).

DSM is in fact equivalent to a directed graph, which is exemplified in Figure 2.9b and it might be easier for the reader to compare that with Figure 2.9a. The IR convention is the exact opposite of the IC and thus the two conventions are merely the transpositions of each other. Both conventions occur in the literature and in the applications of DSM, due to its long history and its diverse areas of implementation. If nothing else is mentioned, this thesis will use the IC convention. (Browning, 2016)

DSMs have gotten increased popularity over the years and has found usage in several different areas (Browning, 2016). A DSM can represent the relationships between almost anything. For example, if the placeholder names (Element 1, Element 2, etc.) in the example DSM in Figure 2.9a are replaced by components in a product, people in an organisation or activates in a process, then the elements in the cells of the matrix can represent the component interactions, the communications between people or the information flows between activities respectively. These three examples would then be examples of a product architecture DSM, an organisation architecture DSM and a process architecture DSM respectively (Eppinger & Browning, 2012, p. 1).

In some situations, there might exist multiple different types of interactions that are needed to be modelled. Instead of doing this in different DSMs, it is possible to split each cell in multiple cells, each representing a different DSM. Pimmler and Eppinger (1994) has developed a method for just this and propose four interaction types to consider when modelling. These are: spatial (S), which represent physical interactions; energy (E), which represent energy transference; information (I), which represent information exchanges or signals; and materials (M), which represent materials exchange. These interaction types can be called categories or domains as well, and the relevance of each depends on the problem at hand. All four domains

Table 2.4: An example of how multiple DSMs can be represented in a single DSM. (a) Each cell is split into four and each has been designated to serve one interaction category of spatial (S), energy (E), information (I) and materials (M) as described by Pimmler and Eppinger (1994). See the legend in (b). When using numerical values for the interactions, as is being done here, the empty cells are cells with four zero values.

(a)

	A	B	C
Component A			2 -2 0 0
Component B			2 0 0 2
Component C		1 0 0 0	

(b)

S	E
I	M

can be represented in the same DSM by splitting each cell of a regular DSM into four cells and mapping each cell to one domain. This is exemplified in Table 2.4.

DSMs are well defined in the literature and there exist many methods, processes and techniques that use these matrices as basis for further analysis. In this thesis, those related to product architectures are most relevant. Two of these will be elaborated on in section 2.2.1.1 and 2.2.1.2, namely the complexity calculation that is possible from DSMs and the risk assessments possible from the change propagation method.

2.2.1.1 Structural Complexity

A method for calculating the structural complexity, which is one of three main complexity dimensions that have evolved in the context of engineered systems and characterises the product architecture, has been developed by Sinha et al. (2013). The structural complexity gives a measure for the architecture's complexity early in the product development process and it is calculated from an assigned value for the components' individual complexity and a DSM containing values for the complexity of the component's interactions. It is possible to store the components' individual complexities in the DSM's otherwise empty diagonal elements, and thus calculate the structural complexity from a single DSM.

The structural complexity is calculated from three complexity terms. Let A represent the DSM. The first term C_1 is a sum of all the components' individual complexities. This will then be the sum of the diagonal elements in the DSM.

$$C_1 = \sum_i A_{i,i} \tag{2.1}$$

The second term C_2 accounts for the number and complexity of the component interactions, which is calculated as the sum of all non-diagonal elements of the DSM as (2.2). This can in turn be calculated as the sum of all elements of the matrix minus the sum of the diagonal elements, which is given by (2.1).

$$C_2 = \sum_i \sum_{j \neq i} A_{i,j} = \sum_{i,j} A_{i,j} - \sum_i A_{i,i} = \{(2.1)\} = \sum_{i,j} A_{i,j} - C_1 \tag{2.2}$$

The third term C_3 is the effect of architecture or the arrangement of the interfaces. It is calculated according to (2.3) as the sum of the DSM's singular values σ divided by the number of components n, which is equal to the number of rows and columns in the DSM.

$$C_3 = \frac{1}{n} \sum_i \sigma_i \tag{2.3}$$

The structural complexity C can then be calculated as

$$C = C_1 + C_2 C_3 = \sum_i A_{i,i} + \sum_i \sum_{j \neq i} A_{i,j} \cdot \frac{1}{n} \sum_i \sigma_i \tag{2.4}$$

2.2.1.2 Change Propagation Method (CPM)

With the change propagation method (CPM) from Clarkson et al. (2004), the risk of a product design represented in a DSM can be calculated. With risk, the ability of a product to respond to change in its components or parts is meant: a high risk means that changes do not happen easily. The method is particularly useful to evaluate how a complex product, or a product with complex component interactions, respond to change and how easily or not changes to individual parts can be made. As Clarkson et al. (2004) write:

> As all parts of a design are connected at least to one other part, design changes can also be connected. If part A is changed then part B might need changing. A change to part B leads to a change in C, which in turn might be connected to part D. (p. 788)

Product designers want to make sure that changes can be made with relative ease in components that are likely to be subject to change during the product's life cycle. Such change can for instance be new technology introduction either to increase the product's functionality or to respond to a sudden demand. For other components, a high risk is accepted since a change in this area is not very likely to happen. CPM accounts for this by calculating risk as

$$risk = likelihood \cdot impact$$

where *likelihood* is the chance that a change will happen and *impact* is the severity of such change. The risk can be visualised in another DSM called a product risk matrix, where each cell contains a coloured rectangle whose base and height represents the likelihood and impact respectively, which means that risk is visualised by the rectangle's area. (Clarkson et al., 2004)

2.2.2 Functional & Non-Functional Requirements

A product can be described based on *functional requirements* and *non-functional requirements*. Functional requirements describe the function or behaviour of a system. Developed concepts are evaluated based on functional requirements by tools like CAD, finite element analysis (FEA) or computational fluid dynamics (CFD), which are tools that require a geometry (e.g., CAD-models) and thus a very detailed concept. Other tools like Modelica or Simulink can be used for evaluating functional requirements before such design embodiment.

Non-functional requirements, on the other hand, describe the operation of a system. Examples of non-functional requirements can be modularity, maintainability, complexity, risk, scalability and new technology integration ability. Non-functional requirements are sometimes called *ilities* since they often end with "-ility" as can be seen above. They can be analysed with relative ease for a very detailed concept. This is however typically achieved late in the product development process when it is also more costly and difficult to perform changes in the design. (Worinkeng et al., 2015)

The functional requirements described in this section should not be confused with the elements of EF-M that have the same name (see section 2.1.2). In the context of EF-M, the functional requirements can describe non-functional requirements as well as the functional requirements discussed here. The functional requirements of EF-M should more be seen as "functions" and functional requirement might sometimes be a misleading word. The reader has to pay close attention to which is meant throughout the thesis, as no specifications of which functional requirements are currently treated are given other than the context.

2.3 Reflection on the State of the Art

A gap can be seen in the literature between non-functional requirements and the concept evaluation during the concept development phase of the product development process. There exists a paradox in the product development process – or rather in any design or problem-solving process – which is the fact that, with time, the information content of the problem increases while the possibility to carry out changes decreases. This is called the design process paradox and is illustrated in Figure 2.10. This affects all parts of the process and hence also the non-functional requirements. The ease of analysing them occurs at first when the product or concept is very detailed or even manufactured, but this is also when it is too late to do anything about an unsatisfactory result of such analyses. Existing tools, like CAD, cannot represent non-functional requirements in an efficient way, especially because they are based on an existing geometry which it is too early in the process to be able to rely on.

Many industries are currently transforming through electrification and automation, so also the aerospace industry. These processes, along with increase environmental demands on products and producers, lead to introduction of new technologies and functionalities. Companies are making these introductions into existing product platforms and architectures to be able to keep up with these changes. In such situations it is crucial to be able to evaluate the compliance with both existing and

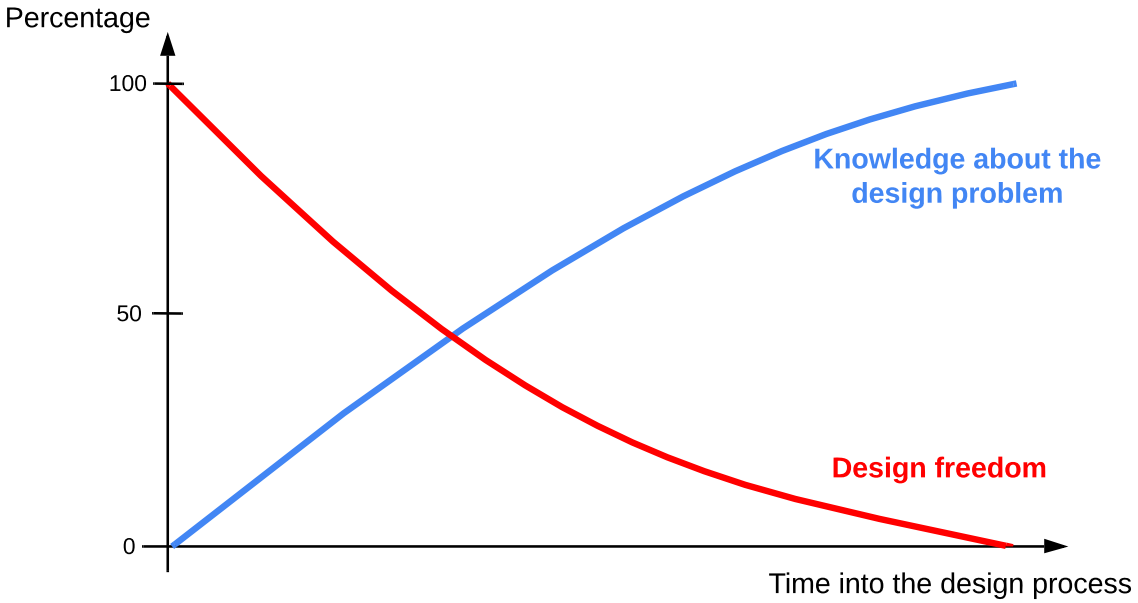


Figure 2.10: The design process paradox. (Ullman, 2010, p. 19) The typical design process, as well as product development process, is characterised by that the freedom in design decreases as knowledge about the problem increases.

future architectures. The same goes for the risk of a particular product layout, since a wrong decision might lead to the developed architecture being noncompliant with technologies introduced in the future.

New methods or tools are thus needed to be able to assess concepts based on non-functional requirements already in the evaluation stage of the concept development process. This can be done by utilising the existing EF-M software tool explained in section 2.1.2.2 to represent product interfaces and component interactions as iw connections. These iw's can then be exported in the shape of DSMs which would take advantage of existing methods of DSM bases analyses. Furthermore, there exist a potential for the tool to be connected to external simulation software, which would mean that concepts can not only be evaluated based on non-functional requirements through DSM, but also based on performance criteria such as induced torque or resulting angular velocity.

3

Method

The goal of this thesis can be summarised as to fill the missing pieces of the literature explained in chapter 1 and in section 2.3. The method used to tackle this problem will be described in this chapter.

A solution for these problems were seen, when considering enhancing the existing EF-M tool explained in 2.1.2.2 with functionalities that correspond to the missing links in the literature and the problem statement. This was also selected as the decided approach. The problem was divided into four functionality levels that were to be implemented in the tool. These are presented in the coming sections. Results of the development of the functionality levels were to be verified with case studies of either real-world problems received from the industry or of realistic examples. The results will be described in chapter 4, the case studies to verify the results can be found in chapter 5 and the results are then discussed in chapter 6.

The development of the four functionality levels followed the agile software development management process called *scrum* as it is described by Dove and LaBarge (2014). An agile process was needed to maximise the thesis' development efforts during its limited timeframe consisting of 20 weeks. In scrum the entire work is divided into iterations called sprints. Each sprint is of constant duration of a month or less and adds new features to the developing product as well as improving on existing. After each sprint and regardless of whether the development effort is finished, a new sprint is immediately started for the next objective. Hence, each functionality level of the thesis was divided into and developed in four-week sprints. If work was still to be done on the previous functionality, this was added to the backlog which could be picked up in another sprint. In addition to scrum, the four concepts and twelve principles of the *Agile Manifesto* (https://agilemanifesto.org/) were followed since they provide further agility to the development process. These concepts and principles specify a set of rules to follow in order to efficiently arrive at a good prototype or a minimum viable product, which indeed was the goal here. A Gantt chart over the project's timeframe is presented in Figure 3.1.

The four functionalities, and how they intend to contribute to concept development, are described in section 3.1 through 3.4. An overview over the functionalities is presented in Table 3.1 and range from enhancing the tool with missing EF-M functionalities to connect it with external software to bridge between existing technologies. After that, external software used in support of the thesis are explained in section 3.5.

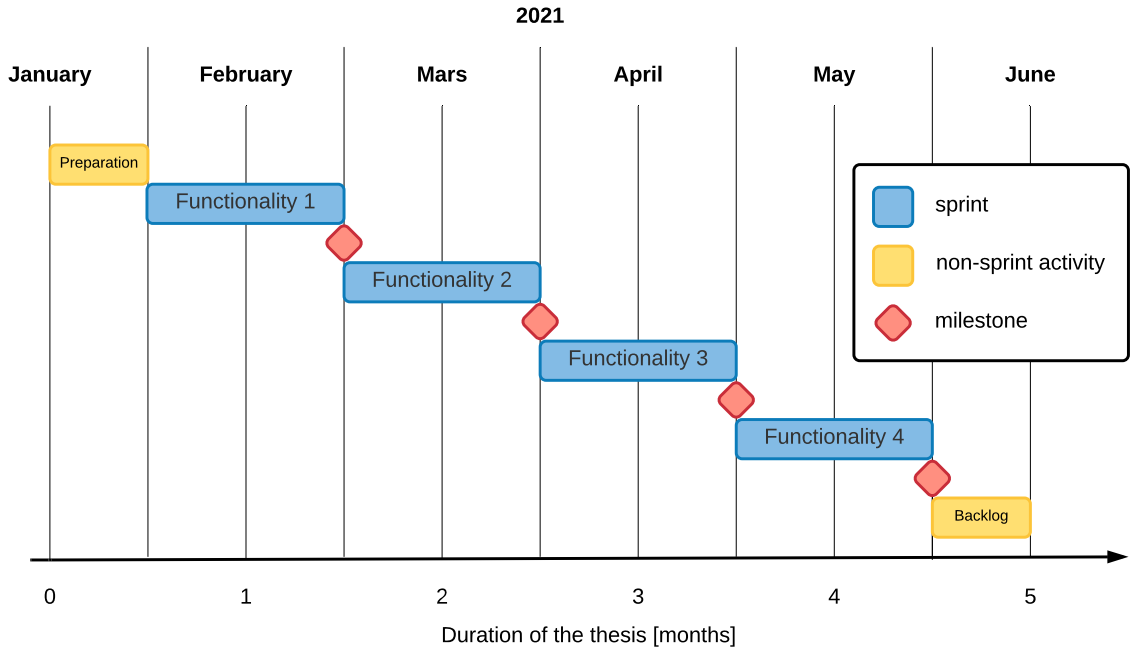


Figure 3.1: A Gantt chart over the thesis' duration. The sprints for each functionality are shown with yellow boxes. Each process ends with a milestone, which is what will be presented in the results chapter for each functionality.

