Development of a Multi-layer V-model Design Process and Computational Tools for Mechatronic Conceptual Design

by

Hani Balkhair

B.Sc., Umm Al-Qura University, 2002
M. Sc., University of Victoria, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Mechanical Engineering)

The University of British Columbia
(Vancouver)

November 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the dissertation entitled:

Development of a Multi-layer V-model Design Process and Computational Tools for Mechatronic Conceptual Design

submitted by Hani Balkhair in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

in Mechanical Engineering

Examining Committee:

Dr. Clarence W. de Silva, Mechanical Engineering
Supervisor

Dr. Ryozo Nagamune, Mechanical Engineering
Supervisory Committee Member

Dr. Dana Grecov, Mechanical Engineering
University Examiner

Dr. José Martí, Electrical and Computer Engineering
University Examiner

Abstract

Conceptual design is a crucial phase in the Design Development Process (DDP) of complex mechatronic systems. Yet, the available design support is not adequate for the Conceptual Design Development Process (CDDP), let alone the entire DDP. Typically, linear methodology, called the V-model, is used in the software development life cycle (SDLC) for the DDP. The developed DDP can as well aid in addressing the customer requirements properly according to the degree of detail that is sought. Also, a Conceptual Integrated Model (CIM) can describe products from different viewpoints and can be developed to aid the simulation-based design.

The primary focus of the present thesis is the conceptual design phase. The thesis proposes a hierarchical DDP, where the V-model process is expanded into multiple layers. These layers assist in providing increased flexibility to the DDP, in which each design phase is subjected to a separate and independent integration and evaluation. Through this approach, the required functions can be realized, and the lengthy iteration loops, due to incompatible subsystems, are avoided. The second key objective of the present thesis is to develop a CIM for formal concept modeling using the modeling language SysML with generic design functional libraries. The first set of design libraries are the FB libraries, which aid in the development of the functional structure. The second set of design libraries are Amesim simulation software elements, which help establish the concept simulation models. Also, the challenges of the transformation and exchange of information between a descriptive modeling language – SysML – and a multi-physics modeling language – Amesim – are explored. The last key objective of the present thesis is the use of fuzzy measures and fuzzy integrals for the evaluation of the non-functional requirements of the conceptual design phase, where Sugeno lamda-measures are employed to address the uncertainty of the requirements.

The present research is conducted starting with a descriptive study for the development of a design concept model, concept simulation, and concept evaluation of an industrial fish cutting machine, which falls into the category of complex mechatronic systems. The evaluation of the approach focuses on improving the quality of the conceptual design.

Lay Summary

The development of the design process is crucial for the modeling and design of mechatronic systems. This dissertation addresses many challenges in the design process of a mechatronic system especially in the early stage of the design. A multi-layer design process is developed to improve the flexibility and responsiveness of the process. It facilitates the data management between different levels of the design process. The development, improves the modeling of the system functions in SysML to increase the modeling formality. An algorithm is developed to automatically generate structural models from functional models. Finally, an evaluation scheme is presented to simultaneously evaluate all conflicting criteria.

Preface

All the presented work in this dissertation was conducted by Hani Balkhair in the Industrial Automation Laboratory (IAL) at the University of British Columbia (UBC), Vancouver. The work was under the direct supervision and guidance of Dr. Clarence W. de Silva, Professor of Mechanical Engineering at UBC. Dr. de Silva proposed and supervised the overall research project, provided research facilities in IAL, and revised dissertation and other publications.

Chapter 4 addressed the modeling challenges of the design data, in which it proposed "interconnection classifications." Part of this chapter has been published in [H. Balkhair and C. W. de Silva, "Data management for multidisciplinary mechatronic systems," *International Journal of Emerging Technologies and Innovative Research,* ISSN:2349-5162, Vol. 5, Issue 8, pp. 46–50, August 2018]. Hani Balkhair was responsible for the development of the concept formulation, experiment validation, and manuscript composition. Clarence W. de Silva supervised the project and revised the manuscript.

Chapter 5 discussed functional modeling and their different representations in the System Modeling Language "SysML." Libraries of functions were developed in Block Definition Diagram "BDD," and the implementations of the libraries in SysML were addressed. A manuscript based on this chapter was submitted to a journal. The main author was Hani Balkhair, who was responsible for the development of the different modeling representations in SysML, experiment validation, and manuscript composition. The co-author was Clarence W. de Silva, who supervised the project and revised the manuscript.

Chapter 7 investigated the evaluation of the different design solutions. An evaluation scheme was proposed and was based on the Mechatronic Design Quotient (MDQ). Part of this chapter was published in [H. Balkhair and C. W. de Silva, "A systematic approach for functional

decomposition of mechatronic system design using mechatronic design quotient (MDQ)," in *the Proceedings of the 9th International Conference on Computer Science & Education (ICCSE),* pp. 135–139, 2014]. Hani Balkhair was responsible for the development of the concept formulation, experiment validation, and manuscript composition. Clarence W. de Silva supervised the project and revised the manuscript.