Table 3.1: An overview of the functionality levels to be implemented in the EF-M tool and what they intend to contribute to the current state of the art. What is stated under this contribution is also the reason for and benefits of their specific implementation.

Functionality	Contribution to the state of the art
First level functionality iw visualisation DSM extraction of iw	User interface for modelling interfaces and dependencies of products, as well as shareability of interface modelling, which means possibility to run further analysis in external software of component dependencies.
Second level functionality iw characterisation DSM extraction of iw	Enabling of multidomain representation of product interfaces and of their analyses in external software.
Third level functionality FR library Load EF-M from DSM	Assisted functional requirement creation and preparation of intersoftware connection of modelling elements. Deepens the connection between software that use EF-M and DSM respectively by allowing EF-M to DSM as well as DSM to EF-M.
Fourth level functionality Connection to Modelica	Possibility to evaluate concepts based on performance before design embodiment by running simulations and creating simulation models of the in EF-M represented concepts.

3.1 First Level Functionality

The first level functionality consisted of two subfunctionalities that were to be added to the tool, namely: *iw visualisation* and *DSM extraction of iw*. The iw visualisation constituted the possibility for the user to optically visualise the iw connections in the EF-M tree. In the literature this is typically done using arrows (e.g. Figure 2.5), where an arrow connects the two design solutions which are interacting with each other. The arrowhead or arrowheads then indicates the direction of the iw, i.e., the direction of the interaction. The iw goes from the design solution without the arrowhead, to the design solution with the arrowhead. If an arrowhead exist at both ends of the line, the iw is said to be a "two-way" or "bidirectional" iw, which means that both design solutions mutually interact with each other or have a mutual interface. This functionality will also require a user interface that efficiently manages the creation and deletion of iw connections. This progress will mean that interactions, dependences and interfaces in product architectures can be modelled and represented within the tool, which is something that was missing from the state of the received tool as it was described in 2.1.2.2.

The DSM extraction of iw is the possibility to convert the iw's, which through visualisation are just arrows in the EF-M tree, into a DSM matrix, which can then be exported in appropriate file formats. This way, the user is offered the advantages that both EF-M trees and DSM matrices possess. For instance, the user can explore the design space and generate concept using the EF-M tree and then generate and export DSM matrices for each concept in a matter of seconds for further analyses in other software. These other DSM analyses can be those previously explained such as structural complexity calculation or CPM.

3.2 Second Level Functionality

The two sub-functionalities of the second level functionality was *iw characterisation* and, again, *DSM extraction of iw*. The iw characterisation meant adding the possibility to categorise the iw's into four categories. The iw visualisation will thus be enhanced with the possibility to draw four different iw's in the EF-M tree at the same time, while still being able to distinguish between the categories. For the iw creation, each iw should now be able to be assigned a category and a value, where the latter then can be used to for instance calculate complexity metrics. At the same time, the first functionality level should be kept intact so a switch between iw modelling with versus without categories will be needed.

The DSM extraction of iw from the first level functionality should then also be enhanced with the possibility to extract the iw categories, while keeping the support for the DSM extraction of non-categorised iw. This will be done by splitting each cell of a regular DSM into four cells, after Pimmler and Eppinger (1994), which is described in section 2.2.1. The possibility to just extract one category will also be possible, since not all external software, which would be used for further DSM analyses, do support four-cell matrices.

The development of categorising iw's in this way and the DSM extract this characterisation mean that multiple types of interactions can be analysed and otherwise treaded separately or together. This allows for a more complex analyses of product interfaces and the possibility to not only model product architectures of different types in the tool, but also the interactions of these architectures. For example, by assigning categories such as electrical, mechanical and information, electromechanical product architectures – the software interactions – can be modelled and analysed in the same tool.

3.3 Third Level Functionality

The third level functionality had the sub-functionalities *FR library* and *load EF-M from DSM*. The first sub-functionality was to include a functional requirement library, which should be used to support the user in creating functional requirements. The FR library should contain two different libraries called the *FR-DS library* and the *FB library*. The creation of functional requirements would be supported in two different ways from these two views.

The first view, the FR-DS library, contains a library of combinations of functional requirements and design solutions. The library, which is called the FR-DS library, would serve as a way to provide users with standardised design solutions mapped to functional requirements. From the library, users should be able to create functional requirements and design solutions at the same time. This would support users in modelling in the tool from a design solution perspective, since functional requirements are otherwise created before design solutions. Furthermore, this sub-functionality was also seen as a priming of the tool for the more challenging fourth level functionality, which relies on such library.

The second view, the FB library, was also to contain an additional view, in which the creation of functional requirements alone was to be supported. This support is to be realised by supplying the user with standard phrasing regarding functional requirements. More specifically these functional requirements are in this part of the library given on a verb-object format (e.g., "transfer torque"). The basis for this is a functional basis (FB) created by Hirtz et al. (2002), which also is used as a basis for the naming of this part of the library. This development would support users to phrase functional requirements in an efficient, correct and standardised manner, which ultimately limits ambiguity and prevents the use of different words for the same things (synonyms).

The possibility to load EF-M models from one or multiple DSMs was also to be implemented. This is a functionality that users, who already have products in the form of DSMs or are not as familiar yet with EF-M modelling, will gain from. Potentially, it will also deepen the link to external software that incorporates DSMs.

3.4 Fourth Level Functionality

To connect the tool to the Modelica language was the goal of the fourth level functionality. A connection was to be made between the tool – a functional modelling software – and a simulation of physical systems environment. The intent is to be able to export files containing Modelica models and the possibility to run simulation directly from the tool. This development would mean that concepts, represented in EF-M, could be evaluated based on performance criteria before the embodiment in CAD systems.

3.5 External Software

This section will describe the external software used in support of the thesis. Part of the enhancement of the EF-M tool existed in creating a connection to these external software to expand the available functionalities. The two software CAM and Modelica are described in section 3.5.1 and 3.5.2 respectively.

3.5.1 Cambridge Advanced Modeller (CAM)

The Cambridge Advanced Modeller (CAM) is a software developed by the University of Cambridge. CAM can model and simulate processes, dependences and change in products and systems. It also includes a tool for visualisation of multidimensional engineering data. (University of Cambridge, 2021)

The main use of CAM in this thesis is to perform analysis on DSMs, such as CPM (explained in section 2.2.1.2). One of the functionalities to be implemented is the possibility to export iw connections as DSMs. It is desired to be able to open these DSMs in CAM for further analysis, which means that they need to be formatted in a special way required by CAM. There already exist a functionality to import DSMs from .csv-files into CAM.

At the time of writing there exist two versions of CAM. In this thesis, the second one – CAM 2 – has been used and is inferred whenever mentioning CAM.

3.5.2 Modelica

Modelica is a modelling language to model complex physical systems and developed by the Modelica Association, which is a non-profit, non-governmental organization (Modelica Association, 2021). A modelling language is similar to a programming language but more used for creating and setting up models. Modelica is very flexible, and the type of system modelled can for instance be mechanical, electrical, electronic, hydraulic, thermal as well as any combination thereof. The possibility to represent and simulate multiple different types of systems is one of Modelica's strengths. The Modelica Standrad Library contains about 1 600 predefined models.

There are multiple environments that provide interfaces to model, set up and simulate Modelica models. The one used to support this thesis is OpenModelica (https:

//www.openmodelica.org/), since it is open-source and relatively easy to use. Files with the file extension .mo can for instance be loaded in OpenModelica and the models they contain are then presented in OpenModelica's user interface. The goal of this thesis is to connect the EF-M tool to Modelica and be able to extract Modelica models possible to be loaded in OpenModelica as well as running Modelica simulations of the EF-M models directly from the tool.

4

Results

The expected outcome of the thesis was divided into four functionality levels as described in chapter 3. In this chapter, the results of these four functionality are presented and described.

4.1 First Level Functionality

The iw connections were visualised by arc shaped arrows between the design solutions as exemplified in Figure 4.1. In the top-down view of the tree, the iw's are drawn to and from the middle of the bottom side of the design solutions, whereas for the left-right view this point is then on the middle of the right-hand side of the design solutions.

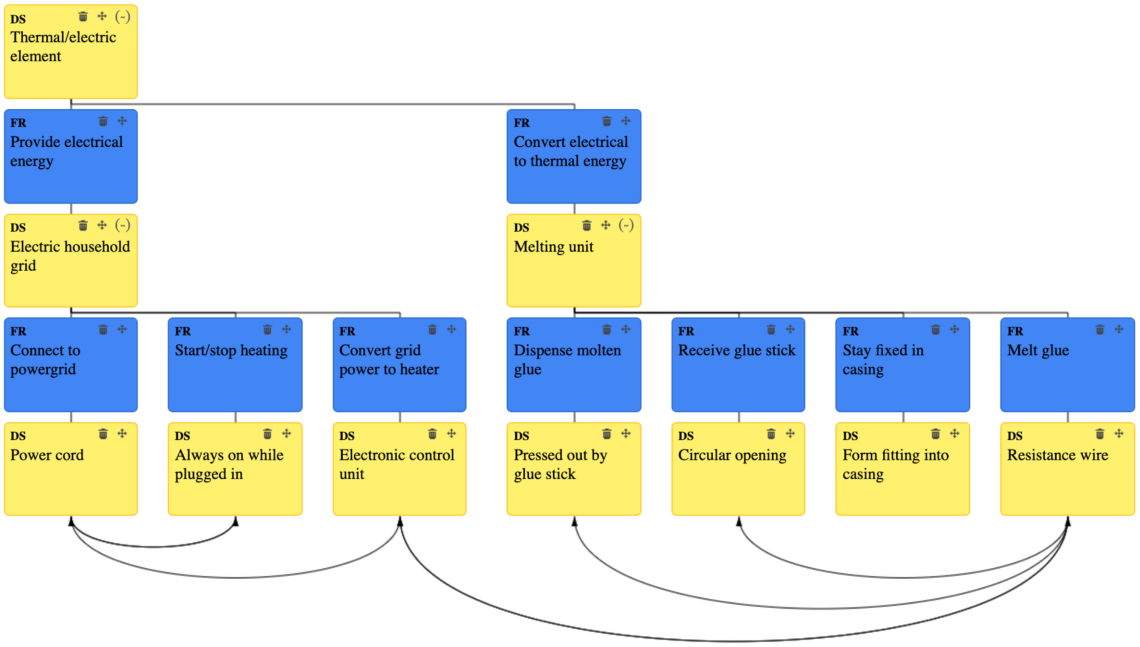


Figure 4.1: The EF-M tree of a thermal/electric element taken from Müller et al. (2019). The iw connections are the arc shaped arrows between the design solutions and are shown here in the tool.

The interface to the user for creating and deleting iw's exists in the detail box. When the user clicks on an object (for instance a functional requirement or a design solution) the details about the object appears in the detail box as described in

section 2.1.2.2. In this box, the user can now also manage the iw connections when a design solution has been selected as is depicted in Figure 4.2. In the section called "interacts with", those design solutions appear, which according to section 2.1.2.1 are possible for iw with the selected design solution. They are shown next to each other in two different columns called "to" and "from", where the "to" column specifies the outbound iw's and "from" column the inbound relative to the selected design solution. Here, the user can simply click on the desired design solution to or from which an iw is to be created. The selected design solution will be marked by applying a grey background colour to it after the iw connection has been stored in the tool's database. An advantage of having both a "to" and "from" column is that the user can specify all iw's that this design solution has, inbound or outbound, without having to navigate to the other design solutions. This is particularly advantageous for large EF-M trees. For instance, two-way iw's can be created by simply clicking once in each column for a certain design solution. Two disadvantages is of course that the user has to click two times to create a bidirectional iw and that the list will be very long when modelling a large EF-M tree.

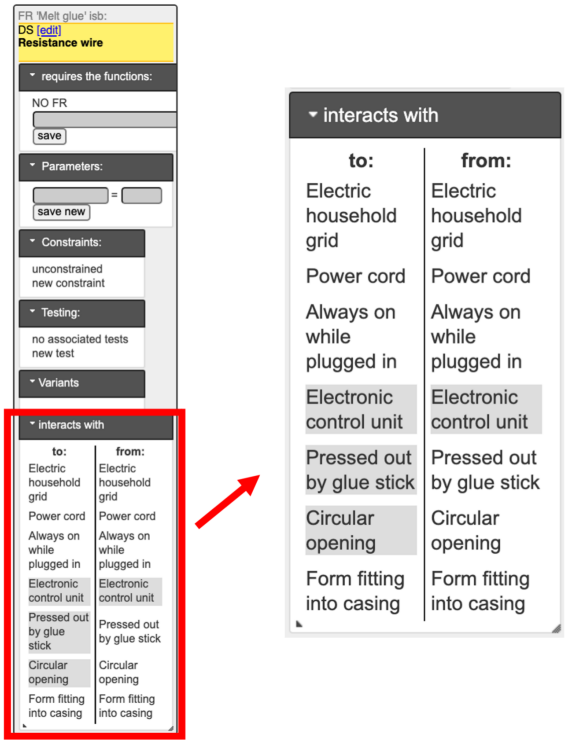


Figure 4.2: When clicking on a functional requirement or design solution in the tool, the detail box fills with information about the clicked object. Here the detailed view of design solution "resistance wire" from Figure 4.1 is shown, with the section for iw connections enhanced. Note here, that "resistance wire" has three outbound and one incoming iw, which the grey highlighting indicates, and that there exists a bidirectional iw between the design solutions "resistance wire" and "electronic control unit", since this is marked in both the "to" and "from" columns.