Table of Contents

List of Tables

List of Figures

Nomenclature

$\boldsymbol{c_k}$	the k^{th} criterion
$\boldsymbol{e_{(i,j)}}$	flow from port i to port j
$\boldsymbol{g^i}$	notation for the fuzzy measure
$\boldsymbol{i_k}$	the individual performance indicator of the k^{th} performance behavior
$\boldsymbol{I}$	motor inertia (kg.m^2)
$\boldsymbol{k}$	PID Gain constant
$\boldsymbol{m_{set}}$	set of mapping between functions and simulation components
$\boldsymbol{p_n}$	input/output port n
$\boldsymbol{P_E}$	energy port
$\boldsymbol{P_S}$	signal port
$\boldsymbol{P_{EE}}$	ports for electrical energy = Current $i(t)$ and Voltage $V(t)$
$\boldsymbol{P_{MRE}}$	ports for mechanical rotational energy =Angular velocity $\omega(t)$ and Torque $T(t)$
$\boldsymbol{P_{MTE}}$	ports for mechanical translational energy = Velocity $v(t)$ and Force $F(t)$
$\boldsymbol{P_{HE}}$	ports for hydraulic energy = Volume flow rate $\varphi(t)$ and Pressure $P(t)$
$\boldsymbol{R_{set}}$	set of requirements
$\boldsymbol{R_a}$	armature winding resistance (ohm)
$\boldsymbol{S_i}$	simulation component i
$\boldsymbol{S_{lib}}$	simulation library
$\boldsymbol{S_f}$	simulation component family
$\boldsymbol{S_{mod}}$	simulation models
$\boldsymbol{T_i}$	PID integration constant (s)

$\boldsymbol{T_d}$ PID drivation constant (s)

$\boldsymbol{\lambda}$ Sugeno measure

$\boldsymbol{\mu_k}$ the density value (fuzzy measure) of the k^{th} criterion

$\boldsymbol{\pi}$ a permutation index

Glossary

ABS	Anti-lock Braking System
AHP	Analytical Hierarchy
BC	Boundary Conditions
BDD	Block Definition Diagram
CAD	Computer-aided Design
CAE	Computer-aided Engineering
CDDP	Conceptual Design Development Process
CIM	Conceptual Integrated Model
CN	Customer Needs
DDP	Design Development Process
DP	Design Parameters
EDA	Electronic Design Automation
FB	Functional Basis
FBS	Function-Behavior-State
FM	Functional Model
FR	Functional Requirements
IBD	Internal Block Diagram
IPI	Individual Performance Indicator
MBSE	Model-based Systems Engineering
MCDM	Multi-criteria Decision Making
MDQ	Mechatronic Design Quotient
NIST	National Institute of Standards and Technology

OCL	Object Constraint Language
OOSEM	Object-oriented System Engineering
PV	Process Variables
RFLP	Requirement-Functional-Logical-Physical
SDLC	Software Development Life Cycle
SE	System Engineering
SysML	System Modeling Language
TGG	Triple Graph Grammar
UML	Unified Modeling Language
VDI	Association of German Engineers

Acknowledgments

I would like to express my profound gratitude to my research supervisor, Professor Clarence W. de Silva, for his supervision, knowledge, guidance, advice and inspiration throughout my entire research program at the University of British Columbia. I truly appreciate all the support and advice that he provided throughout the years.

I wish to thank all my colleagues in the Industrial Automation Laboratory (IAL), Dr. Roland Haoxiang Lang, Dr. Edward Yanjun Wang, Dr. Yunfei Zhang, Mr. Shan Xiao, Dr. Muhammad Tufail Khan, Dr. Lili Meng, Dr. Min Xia, Dr. Shujun Gao, Ms. Pegah Maghsoud, Dr. Yu Du, Dr. Teng Li, Dr, Jiahong Chen, Mr. Zhuo Chen, Mr. Tongxin Shu, Mr. Sheikh Tanvir, Mr, Bilal Riaz, Mr. Fan Yang, Mr. Lucas Falch, Ms. Swapna Premasiri, and Mr. Hiroshan Gunawardane for their friendship and help on both academic affairs and personal life.

Additionally, I would like to acknowledge the financial support for the research through the research grants from Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canada Foundation for Innovation (CFI), the British Columbia Knowledge Development fund (BCKDF), and the Tier 1 Canada Research Chair in Mechatronics and Industrial Automation held by Dr. Clarence W. de Silva. In addition, I would like to thank the Ministry of Higher Education of Saudi Arabia for their granted scholarship, and the financial support throughout my academic program.

Finally, I want to thank my lovely wife, Zainab Daghistani, and my two daughters; Fatimah and Batool. Also, the thanks are extended to my mother, my brother, and my other family members for their continuous support and encouragement.

Chapter 1: Introduction

1.1 Mechatronics and Integrated Design

The term “mechatronics” was coined in 1969 in Yasakawa Electric Corporation, Chiyoda-Ku, Tokyo, by senior engineer Tetsuro Mori. The word is composed of *mecha* from mechanics and *tronics* from electronics, which refers to the combination of mechanics and electronics. After the 1980s, the meaning has broadened to encapsulate computer technologies and software as an integrated part [1]. Nowadays, other disciplines such as optics, thermodynamics (including heat transfer), hydraulics, and pneumatics are involved in the development of mechatronic systems. Hence, a mechatronic system is regarded as a multi-domain (or multi-physics) and complex system. One definition of a mechatronic system is that it is a synergistic combination of precision mechanical, electrical, control, and systems engineering, for the design of products and manufacturing processes [2]. Other literature describes the mechatronic approach as a multi-disciplinary design methodology that solves the functions, primarily of mechanically oriented products through, the synergistic spatial and functional integration of mechanical, electronic, and information processing subsystems [3]. Mechatronic systems are becoming increasingly significant in many industries today, such as the automotive and manufacturing industries, in addition to modern consumer products. Typical examples include autofocusing cameras, engine management systems, food processing machines, anti-lock braking system (ABS) of automobiles, active suspension systems, and industrial robots.

Compared to purely mechanical solutions, mechatronic-system solutions can offer increased functional scope and quality through the integration of various disciplines. In the traditional electromechanical design, the mechanical domain is addressed first, which involves material properties (density, strength, deformability, etc.) and geometry (size, shape, etc.) of the

mechanical structure. Subsequently, the electrical domain, which involves electrical components (sensors, actuators, amplifiers, and other hardware), interconnectivity, and communications are addressed. Finally, a controller is integrated and tuned to rectify any shortcomings in the resulting design to add additional reliability [4]. In this design, tasks are not executed before the previous ones have been completed [5], [6].

In recent years, increased attention has been given to the development of the conceptual phase design processes for mechatronic systems, because the highest influence on the final design in the development process occurs during the conceptual development phase [7]. Fig. 1-1 shows the fundamental difference between design optimization and conceptual design generation. Performing possible improvements and optimization in the early design stage can make a significant impact on the success of a product. About 75% of the cost of a product is set during the conceptual phase [8]. The importance of the conceptual design phase stems from the realization that the level of overall design innovation and the quality are determined through this activity. Moreover, the type of technology and the concept of the design that satisfy the customer needs are specified during the conceptual design phase. In summary, the conceptual design phase may be viewed as the most critical phase of the product design life-cycle because the decisions made there have the greatest impact in the overall design process.

1.2 Challenges in the Design of Mechatronic Systems

Historical data show an increase in the complexity of the development of mechatronic products. This data points to a growth in the developed systems in terms of the number of functions, components, and interactions, as shown in Fig. 1-2.