The DSMs are generated per request by the user based on the iw's stored in the tool's database and an example is shown in Table 4.1. The DSMs are available from the

concept view of the tool, from where DSMs can either be downloaded in a selected file format or shown directly in the browser. The DSMs are available for each concept and can then be downloaded manually one-by-one (referred to as single export) or all at once (called bulk export). The bulk export has two options: either all DSMs will be added in a single file; or one file per concept will be created. The former will be received in the selected file format, whereas the latter will be received as a compressed .zip archive, but can be extracted (or unzipped) in order to access the files in the selected file format. The user can select between the two possible transpositions of the DSM (see section 2.2.1) prior to download, as well as toggle between them as when shown in the browser.

Table 4.1: The corresponding DSM of the EF-M tree from Figure 4.1. This exact example has been generated in the tool and downloaded as with the option "LaTeX table in .txt", which is one of the available file formats (see Table 4.2).

		1	2	3	4	5	6	7	8	9
Power cord	1		x							
Always on while plugged in	2	x								
Electronic control unit	3	x							x	
Electric household grid	4									
Pressed out by glue stick	5								x	
Circular opening	6								x	
Form fitting into casing	7									
Resistance wire	8			x						
Melting unit	9									

The different options the user has and the file formats available are summarised in Table 4.2. The least obvious of them will be elaborated on here. The ".csv for CAM" is a comma separated values (csv) file that is specially formatted to be able to open the DSM in the CAM software (see section 3.5.1). Bulk exporting this type of .csv-file with all concepts in one file will put all DSMs along the diagonal in the same DSM. By doing this for the .xlsx-files, an excel file with the DSM of each concept on a separate sheet will be received.

Table 4.2: The possible options the user has and the different file formats that the user can receive DSM in.

Single export		**Bulk export**	
Alternatives	*File formats*	*Alternatives*	*File formats*
Show in browser	.xlsx	All concepts in one file	.xlsx
Download file	.csv	One file per concept	.csv for CAM
	.csv for CAM		
	.pdf		
	.png		
	.txt		
	LaTeX table in .txt		

4.2 Second Level Functionality

The visualisation of iw characterisation is ensured by drawing the iw arrows in different colours depending on the domain of the iw. To further allow for good visibility of trees containing multiple arrows, the different domains default with different bend factors, which can be seen as a factor that specifies how far out the arrow goes depending on how long it is. To adapt to a problem with the drawing of the iw arrows that is discussed in section 6.3.1.1 and 6.3.1.2, the drawing of the iw arrows has been changed. The earlier version, which can be seen in the previous section of this chapter, made iw's both originate and terminate in the middle of the bottom or in the middle of the right side depending on the view. Now, one-way iw's originate from a point between the left edge and midpoint and terminate in a point between the midpoint and the right edge, whereas bidirectional iw's originate and terminate in the midpoint. This way, the observer can easily identify the direction of an iw without having to look for an arrowhead. This is shown in Figure 4.3, which represents an example EF-M tree that will be used as an example throughout this section.

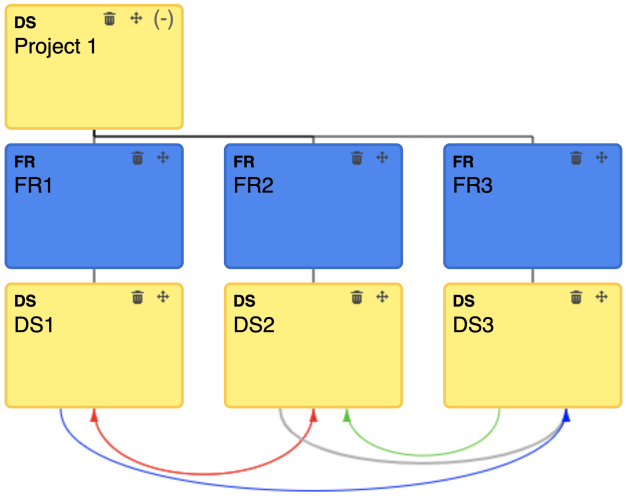


Figure 4.3: An example EF-M tree, which illustrates iw categorisation of the second functionality level. The new way of drawing arrows is also here exemplified. The informed observer can by just looking at design solution "DS2" see that the red arrow is a bidirectional iw, the green is arrow is an incoming iw and the grey arrow is an outgoing iw. The observer gets this information by just identifying the placement of the iw arrows on the design solution.

The domains default as *spatial*, *energy*, *information* and *materials* after Pimmler and Eppinger (1994) and with corresponding abbreviations. The abbreviations, names, colours, line widths and bend factors of the categories can be edited by the user in a widget designed for precisely this purpose. This widget, which is called the *iw legend*, is shown in Figure 4.4 with the default values of each domain's field.

To support the managing of the iw arrows and to give the user a clear overview of which iw's currently exist, a special widget was developed called the *iw manager*. The iw manager houses options for filtering which iw's are visible, possibilities of

Figure 4.4: The widget used for editing the iw categories is shown here. The categories' abbreviation, name, colour, line width and bend factor can here be tailored after the user's needs and wants. The changes are stored on project level and, when saved, instantly updated to the new values at their different points of usage throughout the tool. Reset buttons have been implemented for reverting to the default values for one or more of the categories. The default values are the values that in this figure populate the fields.

creating iw's as well as highlighting, editing or deleting already created iw's. In the default view, the user is presented four lists, one for each domain, where the iw's for that domain are shown with details like the affected design solutions, the direction of the iw and the its assigned value. This is shown in Figure 4.5.

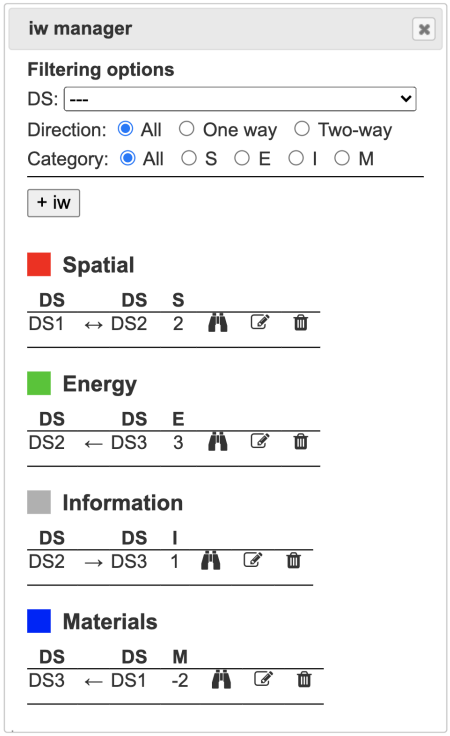


Figure 4.5: The iw manager, which is used to manage the iw's. Each iw can be highlighted (🔭), edited (✎) and deleted (🗑).

4.3 Third Level Functionality

As described in section 3.3, the FR library consisted of two views (or "sub-libraries"), namely the FR-DS library and the FB library. The FR library is accessed from the form where functional requirements are created, which is located in the detail box when a design solution is selected. The product of the user's interactions in the FR library is either a functional requirement with a design solution as a child, or just a functional requirement. Since functional requirements and design solutions alternate on hierarchical levels in an EF-M tree, it was natural to place the FR library on the design solutions since the product will always be at least a new functional requirement. The user can easily switch between the two views by clicking on the toggle in the top left corner which is depicted in Figure 4.6. The toggle indicates the result of the library, e.g., the result of the FB library is a functional requirement, which is why that side of the toggle is marked by "FR".

(a) The FR-DS library is selected. **(b)** The FB library is selected.

Figure 4.6: The widget used to toggle between the two libraries. The text on the toggle hints at the result of that particular library. Note that the FR library consists of two sub-libraries called the FR-DS library and the FB library.

The FR-DS library contains combinations of functional requirements and design solutions pairs. The FR-DS combos are presented in a table and separated based on their affiliated category. Due to the available data, only two different categories exist currently – hydraulic and electronic – but this can relatively easily be extended in the future as long as the data is provided. Each table row houses a functional requirement and a design solution together with an icon depicting the design solution as can be seen in Figure 4.7. When the user selects a row, the there specified functional requirement is created as a child of the design solution from where the FR library was opened. The specified design solution is created as a child of this functional requirement.

The FB library supports the user in creating functional requirements formulated on verb-object form. The library is shown in Figure 4.8 and is based on a standardised function language, which is called a *functional basis* and is developed by Hirtz et al. (2002). The verb and object to be select are shown at the top and default as "[verb]" and "[object]" respectively. Below that the user interface for selecting the verb and objects is located, where the left side is for the verb and the right side is for the object. A colour-coding is used here a yellow for verbs and blue for objects, which has nothing to do with the functional requirements and design solutions as these colours normally depict in the tool. The user can either enter the words by writing in the text box above or by clicking in the tables below. When clicking the "create FR" button, a functional requirement is created with as it is specified. Anything can

Figure 4.7: The FR-DS library shown from the tool. The library consists of three columns in one table per supported domain (currently just hydraulic and electronic). The first column houses the functional requirement, the second the design solution and the third an icon corresponding to the design solution. When the user selects a row by clicking, the functional requirement is created with the design solution as a child element.

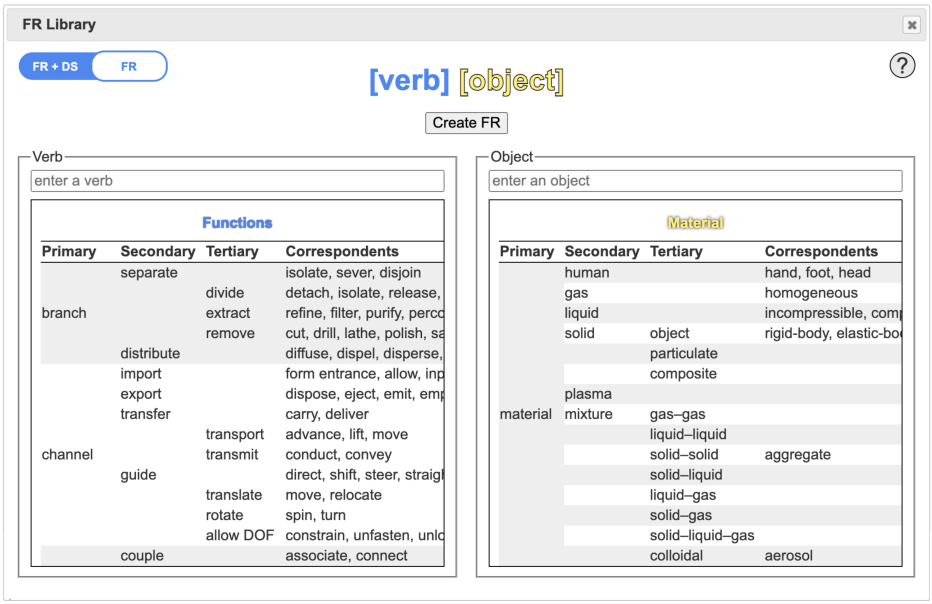


Figure 4.8: The FB library shown from the tool. The verb is created from the left-hand side of the tool whereas the object from the right. The user can either select the verb and object from their respective tables, or by typing into the text boxes above them. When typing, suggestions corresponding to the tables elements are continuously given.

be entered through typing, but suggestions do appear in a drop-down menu based on what has thus far been written. The suggestions are based on the words appearing in the tables. The tables work so that there are three levels of specification that can be used, which from lowest to highest are: *primary*, *secondary* or *tertiary*. When using the tables, the words specified in these columns are the only ones possible to use. An additional column called *correspondents* does also exist and is full of synonyms for the words in the other three columns. When selecting a correspondent cell, the word of either primary, secondary and tertiary is instead entered as a verb or object. The same goes for entering by text: when entering a registered synonym, the correct word will instead be entered into the verb or object slot when finished writing. Hirtz et al. (2002) specified one table for the verb and three for the object. The three object tables are for object in three different domains, namely: material, signal and energy. All four tables have been included in this implementation of the FB library.

The second sub-functionality of the third level functionality was the loading of EF-M models from DSMs. This was solved by allowing an EF-M tree to be represented by three DSMs. An excel file is uploaded which contains the three DSMs on separate worksheets. The first DSM is called FR-FR and specifies the parent-child relationships of the functional requirements. It basically sets the structure of the tree. The second DSM is called FR-DS and maps each design solution to a functional requirement. The third and last DSM is called DS-DS and it stores the iw connections, just like the DSMs previously discussed in the thesis. An example of this is presented in Appendix A, where the three DSMs and the EF-M they are representing are shown. The user interface of this functionality is located in the project view, where the projects are created. Certain checks are performed after uploading the DSMs to make sure they are properly set up. Thereafter the EF-M tree is created in a new project.

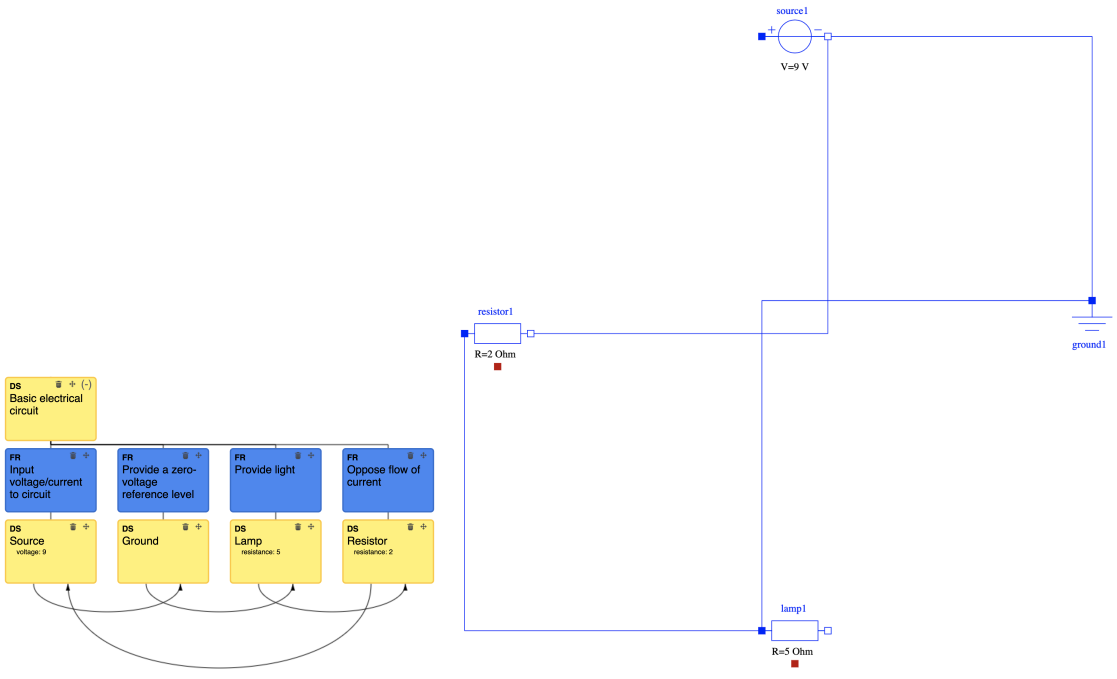
4.4 Fourth Level Functionality

The Modelica connection was realised by mapping elements from the FR-DS library to components represented in the Modelica libraries. The connection happens through an excel file which works as a library of supported Modelica components. The connection relies on an installation of OpenModelica, which provides both a Modelica executable together with many libraries as well as OpenModelica itself, which is a user interface for Modelica modelling and simulating. The excel file was set up manually, which contain certain information need for the connection. This information could for instance be the Modelica component name and path, an id mapping Modelica component to tool element, a list of coordinates for the Modelica component connection nodes, and a list of variables possible to set for the components.

A framework called OMPython was used to execute Modelica commands from the server side of the tool (backend). OMPython is a Python interface to OpenModelica and is used to execute Modelica commands directly from Python, which are otherwise entered in the command terminal. This was done since the tool's backend is written in Python. OMPython can be seen as the Python framework used to send expressions to OpenModelica to create the Modelica models and run simulations on them.

The Modelica connection follows an algorithm which is shown as a flowchart in Appendix B. First, after creating and loading the needed libraries, variables, models and Modelica, the names of the design solutions are compared with design solutions supported for Modelica conversion. If a match to a Modelica component, a corresponding Modelica component is created. The connections between the Modelica components are specified by the iw connections. If an iw exists between two supported Modelica components, these are then connected from one of the first component's nodes to one of the second component's nodes. Variables can be specified by adding parameters to the design solutions. The parameters were a functionality that already existed in the tool. A name needs to be provided for the parameters which exactly matches a supported variable of the component. During the entire process, multiple checks are made to make sure that the commands are executed properly in the Modelica framework. If any problems are found, the process is terminated and an error message is returned. Only the first but most major check is shown in the previously mentioned flowchart, namely the check that OpenModelica is in fact installed on the system.

There are two options to use Modelica from the tool. The first option is to download .mo-file containing the Modelica model of a concept. These files are then possible to open in OpenModelica as is shown in Figure 4.9. It is very intuitive and easy for humans to place elements in a reasonable and comprehensible way, but it is



(a) **(b)**

Figure 4.9: The EF-M tree a very basic electrical circuit in (a) has been exported as a Modelica model, which is shown loaded in OpenModelica in (b). Note that the lines connecting the components are not placed optimally as explained, which means that even though the components are properly connected to form a working electrical circuit, the lines visualising these connections might have a confusing appearance.

very complex to put this logic into code. Some form of artificial intelligence would most likely be needed to do this properly for all kinds of Modelica models. Due to these complexities, the Modelica components are simply placed in a circular pattern in the order they appear in the tool. The lines which indicate the connections between components are not drawn automatically when adding a connection between two Modelica components. Doing this properly would be equally complex as the placement of the components, so the lines are always draw according to the same simple rules. This leads to lines sometimes overlapping each other or the components. A solution to this is that the user can always drag and drop both the components and the lines to better locations when opening the generated files in OpenModelica.

The second option was to run a simulation on a concept directly from the tool. The user is here presented with the simulation setup window shown in . Here, the user can either choose to plot the data, which will open a OpenModelica plot window containing all the variables which can be filtered as desired, or to get the raw data. This second option will provide the user with an excel file containing the raw data for each of the variables.

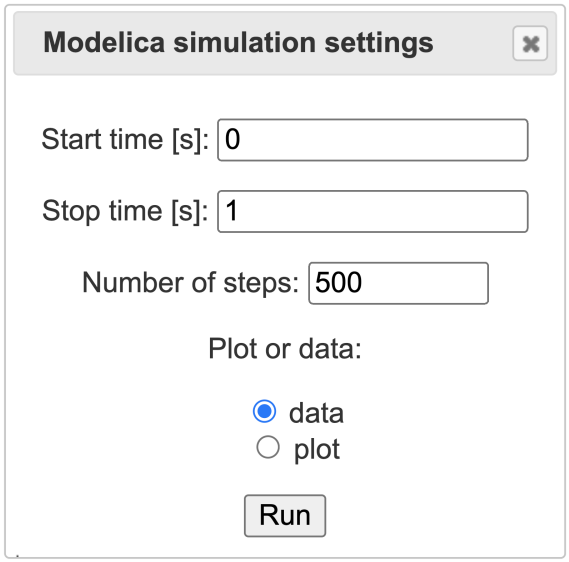


Figure 4.10: The Modelica simulation setup window shown in the tool. The start and stop time of the simulation can be set along with the number of steps, which is a number specifying the number of subintervals of the interval between start and stop time. The possibilities exist to either directly plot the data or to receive it raw in an excel file.

5
Case Studies

A number of case studies are performed to verify the developed functionalities, which are presented in chapter 4. These are done in order to see that the developed functionalities work as expected and to identify problem areas where further enhancement and improvements are needed. The case studies for the four functionalities are presented in sections 5.1, 5.2, 5.3 and 5.4.

5.1 First Level Functionality

The first level functionality was tried out on a jet engine component called static turbine rear structure (TRS). It is located at the rear end of the engine and has several functions. One is to connect the engine shaft. Another is to act as the rear engine mount and connect to the aircraft through the pylon. And a third is to deswirl the airflow that is leaving the low-pressure turbine and is entering the exit nozzle in order to increase the aerodynamic efficiency, which it does through internal guide vanes that also connect the outer with the inner ring.

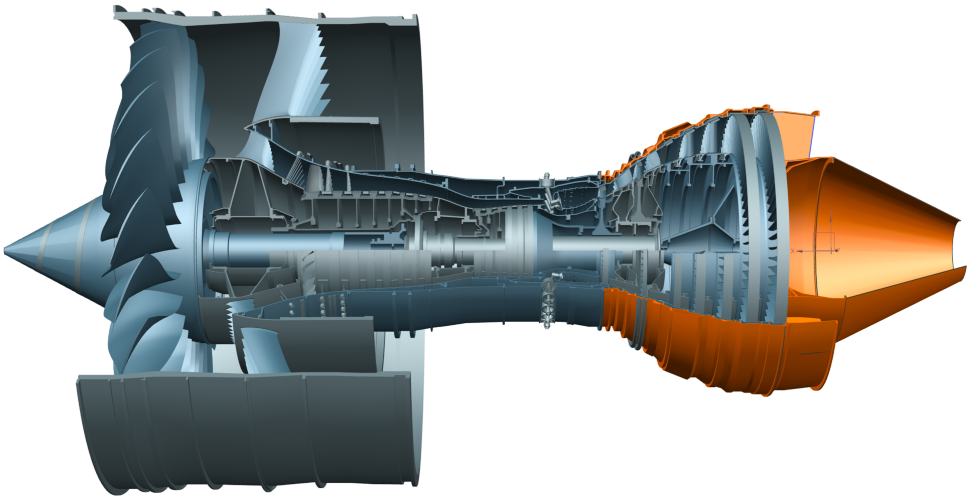


Figure 5.1: A commercial jet engine with the TRS highlighted in orange together with the low pressure turbine case and the cone. (Müller, 2020, p. 4)

5.1.1 iw Visualisation

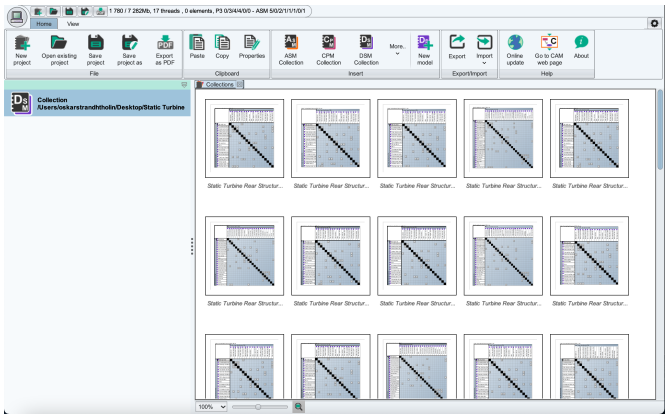
The EF-M tree of the TRS was modelled in the tool and is shown in Figure 5.2. During the EF-M modelling, it was noted that the designed limitation of which

design solutions should be possible to select when creating iw's, which is described in section 2.1.2.1, was offered a sufficient set of design solutions. Since the limitation of offered design solutions is done to restrict the user to only model reasonable iw connections and to make it as easy as possible to do so, the verification that it works on a real practical case is valuable.

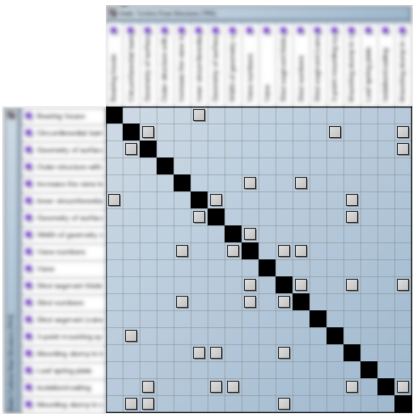
Figure 5.2: The EF-M tree of the TRS. Readability has been reduced in order to protect intellectual property.

5.1.2 DSM extraction

The EF-M tree of the TRS generated 48 different concepts. The DSMs of these could be extracted individually, one-by-one, or in bulk. The generated DSMs could be opened in CAM, either from one file per DSM, as shown in Figure 5.3, or together in a single DSM, as shown in Figure 5.4. The analyses that CAM offer, for instance CPM, could then be run on the workbooks containing the DSMs of the concepts.

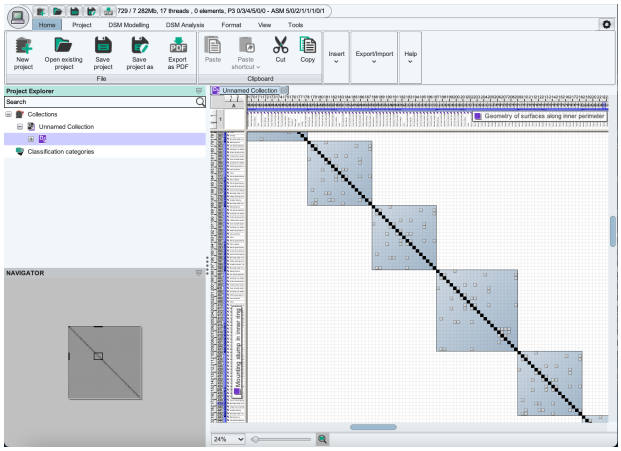


(a) The user interface of CAM, with some of the workbooks containing the 48 exported DSMs, which have all been imported in bulk.

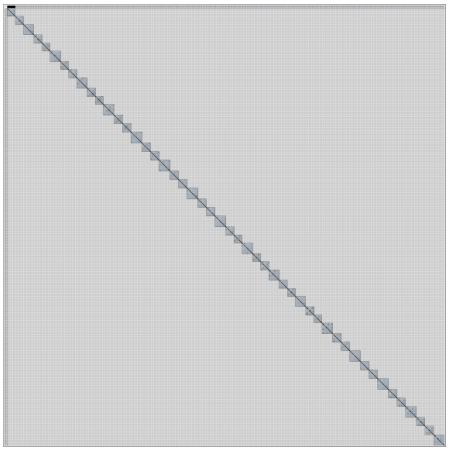


(b) A DSM opened in CAM, with the elements indicating iw connections visible.

Figure 5.3: The 48 concepts of the TRS have had their DSMs extracted in multiple files (meaning one file per concept) and are shown here opened in CAM. Readability has been hindered in order to protect intellectual property.



(a) The user interface of CAM is shown, with the DSM opened and four of the concepts' DSMs are visible along he diagonal, with the elements indicating iw connections visible.



(b) The whole DSM is shown opened in CAM, with each of the DSMs of the 48 concepts visible as small squares along the diagonal.

Figure 5.4: The 48 concepts of the TRS shown here opened in CAM. The DSMs of each concept have been generated into a single DSM, which means that they have been placed along the diagonal of the matrix. Readability has been hindered in order to protect intellectual property.

As this functionality was developed, the people behind CAM developed a dedicated functionality that allowed multiple .csv files to be imported at the same time. However, in the moment of writing, it is not possible to initiate multiple analyses at the same time. This means that if CPM analysis is wished to be run on multiple concepts, the CAM user needs to enter each worksheet (each DSM) individually, convert it to a CPM model and then initiate the analysis. The bulk import surely cut down the workload, since the engineer working on the TRS do not have to import 48 different csv files, but the engineer still needs to initiate 48 analyses per analysis.

One way around this problem would be to import just one DSM, which contains the DSMs of all the concepts, and then running a single analysis, which would in such a case be done on all concepts. This is the reason this possibility was implemented into the EF-M tool. However, CAM does not seem to be built for this purpose, since the time for 48 individual analyses is way less than that of running one analysis on a single DSM containing the DSMs of 48 concepts. This is likely due to the size of such a DSM, which results in an almost 1000-by-1000 matrix.

5.2 Second Level Functionality

The iw characterisation was verified using a bang-bang pressure regulating system, which in this case is used for satellite propulsion and is shown in Figure 5.5. A bang-bang system controls a process that has a binary output (fully on or fully off). Xenon gas is stored in a plenum tank and is released to provide thrust for the satellite. The bang-bang system includes components and interfaces from four domain

– spatial (S), energy (E), information (I) and materials (M) – which in Figure 5.5 are called mechanical, electrical, software and hydraulic respectively. The thesis will continue using the S, E, I and M naming for these domains as this has been used up to this point. The naming used in the figure more represent how these domains are represented in practice, but they are otherwise seen as synonyms.