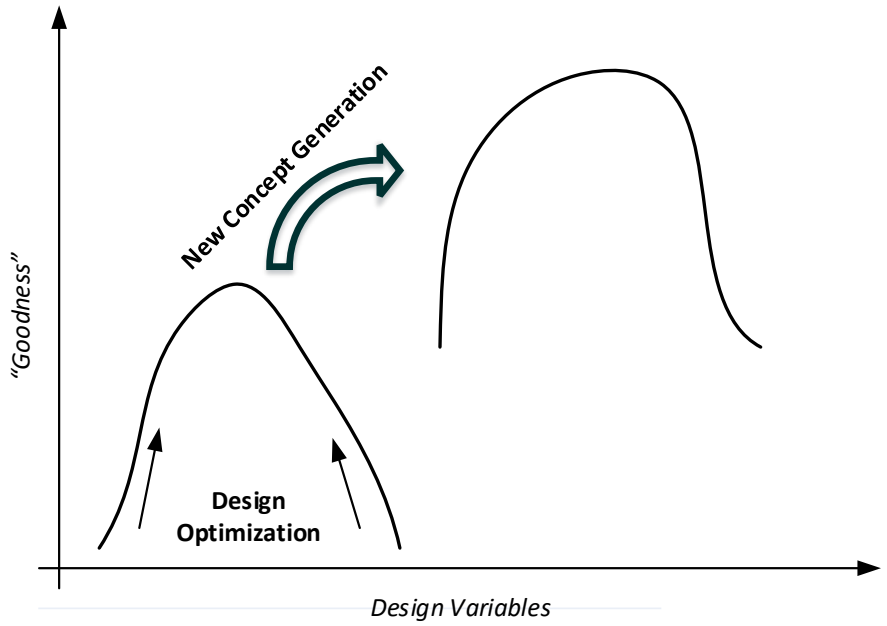


Figure 1-1: The difference between design optimization and the conceptual design

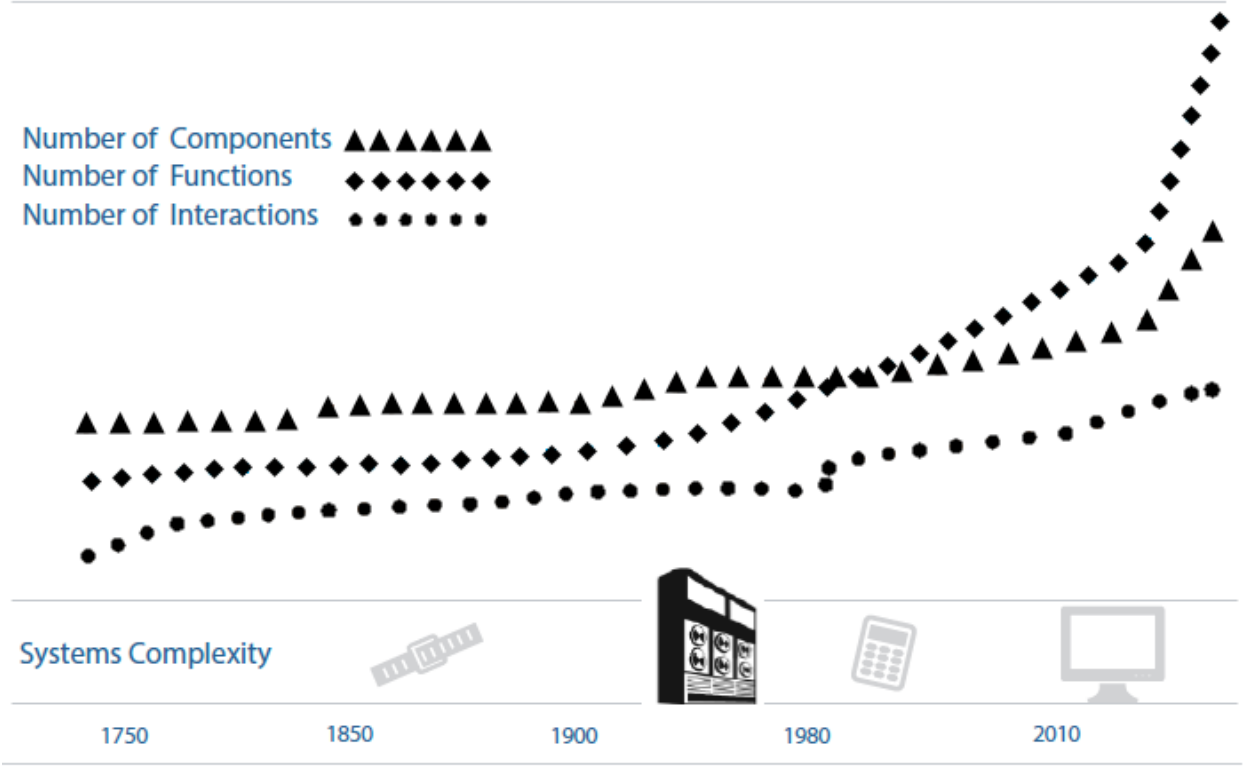


Figure 1-2: Historical data on the increase of system complexity

This design development complexity presents growing challenges for companies that demand technical solutions. We recognize two types of problems that can contribute to the rise of the design development complexity in mechatronic systems: they need to overcome (a) process-based problems, and (b) design data-related problems [9], [10].

1.2.1 Process-based Problems

These problems can be defined as "the coordination and synchronization of the discipline-specific development process, the coherence, and interactions between different disciplines and comprehensive integration across all disciplines" [9]. An additional challenge related to the process-based problem is the increased rate of dynamism due to high, and rapid market demands. Consequently, the traditional approaches are reaching their limit and are often too rigid to handle the large number of rapid design changes efficiently. Therefore, the development of design methodologies for the Design Development Process (DDP) to facilitate the integration of multiple domains has attracted considerable attention.

In the traditional sequential design, after the system is built, it is typically difficult and somewhat costly to change a parameter or a component. Also, the design optimality will be hard to achieve even with a perfect controller. Furthermore, there may exist a lack of compatibility and efficient matching between components and subsystems. This incompatibility arises from the dynamic interactions between components and subsystems, which cannot be appropriately taken into consideration in a sequential design scenario.

A key characteristic of a mechatronic system is the presence of different physical subsystems such as mechanical, electronic, and computer technologies that are integrated. An important effect of this integration is the creation of product innovations through the synergetic interaction between various engineering domains. Hence, DDP requires multidisciplinary and

holistic solutions that are able to realize such systems. Nonetheless, the established DDP suffers from considerable deficiencies in managing process-based problems. Nowadays, companies are struggling with new challenges in mechatronic system design, which require the use of innovative design processes. These challenges make it difficult to manage the development process that concerns increased quality, and reduced development costs and time [9].

In most cases, the process of design development during the conceptual phase still performs the involved discipline in a separate and isolated fashion [11]. Since the conceptual design phase is crucial in the design of mechatronic systems, as discussed in Section 1.1, the associated work is heavily based on the Conceptual Design Development Process (CDDP). Consequently, the process-based problem discussed here will consider challenges such as: the coherence and interactions between different disciplines, the comprehensive integration across all disciplines, and the lack of flexibility in DDP.

Although investigators and the application sector have put some effort in addressing these challenges, some challenges still remain, such as [9]:

- The standard practice in industry still involves the traditional, sequential design process.
- There is a lack of support tools for the synchronization of different disciplines in the design development process.
- The coordination among the activities and tasks in the conceptual phase and the other phases of the product design development process is not sufficiently supported.
- The complex coherences and interactions between the disciplines are only considered in a later development phase.
- A flexible organizational structure is needed. The structure should be able to adapt to the rapid changes in the requirements.

- Different ways to encapsulate the increased demands of the customers and stakeholders throughout the development process need to be developed.
- Current development processes are not adequate to respond to the rapid changes in customer requirements.

1.2.2 Design Data-related Problems

The data that are created throughout the DDP of mechatronic products need to be properly managed. Product models are used to support the product data management (PDM), in which all the pertained information is accessed, stored, served, and reused by stakeholders [10]. Computer-based tools used for the support of product data have always been developed for a specific discipline such as Computer-aided Technologies (CAx), Electrical/Electronic Engineering Solutions (EES), Computer-aided Software Engineering (CASE), and Product Lifecycle Management (PLM) [9]. These tools often produce data about the product model and product structure that may be incompatible with one another.

In addition, the target of a design is to meet the requirements and needs of the customers and stakeholders using technical solutions in a rapid and satisfactory manner. At the same time, economic efficiency must be acceptable. The degree of satisfaction of the customer dominates the extent to which the product solves the specific problem. Also, the framework conditions of project development are subject to increased uncertainty and dynamism, which represent a challenging trend. For example, stakeholders are less and less able to explicitly express their needs or product requirements [12], [13], available development time [14], and the functional scope and interdependencies of the functions among themselves [15].

Therefore, the diversity of data from different disciplines brings challenges, which include the following:

- It is difficult to show, understand, and construct the interdisciplinary and functional relationships between various systems and components.
- Increasing the efficiency of directing and organizing the design development process can be achieved by using the information extracted from product models.
- Conceptual Integrated Model (CIM) that models the dynamic behavior of different multidisciplinary systems, functions, and components are not adequately realized.
- Different classes of customer requirements should be properly addressed in the corresponding phases of the development process.
- The imprecision and incompleteness of the design requirements pose challenges in the product data analysis and exchange.
- Developing methodologies to evaluate the conflicting requirements of different customer should be further investigated.

1.3 Requirements of Mechatronic System Integration

Mechatronic systems have displayed success in developing complex and advanced products thanks primarily to the close integration and collaboration of mechanical engineering, electronics, and computer science. Mechatronic systems are characterized by being multidisciplinary, highly complex, and are subject to rapid changes and conflicting requirements. Therefore, specific requirements need to be met and suitable procedures have to be followed in their development.

A practical approach to managing the mentioned problems is through the exploration of the conceptual design phase of the DDP and CIM. This phase is particularly important because the decisions that are taken in this design phase, and the data management have the highest impact on the rest of the design decisions. Moreover, the conceptual phase addresses the abstract, concept,

function, behavior, and the general form of the designed components and systems, which creates increased ability to address the multidisciplinary design. Also, this phase of design has the advantage of increased availability of design freedom. In addition, there are other challenges, which are related to the design in this phase. However, we believe that addressing the mentioned problems during the concept development is an effective approach to tackle the increase of the design complexity and the number of functions, components, and interactions.

In order to manage the high complexity in the design of a mechatronic system, several key requirements need to be addressed, which are indicated next.

1.3.1 Conceptual Process-based Requirements

The development of increasingly mechatronic products presents designers with new challenges that require the use of adapted processes and methods. These challenges have created a new and growing demand for a comprehensive process model. This model should encapsulate a holistic view of the process while facilitating the cooperation and coordination of the involved disciplines. It can be represented as a general structure with specific guidelines for designers to ultimately satisfy the needs and requirements of the customers. Inexperienced designers tend to start with a preliminary concept and proceed towards the detailed design without proper utilization of the design freedom. Such an approach has a high possibility of leading to inferior design selections. Therefore, the current practice is to generate different conceptual design alternatives, which can help find a good solution [16]. This approach can reduce the expense of time and cost of the development process.

Decision making is used to evaluate the conceptual design alternatives in different levels of detail in an iterative matter. The iteration should provide a less expensive design in a particular cycle than in the previous cycle. The early analysis and simulation of different design alternatives

are essential even in the absence of rigid mathematical parameters and the presence of various indefinite constraints [17]. Applying an extensive evaluation scheme to the design development process may result in a fewer number of iterations. One way to satisfy that condition is to subject each phase to a separate and independent evaluation scheme. Also, an approach to systematically guide and organize the mechatronic design process of products can ultimately reduce the cost, time, effort, and needed resources of the development.

These requirements can be summarized as follows:

- The development of advanced approaches to organize, manage, and guide DDP is necessary.
- An increase in the flexibility of the DDP of mechatronic products is required in order to increase the evaluation capability and the level of satisfaction of the customer requirements.
- Different engineering disciplines should be integrated at the beginning of the DDP.
- A solution-neutral, and domain-independent specification, description, and definition, in the early development stages should be supported.

1.3.2 Conceptual Data-based Requirements

The data of product models are used to describe the links, connections, and interfaces of product elements and functions of various domains, and in different levels of detail. The necessity of viewing the integrated and complete mechatronic system alongside the interfaces and connections between the associated different disciplines, throughout the entire DDP of the system, is essential. Also, a common language is necessary in order to enable traceability and reasoning between different components and functions of the designed system.

Model-based Systems Engineering (MBSE) is a multidisciplinary approach to help understand the context and specification for satisfying the specified customer requirements by developing a system solution in response to different needs of the stakeholders [18]. Aspects of MBSE include behavioral analysis, system architecture, requirement traceability, performance analysis, system simulation, testing, and so on. [19]. System Modeling Language (SysML) is an extension of MBSE and can be utilized as a computational model of a mechatronic product.

Other relevant key factors include the uncertain and limited knowledge about the product design and customer requirements, and the lack of communication between components and subsystems, which need to be properly addressed. Therefore, the following criteria need to be fulfilled:

- The development of early design stage modeling is needed for CIM that moves beyond geometry to replace paper-based modeling methods [20].
- CIM should allow representation of the product's behavior in an integrated multidisciplinary system. The data model should illustrate the details of the individual disciplines to support the design, analysis, and evaluation of the overall system.
- The models must display abstract mapping of the product functions, activities and components, and their dependencies. They should also provide information about the internal changes between the disciplines to aid in the product's development process.
- The models may also be able to contain meta-model information, in which the traceability, and reasoning between systems, sub-systems and components are permitted.
- Data of the product model should be used in the advancement, guiding, and organization of the process development models.

- New approaches and methods should be investigated to increase the confidence of the models in the presence of a lack of information and communication.
- CIM should be able to transfer the stored information and knowledge to and from other models (e.g., simulation or behavioral models) for further computational analysis and simulation.