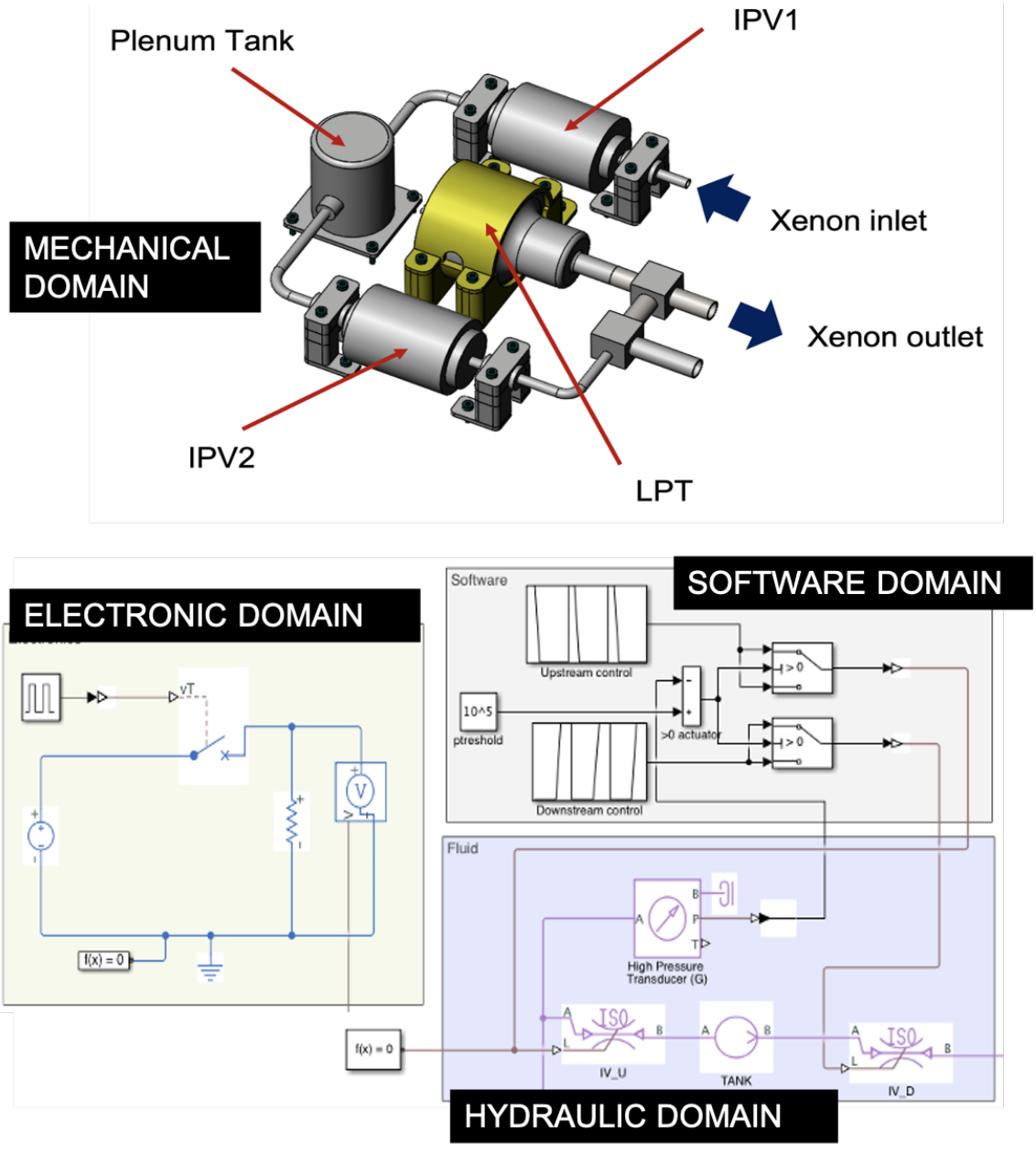


Figure 5.5: The bang-bang system used in the case study pictured in its mechanical, electronic, software and hydraulic domain. These can be seen as the same domains as spatial, energy, information and materials respectively, which have been discussed earlier in the thesis. The software Simulink has been used for all but the mechanical domain. Image is used with courtesy of Associate Professor Massimo Panarotto from the department of Industrial and Materials Science at Chalmers University of Technology.

The EF-M tree could successfully be set up using the developed methods for iw characterisation and is shown in Figure 5.6. The iw connections of each of the domains was possible to set up and a value could be given to each connection. The design solutions were given individual values for their inherent complexity. The DSM could be generated viewed and exported. A sample, which depicts the DSM as it is shown in the tool, is shown in Figure 5.7.

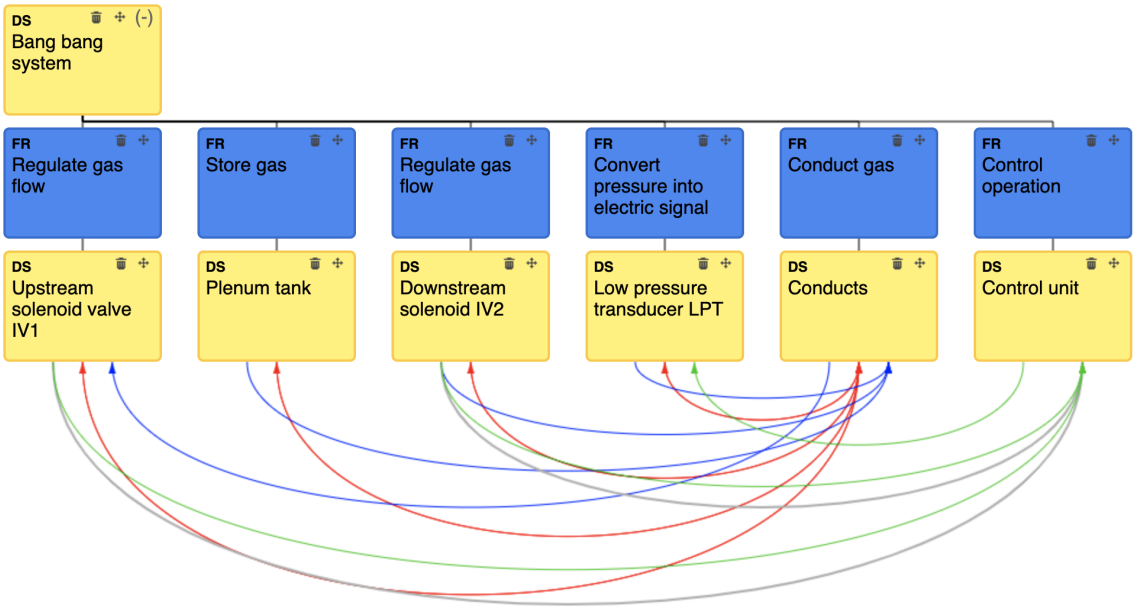


Figure 5.6: EF-M tree of the bang-bang system with the iw connections categorised in the four domains spatial (red), energy (green), information (grey) and materials (blue).

		1	2	3	4	5	6
Upstream solenoid valve IV1	**1**	50 7.5 0 6				4	2 2
Plenum tank	**2**		11 0 0 1			4 3	
Downstream solenoid IV2	**3**			50 7.5 0 6		2 3	2 2
Low pressure transducer LPT	**4**				19 607 0 0	2 1	
Conducts	**5**	4 3	4	2	2	18 0 0 2	
Control unit	**6**				1		0 97.5 177.2 0

S E
I M

Figure 5.7: The DSM of the EF-M tree of the bang-bang system from Figure 5.6 as it is shown in tool. Note the legend on the right-hand side to distinguish between the four domains. Also note that the diagonal cells contain values for the design solutions' inherent complexity.

The structural complexity was calculated as described in section 2.2.1.1 and shown directly in the tool's concept view. This shows that concepts can be evaluated based on complexity using the tool. The structural complexity of the bang-bang system for the four domains is shown in Table 5.1.

Table 5.1: The structural complexity values for the bang-bang system. All values have been rounded to one decimal point.

Domain	C_1	C_2	C_3	$C = C_1 + C_2 \cdot C_3$
Spatial	148	24	24.6	740.0
Energy	719.5	5	119.9	1319.1
Information	117.2	4	29.5	295.3
Materials	15	10	2.9	44.1

5.3 Third Level Functionality

To verify the FR library's intended functionalities, a mechatronic product architecture with some hydraulic components was sought. A refrigerator was selected since it follows these requirements and as it is of reasonable complex. The FR-DS part of the library was here of interest to verify since it offered the most interesting functionalities. Also, the FB library as well as the loading of EF-M from DSMs has been tested to work during development.

The refrigerator was modelled as a physical, a hydraulic and an electrotechnical domain. The physical domain was here not as important as the others, since there is no such representation yet in the FR-DS library, so it is just modelled as a box with a door. The hydraulic domain is modelling the refrigeration cycle typically found in an arbitrary refrigerator. The electromechanical domain modelled a power source that was connected to a thermostat that controlled an electrical motor connected to the compressor of the refrigeration cycle. Also, a lamp was included in the electric circuit, which was connected to the door (the lamp turns on when the user opens the door).

The refrigerator was possible to model with just a few new FR-DS pairs having to be added to the library. Only the refrigeration cycle and the electronic circuit was modelled using the library, since the physical domain (box with door) was not supposed to test the library's function but was merely needed to accurately represent a refrigerator. Due to the size of the EF-M tree, the refrigerator is shown in Appendix C, with iw connections corresponding to that explained above.

5.4 Fourth Level Functionality

To verify the simulation functionality of the Modelica connection, a simple DC motor with an electrical circuit diagram as shown in Figure 5.8 was used. The DC motor was modelled in the tool using components from the FR-DS library exclusively, as it is from here the connection between the EF-M elements and the components of

Modelica's libraries is realised. To verify the functionalities for different concepts as well, two alternative power sources were used, so that at least two concepts could be analysed.

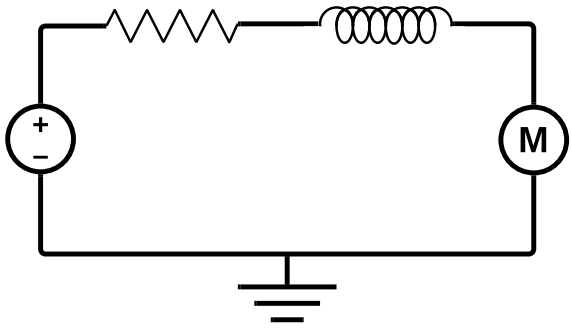


Figure 5.8: The electrical circuit diagram of a simple DC motor. Starting from the symbol marked with M and going counter-clockwise, the meaning of the symbols are electrical motor (or a device producing an electromotive force), ground, voltage/electrical source, resistor and inductor.

The EF-M tree of the DC motor is shown in Figure 5.9 and the two alternatives for the electrical power source are composed of a constant voltage source and a sine voltage source. The constant voltage source delivers its set voltage constantly, whereas the since voltage source has its voltage vary after a sine wave. As can be seen in Figure 5.9, both sources have their voltages set to 13 V, whereas the sine source has its frequency and phase set to 50 Hz and 0° respectively.

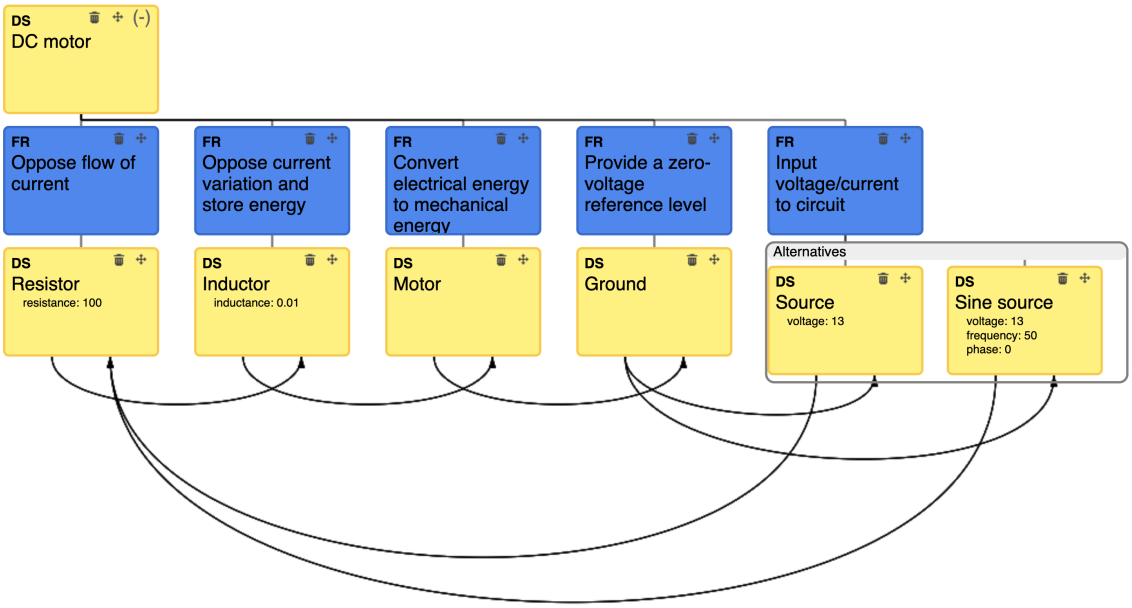
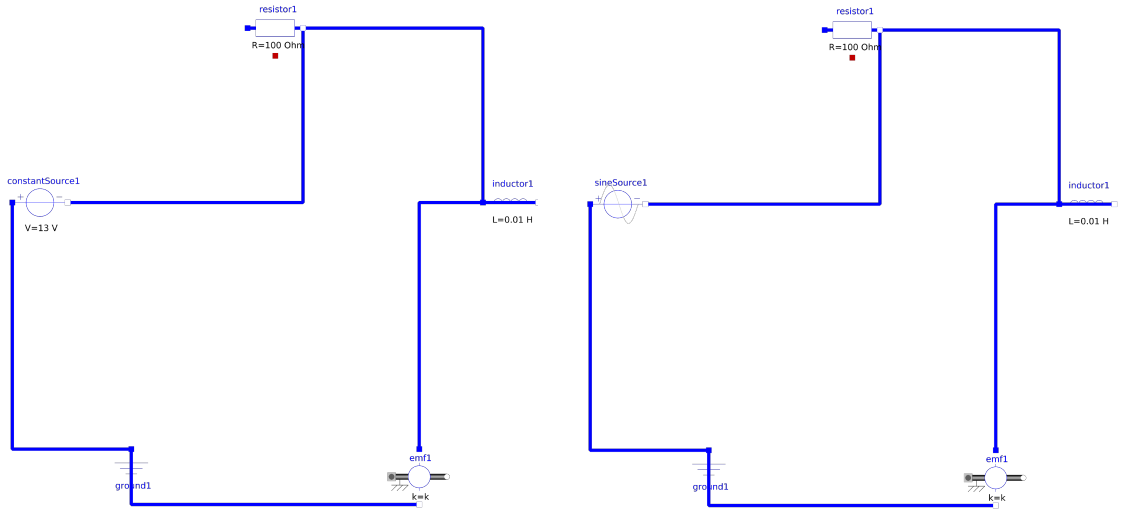


Figure 5.9: The EF-M tree of the DC motor from Figure 5.8 with the power source having two alternative design solutions: constant voltage source (just called source) and a sine voltage source. Note that the Modelica parameters for some of the design solutions have been set directly on the tool.