Library-based Support for the Conceptual Design

Sources of knowledge and information are essential to assist mechatronic CIM. There is a large body of knowledge for designers that is captured from past designs, which can help during the design activities of ever-increasing mechatronic design problems [21]. Knowledge and knowledge modeling are used to aid the conceptual design phase; as opposed to geometrical modeling, which is used to support the detail design phase. Knowledge reuse is essential to achieve the targets of systems engineering vision 2025 [22], [23].

Design libraries are an important resource to support product modeling, in which all conceptual design knowledge is captured and classified into different categories [24]. For example, the systematic reuse of such libraries in object-oriented software development is widely used, where the libraries offer basic support of various functionalities [25]. One way to support the design libraries is by classifying the captured knowledge to categories; for example, functional libraries [26], behavioral libraries [27], form libraries [28] and so on.

Several requirements are listed below to improve the usage of libraries in engineering design:

- The design libraries should raise the formality, in which systematic guidance for the design through the reuse of clearly defined items from the libraries can be further investigated.

- The use of the design libraries to further support the CIM. Some examples are automatic model generation, compatibility and consistency checking, or the evaluation of design alternatives.

1.3.3 Multi-criteria Evaluation

The elimination of different design alternatives in an early stage without using extensive details of the system is a desirable practice. However, the extensive search space of design alternatives makes the identification of design concepts a tedious process.

The dynamic interactions of different components and subsystems degrade the performance of the overall mechatronic product, if not taken into account during design. It exists because of the lack of compatibility and improper matching between components and subsystems, and these are not usually taken into consideration when evaluating the design solutions.

Design criteria requirements provide a measure of how well the system should function or behave. In the same context, the design specifications may include system attributes, and constraints. Determining the design criteria requirements is done after reviewing the design tasks and customer needs, and on performing a requirement analysis. The attributes and sub-attributes selected in this manner have to be operational in order to identify how well each conceptual design solution meets the design requirements. Therefore, the following points should be addressed:

- The design evaluation should take into account the complexity of both, correlations between system requirements and interactions between multidisciplinary subsystems.
- Efficient and effective methods are needed to decrease the design optimization time and computational costs, which result due to the presence of many design variables and a vast search space.

1.4 Research Objectives

The main objective of the present research is to support the design of mechatronic systems by improving the CDDP and CIM. The proposed development of the conceptual design phase is built upon a general reconstruction of the DDP, which is termed the "Macro-level development process." Specifically, the design phases are broken into separate phases, in which each phase is guided by independent modeling, simulation, and evaluation schemes. At the same time, the systematic design guidelines and formality will be maintained. This modification will increase the flexibility of the development process, leading to the following key advantages:

- The ability to consider diverse requirements of stakeholders, in which the level of detail of the requirements is properly incorporated.
- The reduction of the processing time and the resource costs can be achieved by minimizing the development cycles and interactions.

The development of the conceptual design phase will be investigated further, which will include the formulation of the "conceptual macro-level development process." The present thesis will address three aspects of development investigation: concept design and modeling, run-time concept integration and simulation, and concept analysis and evaluation.

In the present work, Model-based System Engineering (MBSE) will be heavily used for the development of the "conceptual micro-level development process" through the System Modeling Language (SysML). The goal is to develop an integrated tree-based concept modeling approach in SysML. The developed model will improve the consistency checking and traceability with respect to the computational support and the ability to display different levels of model details. An implementation of different design libraries into the model will be developed in order to increase the formality and minimize the errors. We believe that DDP controls the design and

provides baselines that coordinate the design efforts. CIM provides a suitable structure for solving design problems and can integrate involved customers in the DDP to ensure the developed system is viable throughout the product life cycle.

A seamless, logical, and systematic transformation of the model between different modeling environments will be established, where knowledge extraction will be performed between a descriptive modeling environment and a simulation modeling environment. The first environment supports the abstract level modeling where functional libraries [26] will be introduced and implemented. The second environment supports the concept level simulation where behavioral libraries [27] will be introduced and implemented.

Finally, a formal approach for the evaluation of conceptual design alternatives of mechatronic systems will be developed. It will be a general approach that covers a wide range of systems within the umbrella of mechatronic systems. Also, it will be able to incorporate multiple criteria in the design evaluation together with an intuitive aggregation method. The developed approach is built on the concept of Mechatronic Design Quotient (MDQ) [29] and enhances the application and tools for the developed concept.

1.5 Contributions and the Organization of the Thesis

The main contributions of the present dissertation may be summarized as follows:

1. A multi-layer design process structure for mechatronic systems, based on the V-model, is developed. Compared with the other design processes, the presented framework maintains the systematism, increases the flexibility, and improves the integration and evaluation capabilities. In addition, the proposed structure adequately addresses the customer requirements at different levels of detail. The multi-layer nature of the process allows the designer to revisit the requirements in every phase of the process.

2. This work introduces “Interconnection Classifications” for modeling the communication data that take place between and within the various activities in the early phase of the design process. These activities are the requirements modeling, the functional modeling, and the structural modeling. The communication data are computationally modeled in UML. The data modeling provides a common terminology and language between different design teams. This helps to reduce the information misuse in an early design phase.
3. The development of a library of functions in SysML, to support the functional modeling, is presented. This library is used to support the modeling of the functional model in SysML. Such support would increase the reusability and consistency of the model. Also, the usage of the library helps in the adaptation of the model to modifications. Compared to other existing work, the present work improves the formality of the SysML model through the utilization of SysML diagrams, such as Block Definition Diagram (BDD), and Internal Block Diagram (IBD) for the modeling.
4. An algorithm is developed that enables an automatic transformation between functional models and structural models. In addition, the algorithm ensures the satisfaction of the customer requirements during the model transformation. This requires a precise algorithm description of the requirement model, functional model and structural model. The support of simulation libraries is exploited. The synthesizer algorithm dissects the simulation library components and matches the interfaces of these components with the corresponding functions. The advantage of the synthesizer algorithm is that it increases the model accuracy according to the requirements, and eliminates any biased selections. This work illustrates the utilization of the algorithm in generating different kinematic behaviors of two functional sub-models.

5. An evaluation methodology scheme is developed for mechatronic systems. The proposed methodology is used to evaluate different design solutions. It takes into account the interactions between different design heterogeneous components. In addition, the conflicts in the design criteria are considered. This work proposes an Individual Performance Indicator to reveal the behavior of different simulation outputs with respect to the criteria. Lamda-measures are employed to calculate the weights of interaction between the criteria. Finally, the Mechatronic Design Quotient is used to aggregate all the criteria, with their weights and the individual performance indicators.

The overall result of these contributions is the development of an integrated, unified, systematic, and unique systems. The presented design process conserves the systematism. Functional modeling in SysML provides a domain-independent system model. The evaluation methodology scheme produces integrated and unique systems.

The rest of the present dissertation has the following structure:

Chapter 2 introduces different system engineering approaches for mechatronic design processes. More details of Axiomatic design and the different V-model variations are provided. Model-based system engineering is presented with the focus on UML and SysML. Function-Behavior-State model framework is discussed in this chapter, and their different corresponding knowledge libraries are introduced. It presents different model representations and model transformations that are used during the conceptual design process. Finally, the chapter discusses the design evaluation of mechatronic systems.

Chapter 3 introduces an overview of the developed design process methodology and the details of the design process model. The chapter first presents the proposed macro-level design process model, where the multi-layer V-model design process is described. Second, the chapter

discusses the proposed micro-level design process model, in which the characteristics of the design tasks and activities within the macro-level design process are given. These two models are based on Model-based System Engineering for supporting the integrated development process of a mechatronic system. The chapter demonstrates the presented design methodology through an industrial fish cutting machine, which investigates the rationalization of the choices of the current solutions.

Chapter 4 discusses the details of the micro-level process model, where the underlying organization of the different design activities in the concept and modeling sub-process phase are defined; specifically, requirement modeling, functional modeling, and structural modeling. Moreover, the chapter describes the relationships between different design activities, which introduces the proposed interconnection classifications; namely, *Allocation*, *Type*, *Class*, *Conversion*, and *Confidence*. A case study is presented where these interconnection classifications are implemented.

Chapter 5 describes the modeling approach of the conceptual phase in SysML. In particular, the functional modeling and its library, i.e., the Functional Basis, which provide additional reinforcement for the conceptual design development phase, are presented. It introduces a computational modeling approach for the functional model in SysML, in which the *Block Definition Diagram* is utilized. The representation of the functions and ports in SysML is also discussed, where a library of functions is developed. Finally, the chapter presents an implementation of the proposed functional modeling approach in SysML, as a case study.

Chapter 6 describes how structural modeling is established, where a set of components/subsystems are interrelated. It illustrates the development of an algorithm for transforming the functional model into a simulation model. The chapter describes this

transformation, which requires an algorithmic description of the requirement model, functional model, the simulation model including the simulation library, and simulation components. Then, the principles and the algorithm of the proposed simulation synthesizer are presented. This synthesizer is demonstrated in a case study of an electro-mechanical conveyer system.

Chapter 7 presents a developed evaluation scheme for the evaluation of conceptual design solutions of mechatronic systems. The underlying principles of the evaluation indicator, which is based on the Mechatronic Design Quotient (MDQ), are demonstrated. The chapter describes how the interaction between different design criteria is addressed, even in the presence of insufficient information. The proposed framework for the system evaluation is given, and it is demonstrated in a case study.

Chapter 8 concludes the dissertation by summarizing the main research contributions. It also discusses the possible directions for future research.

Chapter 2: Background and Related Work

An overview of the thesis topic and important related work of the research are presented in this chapter. It begins with the available models for a design process and the design methodologies for mechatronic systems, with a focus on the conceptual design phase and its importance. The system engineering approach is then introduced, which includes a description of MBSE and SysML.

2.1 Design Process of a Mechatronic System

A design of a system can be considered as an interplay between *what* we want to achieve in the system and *how* we want to achieve it [30]. To go from what to how, the designer is challenged by many conflicts, trade-offs, and risks. A designed product is not created just in a single big step, but rather in many small steps, which must be precisely defined, and their interfaces must be precisely described. The resulting sequence of steps is called a " design process" [31]. Designing a new product progresses through a sequence of steps, which can be used as guidelines for the designer. The overall process steps are referred to as the "Design Development Process" (DDP) [31]. It summarizes several activity steps to achieve the intended result, starting from the product concept to the finished product. In general, the DDP can be described as the process of organizing and developing a plan to transform a concept into a final product. The development of the design process itself is often very complex, depending on the objectives and the complexity of the designed product. Decisions taken during the DDP of a product will impact on the design of the product. As discussed in section 1.2.1, the essential functions of mechatronic products, in contrast to traditional electro-mechanical engineering products, are characterized by the “integrated” interaction of mechanical, electrical and information technology subsystems. The

development of mechatronic products thus requires the goal-oriented and efficient integration, specifically collaboration, of the involved disciplines in the DDP.