Both concepts could be exported as Modelica models from the tool as intended. The Modelica code generated from the tool is placed in .mo-files, which is a dedicated OpenModelica file format. These files are essentially text files containing the code that describes the Modelica models. The code describing the two concepts are shown in Appendix D and the two concepts opened in OpenModelica are shown in Figure 5.10.



(a) The DC motor with constant voltage source.

(b) The DC motor with sine voltage source.

Figure 5.10: The two concepts represented in the EF-M tree in Figure 5.9 has been exported and shown here after loaded into OpenModelica.

Modelica simulations could be run on both concepts directly from the tool. As explained in section 4.4, the user can choose between receiving the raw data in a file or plotting it directly from the tool in OpenModelica. When running the simulation, all the resulting data for all available variables are received, independently whether the data is returned or plotted. Both concepts were simulated between 0 and 0.5 seconds with 5 000 steps. The constant voltage source yielded an angular velocity of -13 rad/s, which is assumed to be counter-clockwise. The sine source yielded an angular velocity that changed between 13 rad/s and -13 rad/s according to the sine wave of the source as is shown in Figure 5.11. It was noted when varying the parameters in the EF-M model that changing the voltage of the different sources changed the angular velocity accordingly. That is, changing the voltage to for instance 4 V yielded an angular velocity of 4 rad/s or made it vary between ± 4. This is not very realistic and is due to the selected Modelica component, which is used to represent the conversion of electrical to mechanical or rotational energy (no load is applied for instance).

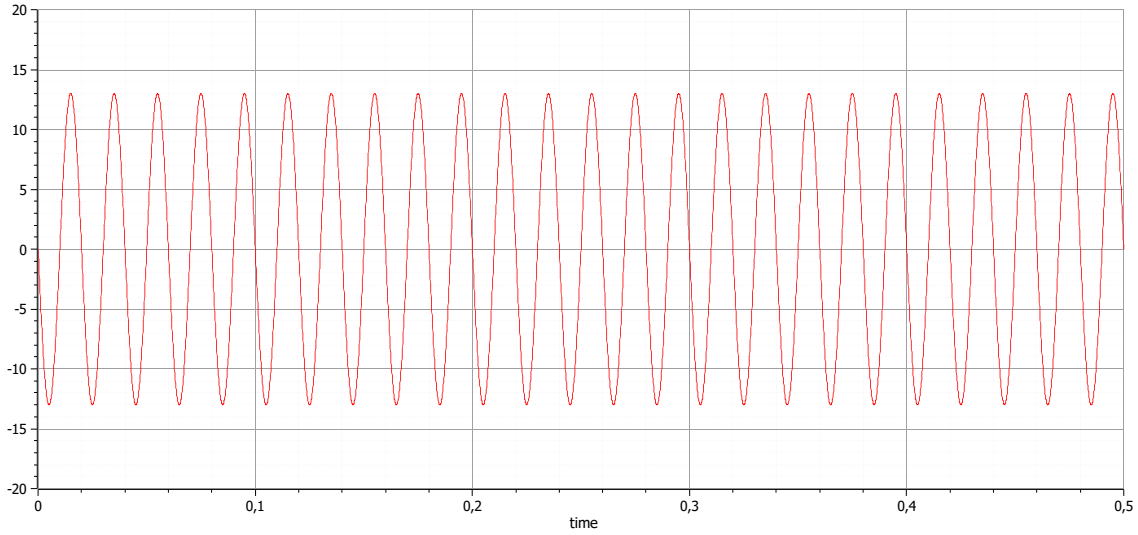


Figure 5.11: The resulting angular velocity from the simulations and of the concept which has a sine source is shown here plotted in OpenModelica. The vertical axis shows angular velocity in rad/s. Note how the angular velocity follows the sine wave of the source with a frequency of 50 Hz between 13 rad/s and -13 rad/s.

6
Discussion

This chapter will discuss the presented results and case studies in terms of what impact this will have on the current state of the art and state of the practice. This takes place in section 6.1 and 6.2 respectively. Thereafter, some more technical considerations are discussed in section 6.3.

6.1 Contribution to the State of the Art

A new EF-M modelling software tool has been developed in the thesis. The tool contributes to the current state of the art by enhancing the design space exploration. As large part as possible of design space should be explored when generating concepts (Ulrich & Eppinger, 2016, p. 119). At the same time, we know that companies typically are focusing their development efforts on only one concept or architecture (Isaksson et al., 2016), and that available tools and methods are expensive and offers a limited flexibility and exploration of the design space (Müller et al., 2020). The tool changes this status quo by making it possible to represent completely different products in the same models together with several variations of them. The tool does this based on functional decomposition, meaning that the problem-solving advantages that this brings as described by Ulrich and Eppinger (2016, pp. 121–123) can be exploited. Additionally, this also means that the tool is not relying on an existing geometry, which means that it can be employed early in the product development process.

Multiple concepts can be represented efficiently in a single EF-M tree (see for instance Figure 5.2 for one containing 48 concepts) in the tool. These concepts meaning multiple different architectures can be represented together in the same model, meaning that they can be analysed equally efficient and with the same methods. The tool can, independent of the type of architecture or interactions (see section 5.2), generate these concepts and export them as DSMs. This means that existing DSM based methods, such as structural complexity (Sinha et al., 2013) or change propagation method (Clarkson et al., 2004), can be utilised for evaluation of these concepts. This means that concepts can be evaluated before design embodiment based on non-functional requirement, which is very important in industries that are subject for change and innovation since non-functional requirements affect both the novelty and variety of generated concepts (Worinkeng et al., 2015). Furthermore, the effects of the problem described as the design paradox (Ullman, 2010) in section 2.3 are lessened, since better informed decisions can be made earlier. Concepts can be generated in greater extent compared to existing methods, such as CAD (Müller

et al., 2019), and evaluated more effectively with the tool. This means that the explored design space is both larger and more easily explored compared to before. The combination of these two effects means that more time can be invested in either selecting the right alternatives, in refining the concepts selected for further development, or in the upcoming stages and phases of the product development process.

The concepts represented in the tool can be represented as Modelica models (Modelica Association, 2021), on which simulations can be run as proven in section 5.4. This means that the concepts can be evaluated based on performance criteria while being represented in a function model. A connection between the EF-M tool and CAD model generation has previously been demonstrated by Müller et al. (2020), but a connection to a simulation software like Modelica is a novelty.

6.2 Contribution to the State of the Practice

Companies today are heavily invested in extensive CAD-models and assemblies (Isaksson et al., 2016). They set up comprehensive parameterisation generate a vast number of CAD-models, which can be analysed using methods like FEA and CFD. These are well-defined and robust methods and allows the designer to make very well-informed design decision and to understand the functional performances of the product. However, these methods rely on an existing geometry and typically only consider one type of product architecture (Isaksson et al., 2016). Companies need to have better methods for considering more radically different alternatives and for exploring a larger design space during the concept development. Early in the development process, no geometry or CAD-model exists, and companies cannot only consider one type of architecture in the changing industries today. The tool developed in this thesis enhances the concept generation to explore a larger design space and can represent multiple types of architectures simultaneously. It also does not rely on an existing geometry, meaning that companies can explore a large design space before committing to a geometry or product architecture. They can then be more confident with the product architecture that they choose to continue development than they can today.

Furthermore, trends seen in the industry and on the market, such as electrification, automation, new-technology introduction and increasing environmental constraints, are both enabling and requiring new product architectures. Companies also find it difficult to integrate new technologies into existing architectures (Panarotto et al., 2018), and new technologies need to be fully understood by companies before they can commit fully to them (Isaksson et al., 2016). Thus, companies need to be able to assess the risk of a product architecture to be able to determine the degree of difficulty of future integration of new technologies into the mentioned architecture. Here is where the DSM based methods described in section 2.2.1 can help. The tool has been connected to these methods, as described in section 4.1 and 4.2 and verified in section 5.1 and 5.2. This means that companies can both model and assess multiple different architecture based on these non-functional requirements, which are so important to be able to evaluate in innovative situations (Worinkeng et al., 2015).

6.3 Technical Considerations

After some of the functionality levels had been developed into the tool, and often during the case studies, were many problems with the implementation discovered. Many of these were minor bugs or incomplete features, which both could be solved relatively quickly to make the tool work properly. However, some of these problems required a more dedicated problem-solving approach and time investment to be tackled. These problems are discussed in this section. They are divided based on functionality level of either where their solutions are most relevant or where they were first encountered.

6.3.1 First Level Functionality

Two problems were found with the way the first level functionality was implemented, namely obscurity and ambiguity of the iw visualisation. These are elaborated on in the two next subsections. Both led to the development of improvements of the way iw's were visualised in the tool.

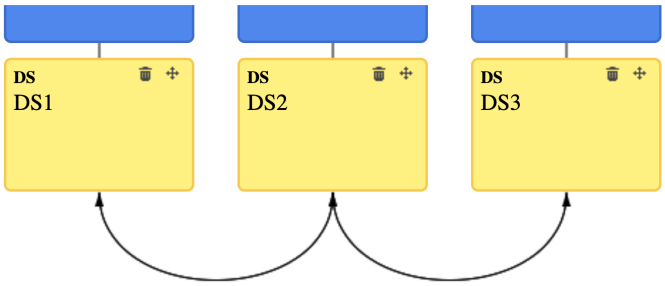
6.3.1.1 Obscurity of iw Visualisation

This case study highlights one limitation to the EF-M tree, which is its limited ability to display iw connections for large trees. By observing Figure 5.2, one can identify multiple clusters iw connections, where the lines are so entangled that they are difficult to follow. These clusters obstruct the EF-M tree's ability to give a clear and quick overview of the interactions between design solutions, as the EF-M tree typically otherwise does, for instance of the functional requirement and design solution relationships.

One way to solve this is to include the possibility of highlighting the iw's that are in- or outbound from a selected design solution, by either changing their colour or reducing the visibility of the other design solutions. Another solution is to allow the user to set the tool to only visualise the closely related iw connection, such as those in the same configurable component, which means that only short arrows will be drawn. The same functionality for longer arrows can of course be developed.

6.3.1.2 Ambiguity of iw Arrows

Another problem is that if a design solution is connected with an iw arrow to another design solution, which has another iw arrow connected to it, it is impossible to say if the firstly mentioned iw is two-way or not if there exist one arrowhead at both design solutions. This is because it is impossible to say if it is a single arrowhead, or multiple at the design solution with multiple iw's, since if it is multiple arrowheads they are drawn on top of each other which is visually identical to a single arrowhead. This ambiguity is exemplified in Figure 6.1, where it is impossible to say if it is DS1, DS3 or both that interacts with DS2. The DSM of the EF-M tree in question has to be consulted to determine the direction of the. This illustrates one of the advantages of DSMs over EF-M trees and emphasizes the benefits of DSM extraction from the tool.



(a) An example EF-M.

		1	2	3
DS1	1		X	
DS2	2	X		X
DS3	3		X	

(b) The DSM of (a).

Figure 6.1: An example of how the direction of iw arrows that converge at the same point are impossible to interpret. It is impossible to say if it is DS1, DS2 or both that constitute an inbound iw connection at DS2. One has to consult the DSM to be able to make this determination, which here shows that both iw's are two-way iw's.

This problem can of course be solved by separating the point of contact for the different iw's on the design solution. The tool can for instance be set up so that a design solution has a point on its left-hand side from which all outbound iw's originate from, and equally a point on its right-hand side for the inbound iw's. That way, overlapping arrows would not be a problem, since the point alone to which an iw is drawn indicates its direction. This has been done in another version of the tool and is pictured in Figure 6.2.

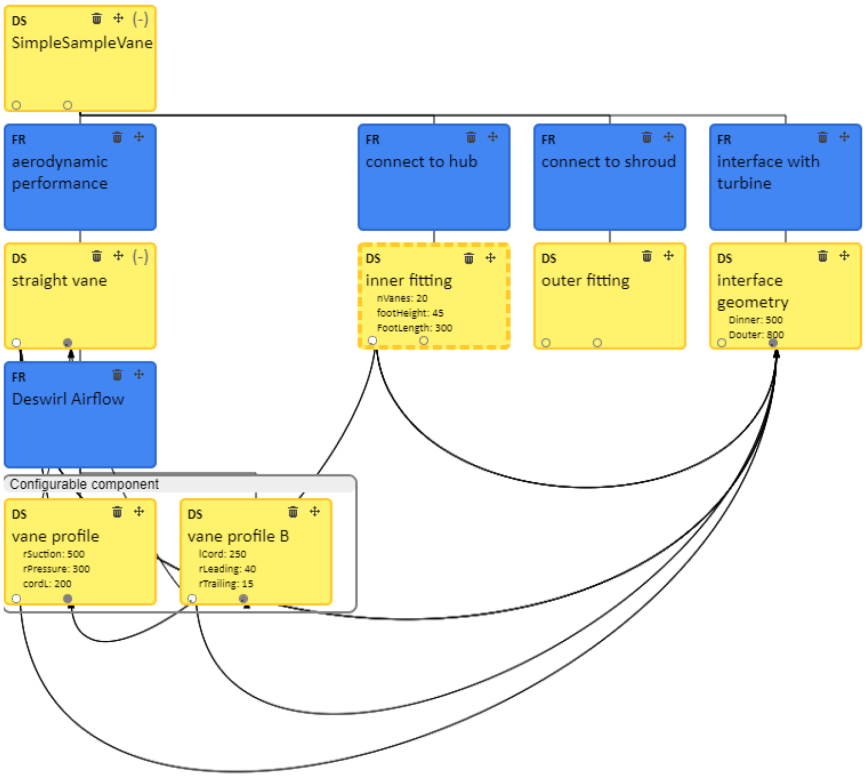
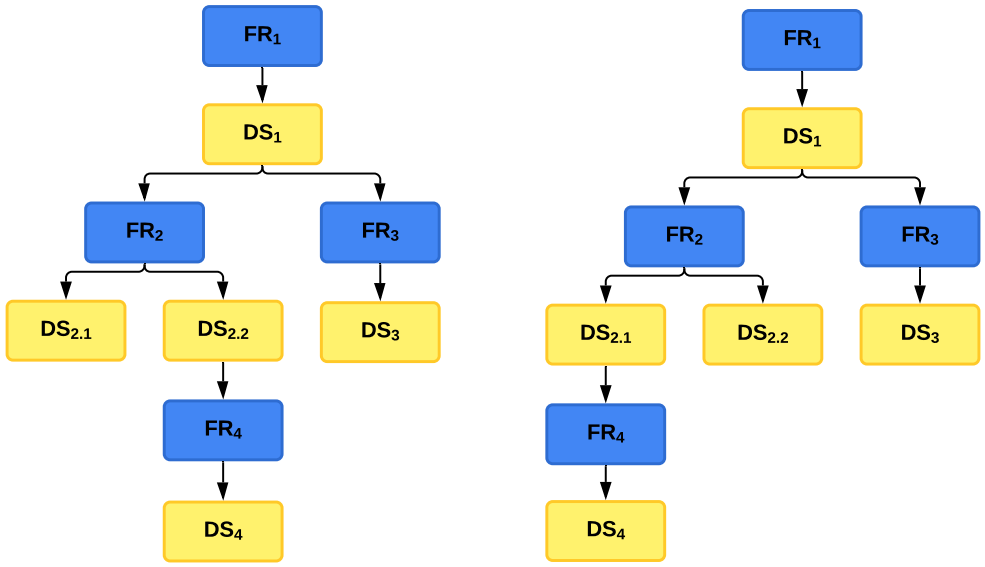


Figure 6.2: An example, where design solutions have two different points iw connections. The left point is for outbound iw's and the right for inbound. (Müller et al., 2020)

Adding such a functionality will, however, lead to more arrows being created since two-way arrows will with this implementation be two separate arrows, and not just one arrow with two arrowheads as in the current implementation. This will mainly be a problem for larger trees, but it is also for larger trees where the value of iw connections diminish and do not provide a clear overview anymore (see section 6.3.1.1 or compare Figure 5.2 and Figure 6.2), so for the sake of supporting smaller trees this was considered to be a reasonable solution to implement. This was also done for the second functionality (see section 4.2). The selected solution, which is visualised in Figure 4.3, has an origin point a small distance to the left of the mid-point of the bottom side, and a destination point a small distance to the right of this point. For complex iw modelling, which is part of the second level functionality), bidirectional arrows use the midpoint, which solves the problem of bidirectional arrows creating more arrows. They way that the iw arrows were stored in the database led to this was not being possible to implement for the bidirectional iw connections of this first functionality level.

6.3.2 Third Level Functionality

A sub-functionality for creating EF-M trees by uploading DSMs were included in the third level functionality. The feature works as intended where the user uploads an excel-file containing three DSMs which all together represent an EF-M tree. One problem with the selected approach, however, is that EF-M trees that represent multiple concepts cannot be represented in three DSMs by using this method. The reason for this is that two different EF-M trees can have the same FR-FR, FR-DS and DS-DS matrices. For example, the two EF-M trees depicted in Figure 6.3 have



(a) (b)

Figure 6.3: Two EF-M trees that have the exact same FR-FR, FR-DS and DS-DS matrices. The matrices are presented in Appendix E

the same three matrices, which are presented in Appendix E. The problem lies in the FR-FR matrix and only emerges when at least one functional requirement is a child of one of many alternative design solutions. That is, if the alternative design solutions are all place at the end nodes of the tree, and thus have no children of themselves, the selected approach for DSM-to-EF-M would work. However, the support for representing alternatives with the selected approach has been fully deprecated, since the multiple-alternative representation was deemed too limiting.

One way to allow alternative design solutions to be modelled, is to use a DS-FR matrix instead of the FR-FR matrix. The DS-FR matrix would then specify which functional requirements are children of which design solutions. The problem with the two trees in Figure 6.3 having the same matrices, is that the method using an FR-FR among others cannot decide under which design solution the functional requirement FR_4 is placed. This problem would be solved with the DS-FR matrix. This approach is however not ideal since it will make the user having to jump between two DSMs when creating the EF-M tree. The method using an FR-FR matrix is more intuitive.

Another way is to just use one additional matrix to the one specifying the iw connections (the DS-DS matrix). This other matrix would be an FR-DS matrix (could also be a DS-FR), but which has different handling for the cells depending on if they are above or below the diagonal of the matrix. The cells located above the diagonal of the matrix would be specifying which design solutions are children to which functional requirements, whereas the cells below the diagonal would specify which functional requirements are children to which design solutions. This requires, however, that the functional requirements and the design solutions are ordered in the same way, otherwise it cannot be guaranteed that the cell assigning a design solution to a functional requirement is located above the diagonal. This leads to high demands on the user when creating the EF-M tree, as the user needs to move both the design solution and the functional requirement if a change of structure of the EF-M tree happened or if one or multiple elements are added in the middle of the tree. This means that the entire column (design solution) and entire row (functional requirement) need to be moved. Of course, this can in turn be solved by having multiple cells per cell of the DSM, or by one marking for the FR-DS relationship and another for the DS-FR relationship. However, these options do also have their disadvantages, such as they would lead to worsened readability and overview of the DSM.

6.3.3 Fourth Level Functionality

The Modelica connection works, as was verified in section 5.4. The results currently yielded by simulations might however not always be neither realistic nor accurate. This is however due to the used Modelica components, which are all from Modelica's standard library. The connection was set up to the components of Modelica's libraries for two reasons. Firstly, this was done because the Modelica libraries contain thousands of predefined models, which now do not have to be manually created. Secondly, if a connection to the Modelica libraries could be set up, a connection to any other library, such as those that will be created through future contributions or those created by the tool's user, is trivial. The current approach thus relies on the

existing Modelica libraries. The libraries are very extensive, but Modelica users are also expected to change their models according to their specific needs. Many models are also very specific. For instance, there are many different battery types supported in the standard Modelica library. However, if a user wants a "simple" battery, and only cares about setting a voltage, current and capacity of it, there might not exist a battery that is defined generally enough. Of course, a battery type that is general enough can be defined easily in a Modelica model or another electrical power source can be used and renamed to "battery". This is exactly what is needed for the future work of this Modelica connection. In other words: relevant Modelica components need to be found or defined and then mapped to the FR-DS library in order to yield relevant, realistic and accurate results from simulations. This has, as stated, not been done in this thesis as the goal was to connect to a simulation software, in this case Modelica, and precisely this has been achieved.

The robustness of the connection between the tool and Modelica strongly depends on the size and contents of the library, not just the accuracy of the Modelica-to-design-solution mapping as discussed above. Depending on the final size of the library, a potential further development here would be to store multiple design solutions that fulfil a particular functional requirement, instead of the one-to-one mapping of today. Would this happen, users could select a certain functional requirement or design solution in the library and the functional requirement together with all alternative design solutions that solve this functional requirement would be generated. The future development of this, in turn, would be that users select only the functions that their product should fulfil, and an EF-M tree is thereby generated containing all possible variants of such product. The user could then run a batch simulation on all these concepts and get a numerical answer to which product variant is best in fulfilling the top-level functional requirement of the EF-M tree. This would be the same as the user asking the software "what is the best product for solving this problem?" and getting a numerical answer back for each of the possible product variants describing how suited they are in fulfilling the specified top-level function. A vast library would of course be required for this to work, which would be the result of many years work by many contributors, which indeed is the reason for the current size of Modelica's standard library.

7
Conclusion

An enhanced tool to support the concept development phase of the product development process has been presented in the thesis. The tool, which is based on EF-M modelling can be used to represent multidomain architectures in the same model. The models can be analysed based on complexity or exported to external software for further analyses. A new state of the art connection between functional modelling and simulation software has been made, by connecting the tool to Modelica. This means that concepts represented in a functional model can be assessed and evaluated based on performance criteria and functional requirements.

Below follows the answers to the two research questions from chapter 1. After this, some recommendations for future research are presented.

In what way does the enhancement of the tool contribute to the early stages of the product development process?

Product architectures of multiple domains can be represented in the same model together with the product interfaces, which too can simultaneously be of different domains. The tool now connects the EF-M tree representation of products with DSM, and hence also the methods and external software that are basing on DSMs. Concept can directly from the tool be converted to Modelica models, on which simulations can be run. This means that the concepts represented in the EF-M tool can be evaluated on performance criteria. All this is done before design embodiment and thus already in the concept development phase of the product development process.

What key non-functional modelling criteria can be modelled with the enhanced tool compared to existing CAD systems?

The non-functional requirements possible to capture are mainly those that follows from the DSM connection. This thesis has mainly focused on complexity and risk, which can be analysed with the structural complexity and change propagation method respectively.

This thesis utilised a number of case studies to verify its results. The results, however, have not been validated. Future research should perform this needed validation by for instance interviewing industry experts after they have used the tool. Such insight would be needed to confirm the relevance and robustness of the tool and

developed method, as well as to adapt the tool more accurately after its target users. Furthermore, more work needs to be made on the connection to Modelica, in order to make the connection more robust and to include a larger library. The strength and advantages of the Modelica connection are strongly connected to the amount of Modelica components that are mapped to the functional modelling elements though the FR-DS library. Future research should explore this avenue, as it will fully take advantage of the possibilities of this functionality and might lead to automated design space exploration.

References

Alhamaly, A. (2019). *Jet Engine Remaining Useful Life (RUL) Prediction*. Medium. https://medium.com/@hamalyas_/jet-engine-remaining-useful-life-rul-prediction-a8989d52f194

axiom, n. (2021). Oxford English Dictionary. https://www.oed.com/viewdictionaryentry/Entry/14045

Benavides, E. M. (2012). *Advanced Engineering Design: An integrated approach*. Woodhead Publishing. https://doi.org/10.1533/9780857095046

Browning, T. R. (2016). Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. *IEEE Transactions on Engineering Management*, *63*(1), 27–52. https://doi.org/10.1109/TEM.2015.2491283

Clarkson, P. J., Simons, C., & Eckert, C. (2004). Predicting Change Propagation in Complex Design. *Journal of Mechanical Design*, *126*(5), 788–797. https://doi.org/10.1115/1.1765117

Cooper, R. G. (2011). *Winning at New Products: Creating Value Through Innovation* (4th ed.). Basic Books.

Dove, R., & LaBarge, R. (2014). Fundamentals of Agile Systems Engineering - Part 2. *INCOSE International Symposium*, *24*(1), 876–892. https://doi.org/10.1002/j.2334-5837.2014.tb03187.x

Eckert, C., Isaksson, O., Hallstedt, S., Malmqvist, J., Öhrwall Rönnbäck, A., & Panarotto, M. (2019). Industry Trends to 2040. *Proceedings of the Design Society: International Conference on Engineering Design*, *1*(1), 2121–2128. https://doi.org/10.1017/dsi.2019.218

Eppinger, S. D., & Browning, T. R. (2012). *Design Structure Matrix Methods and Applications* (Vol. 4). MIT Press.

Gonzalez Castro, S., Panarotto, M., Borgue, O., & Isaksson, O. (2020). Analysing Increase of Functionality and Complexity in Integrated Product Architectures. *Proceedings of the NordDesign 2020 Conference*, *DS 101*. https://doi.org/10.35199/NORDDESIGN2020.16

Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., & Wood, K. L. (2002). *A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts* (NIST Technical Note No. 1447). National Institute of Standards and Technology. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=821676

Hubka, V., & Eder, W. E. (1988). *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. Springer-Verlag. https://doi.org/10.1007/978-3-642-52121-8

Isaksson, O., Bertoni, A., Levandowski, C. E., Müller, J. R., Wiklund, D., & Johansson, P. B. V. (2016). Virtual Contextual Validation of Technologies and Methods for Product Development. *DESIGN 2016: 14th International Design Conference*, *2*, 669–678.

Johannesson, H., & Claesson, A. (2005). Systematic Product Platform Design: A Combined Function-Means and Parametric Modeling Approach. *Journal of Engineering Design*, *16*(1), 25–43. https://doi.org/10.1080/09544820512331325247

Levandowski, C., Raudberget, D., & Johannesson, H. (2014). Set-Based Concurrent Engineering for Early Phases in Platform Development. *Advances in Transdisciplinary Engineering*, *1*, 564–576. https://doi.org/10.3233/978-1-61499-440-4-564

Modelica Association. (2021). *Modelica Language*. https://modelica.org/modelicalanguage.html

Modelica Libraries. (2000). LiU. https://www.ida.liu.se/labs/pelab/realsim/library/ModelicaExamples.html

Müller, J. R. (2020). *Does Form Follow Function? Connecting Function Modelling and Geometry Modelling for Design Space Exploration* [Doctoral dissertation, Chalmers University of Technology]. Chalmers Research. https://research.chalmers.se/en/publication/?id=520355

Müller, J. R., Isaksson, O., Landahl, J., Raja1, V., Panarotto, M., Levandowski, C., & Raudberget, D. (2019). Enhanced function-means modeling supporting design space exploration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *33*(4), 502–516. https://doi.org/10.1017/S0890060419000271

Müller, J. R., Panarotto, M., & Isaksson, O. (2020). Design Space Exploration of a Jet Engine Component Using a Combined Object Model for Function and Geometry. *Aerospace*, *7*(12), 173. https://doi.org/10.3390/aerospace7120173

Page, L. (2009). *NASA working on 'open rotor' green (but loud) jets: Want to save the planet? put up with noisier airports*. The Register. https://www.theregister.com/2009/06/12/nasa_open_rotor_trials/

Panarotto, M., Isaksson, O., & Asp, L. (2018). Assessing the Value of Radical Technology Alternatives at System Level. *DESIGN 2018: 15th International Design Conference*, *2*, 633–642. https://doi.org/10.21278/idc.2018.0398

Pimmler, T. U., & Eppinger, S. D. (1994). Integration Analysis of Product Decompositions. *ASME Design Theory and Methodology Conference*, 343–351.

process, n. (2021). Oxford English Dictionary. https://www.oed.com/viewdictionaryentry/Entry/151794

Schachinger, P., & Johannesson, H. L. (2000). Computer Modelling of Design Specifications. *Journal of Engineering Design*, *11*(4), 317–329. https://doi.org/10.1080/0954482001000935

Sinha, K., Omer, H., & de Weck, O. L. (2013). Structural Complexity: Quantification, Validation and Its Systemic Implications for Engineered Complex Systems. *Proceedings of the 19th International Conference on Engineering Design*, *4*, 189–198.