Since the end of the nineteenth century, efforts have been made to systematize the design process of a system and to carry it out in a targeted manner. As a consequence, a design methodology was developed [32]. Since the 1940s, many research outcomes have been published on methodological designs of mechanical engineering products, in both Europe and the USA [33], [34]. Significant challenges in the development of new products today arise through the continuous development of associated technologies. Therefore, several DDPs of products that describe these steps have been proposed. In order to manage the challenges during product development, especially in mechatronics, a structured, systematic and goal-oriented approach in product design and development is needed, leading to high-quality results [35]–[38].

2.2 System Engineering Approaches

In the late 1950s and the early 1960s, System Engineering (SE) has been used as an approach for multidisciplinary and concurrent design of complex systems [39]. A concurrent (or, integrated) design process is a way to decrease the process development process time and to manage the synergy of a multidisciplinary design [40] while simultaneously addressing all physical domains (e.g., mechanical, electrical, fluid, and thermal) of the problem [4]. SE is a science that is applicable to the design, development, and maintenance of highly complex products such as trains, cars, airplanes, power plants, and manufacturing processes. The goal is to analyze and combine all the behaviors of the system into an efficiently functioning design. In the SE context, a system can be defined as “a set of elements that interact to achieve a stated purpose” [41]. Systems can be classified into four main characteristics: Closed/open systems, Natural/human-made/human-modified systems, Physical/conceptual systems, and

Precedented/unprecedented systems. A wide variety of combination of system characteristics can lead to many types of systems, each of which would have different properties [41].

SE relies on a system-centered thinking to solve problems. It seeks to develop a system based on an initial abstract model of the system with input and output quantities, a system environment, and an initially-unknown inner life. On this basis, an attempt is made to develop a fundamental understanding of all internal and external interactions of the considered problem (system) [42]. SE enables a transparent process across the entire development cycle. SE is a cross-functional approach, and a means to enable the successful realization of a requirement-based system. It focuses on defining the customer needs and the required functionality early in the design development process, documenting the requirements, and then proceeding with the design synthesis and the design validation, while taking into account all aspects of the product lifecycle. SE considers both economic and technical needs of all customers, to provide a high quality product that meets the user needs [43].

The work in the present dissertation is heavily built based on the SE approach while considering multidisciplinary aspects of the system early in the development process. The system characteristics applied in the present work concern open and physical systems that are human-made/modified largely from available elements.

Since the 1980s, different design process models have been used for SE; for example, a waterfall model [44], spiral model [45], and V-model [46], to handle the increased complexity of mechatronic systems. These approaches are still inefficient as they do not account for integration of different physical domains in an early stage of the design process – the conceptual design [17]. The development of new approaches for abstract modeling and evaluation of the associated

concepts at an early stage of the design process is needed. In the next section, an overview of some common system engineering design approaches is discussed.

2.2.1 Systematic Design Approach

Widely acknowledged engineering design guidelines developed by Phal and Beitz characterize the design phases into four stages: product planning and clarification, conceptual design, embodiment design, and detailed design [17], [37]. This approach has been included in design textbooks [16], [36]. Table 2-1 summaries the associated activates in each phase.

Table 2-1: Activates During Design Development Process.

Design Phase	Activates
Product Planning	• Market analysis • Finding and selecting product ideas • Defining the intended functionality and requirements of the product
Conceptual Design	• Establishing detailed functionality • Identifying solutions to functions • Combining solutions into working structures • Selecting combinations of solutions • Developing principal solution variants • Evaluating variants
Embodiment Design	• Identifying product layouts and form • Finding solutions to auxiliary functions • Developing detailed and compatible layouts for main and auxiliary functionality • Evaluating and optimizing the design
Detailed Design	• Finalizing layout and creating drawings • Developing assembly drawings • Completing production documents • Checking documents for compliance, completeness, and correctness

For the potential success in the design of a mechatronic system, a systematic approach is necessary for the early stages of the product development process, where the emphasis is on

modeling and model analysis. In this context, the aim is to realize products that always meet the desired performance requirements despite the presence of a wide variety of external influencing factors. Earlier customer involvement will enhance the interactions with customers and customer integration. The customer-oriented development of individual products poses considerable challenges such as continued change of the internal and external framework conditions, and the customer requirements, during the design process. These challenges lead to a further increase in the complexity of the development process, in which the main focus is on the development of products whose functional fulfillment is ensured despite a wide variety of influences [47].

2.2.2 Axiomatic Design

Suh [30], [38] has attempted to establish a product development methodology based on a system of axioms. Every design process involves four different areas: the customer area, the functional area, the physical area, and the process area (Fig. 2-1). A set of variables characterizes each of these areas. In the customer area, this information encapsulates the desired product properties $\{CNs\}$. When entering the functional area, the desired product properties must be translated into functional requirements $\{FRs\}$ and boundary conditions $\{BCs\}$. In the physical domain, design parameters $\{DPs\}$ must then be defined that will fulfill the functional requirements while complying with the boundary conditions of the product. The design process is completed by developing appropriate manufacturing processes for the product, which are defined by process variables $\{PVs\}$.