Suh, N. P. (1990). *The Principles of Design*. Oxford University Press.

Ullman, D. G. (2010). *The Mechanical Design Process* (4th ed.). McGraw-Hill.

Ulrich, K. T., & Eppinger, S. D. (2016). *Product Design and Development* (6th ed.). McGraw-Hill Education.

University of Cambridge. (2021). *Cambridge Advanced Modeller 2*. https://camtoolkit.eng.cam.ac.uk/

Wang, Y., Liu, A., Tao, F., & Nee, A. Y. C. (2020). Digital Twin Driven Conceptual Design. In F. Tao, A. Liu, T. Hu, & A. Y. C. Nee (Eds.), *Digital Twin Driven Smart Design* (pp. 33–66). Academic Press. https://doi.org/10.1016/B978-0-12-818918-4.00002-6

Worinkeng, E., Joshi, S., & Summers, J. D. (2015). An Experimental Study: Analyzing Requirement Type Influence on Novelty and Variety of Generated Solutions. *International Journal of Design Creativity and Innovation*, *3*(2), 61–77. https://doi.org/10.1080/21650349.2014.909294

A

DSM to EF-M

The three DSMs in (a), (b) and (c) are representing the EF-M tree in (d). The FR-FR specifies that FR2, FR3 and FR4 are children of FR1. More precisely these are children of the design solution which is child of FR1. The FR-DS specifies that DS1 is child of FR1, DS2 is child of FR2, and so on. The DS-DS specifies the iw connections following the IC convention.

	FR1	FR2	FR3	FR4
FR1		×	×	×
FR2				
FR3				
FR4				

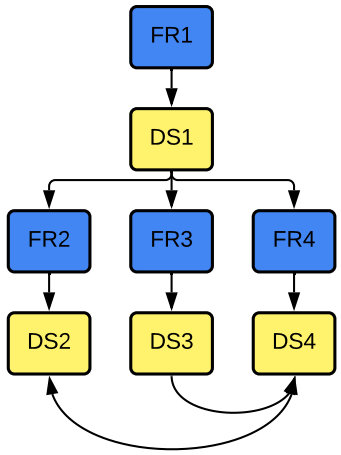
(a) FR-FR.

	DS1	DS2	DS3	DS4
FR1	×			
FR2		×		
FR3			×	
FR4				×

(b) FR-DS.

	DS1	DS2	DS3	DS4
DS1				
DS2				×
DS3				×
DS4		×		

(c) DS-DS.

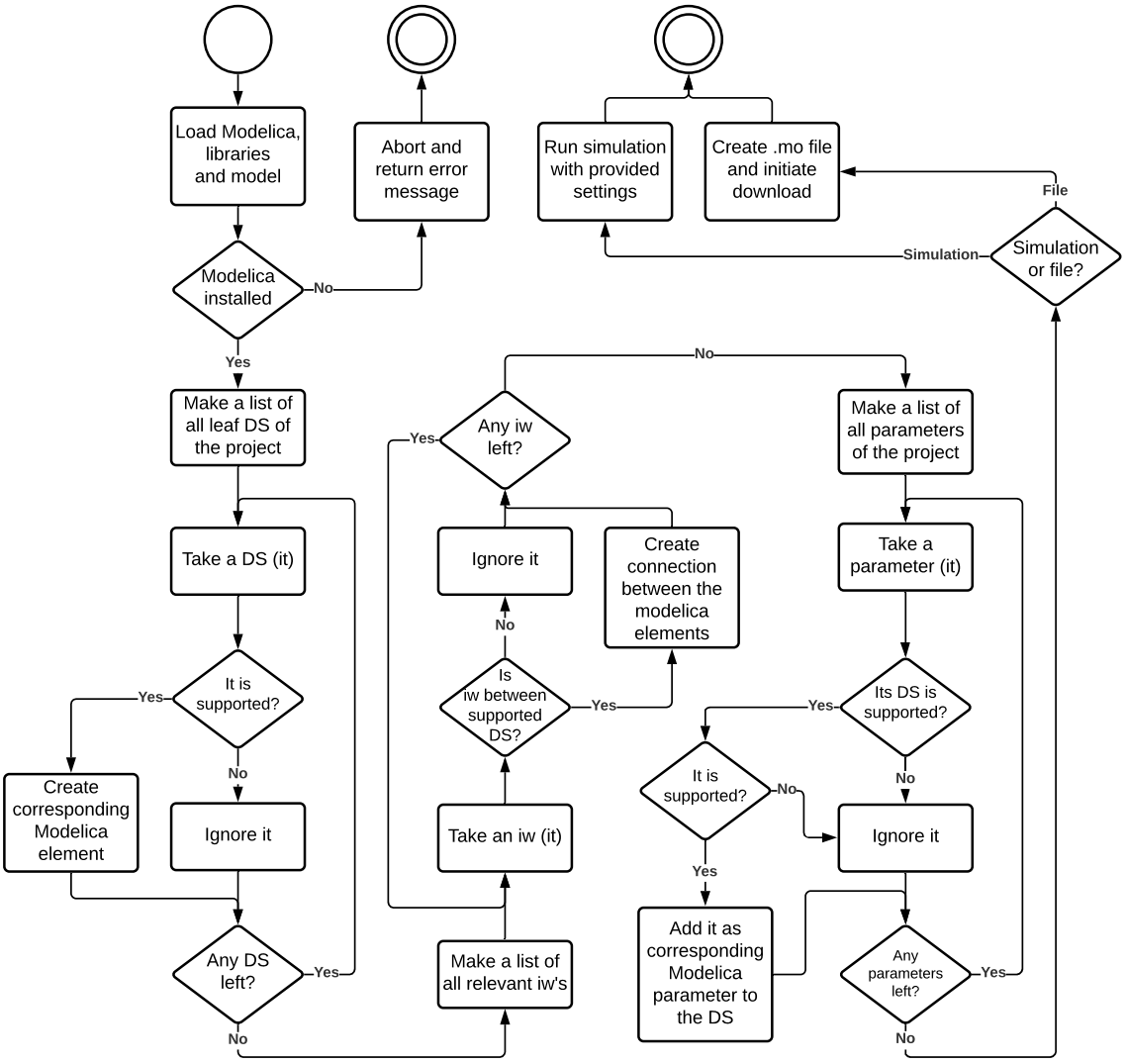


(d) The EF-M tree.

B

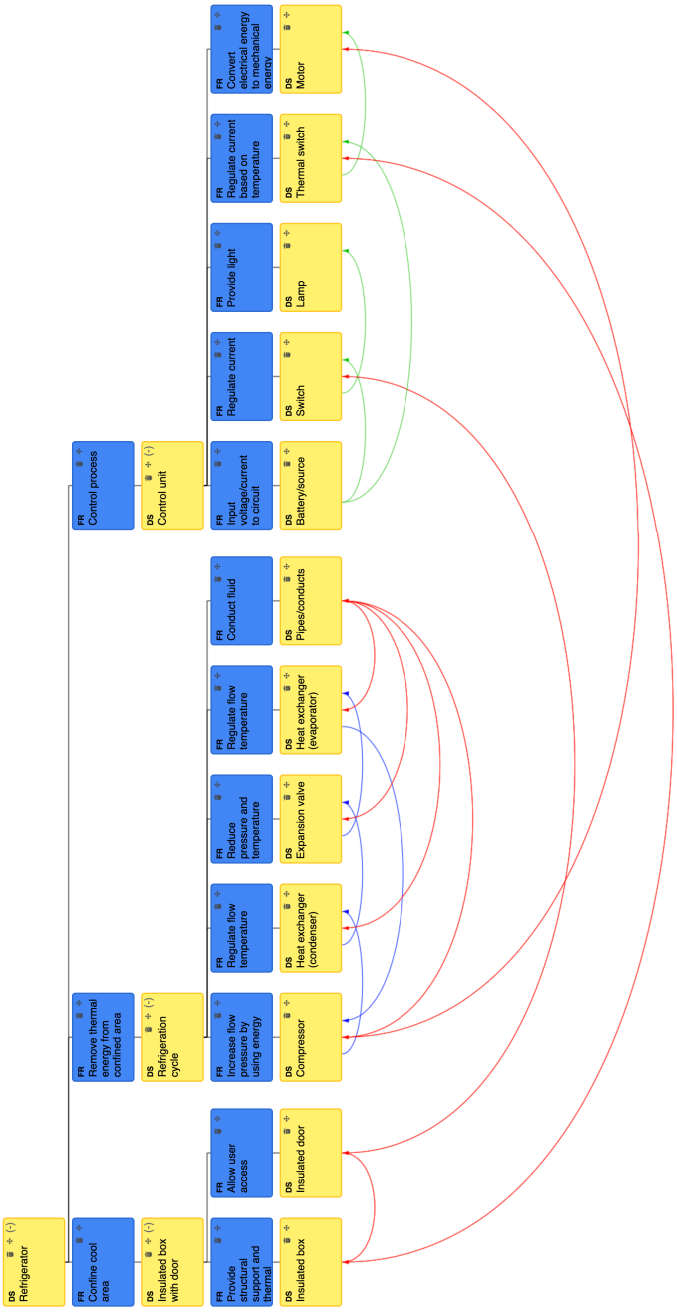
Modelica Algorithm

Flowchart or activity diagram of the method or algorithm used for connecting the tool to Modelica.



C

EF-M Tree of a Refrigerator



D

Modelica Code of Two Concepts

The concept using a constant voltage source:

```
model DC_motor_1
  Modelica.Electrical.Analog.Basic.Resistor resistor1(R = 100)
     annotation(
    Placement(visible = true, transformation(origin = {0, 90},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.Inductor inductor1(L = 0.01)
     annotation(
    Placement(visible = true, transformation(origin = {85, 27},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.RotationalEMF emf1 annotation(
    Placement(visible = true, transformation(origin = {52, -72},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.Ground ground1 annotation(
    Placement(visible = true, transformation(origin = {-52, -72},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Sources.ConstantVoltage
     constantSource1(V = 13) annotation(
    Placement(visible = true, transformation(origin = {-85, 27},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
equation
  connect(resistor1.p, inductor1.p) annotation(
    Line(points = {{-10, 90}, {75, 90}, {75, 27}}, color = {0, 0,
       255}));
  connect(inductor1.n, emf1.p) annotation(
    Line(points = {{95, 27}, {52, 27}, {52, -62}}, color = {0, 0,
       255}));
  connect(emf1.n, ground1.p) annotation(
    Line(points = {{52, -82}, {-52, -82}, {-52, -62}}, color = {0,
       0, 255}));
  connect(ground1.p, constantSource1.p) annotation(
    Line(points = {{-52, -62}, {-95, -62}, {-95, 27}}, color = {0,
       0, 255}));
  connect(constantSource1.n, resistor1.n) annotation(
    Line(points = {{-75, 27}, {10, 27}, {10, 90}}, color = {0, 0,
       255}));
end DC_motor_1;
```

The concept using a sine voltage source:

```
model DC_motor_2
  Modelica.Electrical.Analog.Basic.Resistor resistor1(R = 100)
      annotation(
    Placement(visible = true, transformation(origin = {0, 90},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.Inductor inductor1(L = 0.01)
      annotation(
    Placement(visible = true, transformation(origin = {85, 27},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.RotationalEMF emf1 annotation(
    Placement(visible = true, transformation(origin = {52, -72},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.Ground ground1 annotation(
    Placement(visible = true, transformation(origin = {-52, -72},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Sources.SineVoltage sineSource1(V =
     13, phase = 0, f = 50) annotation(
    Placement(visible = true, transformation(origin = {-85, 27},
       extent = {{-10, -10}, {10, 10}}, rotation = 0)));
equation
  connect(resistor1.p, inductor1.p) annotation(
    Line(points = {{-10, 90}, {75, 90}, {75, 27}}, color = {0, 0,
       255}));
  connect(inductor1.n, emf1.p) annotation(
    Line(points = {{95, 27}, {52, 27}, {52, -62}}, color = {0, 0,
       255}));
  connect(emf1.n, ground1.p) annotation(
    Line(points = {{52, -82}, {-52, -82}, {-52, -62}}, color = {0,
       0, 255}));
  connect(ground1.p, sineSource1.p) annotation(
    Line(points = {{-52, -62}, {-95, -62}, {-95, 27}}, color = {0,
       0, 255}));
  connect(sineSource1.n, resistor1.n) annotation(
    Line(points = {{-75, 27}, {10, 27}, {10, 90}}, color = {0, 0,
       255}));
end DC_motor_2;
```

E

Three DSMs

The FR-FR matrix:

	FR1	FR2	FR3	FR4
FR1		×	×	
FR2				×
FR3				
FR4				

The FR-DS matrix:

	DS1	DS2.1	DS2.2	DS3	DS4
FR1	×				
FR2		×	×		
FR3				×	
FR4					×

The DS-DS matrix:

	DS1	DS2.1	DS2.2	DS3	DS4
DS1					
DS2.1					
DS2.2					
DS3					
DS4					

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY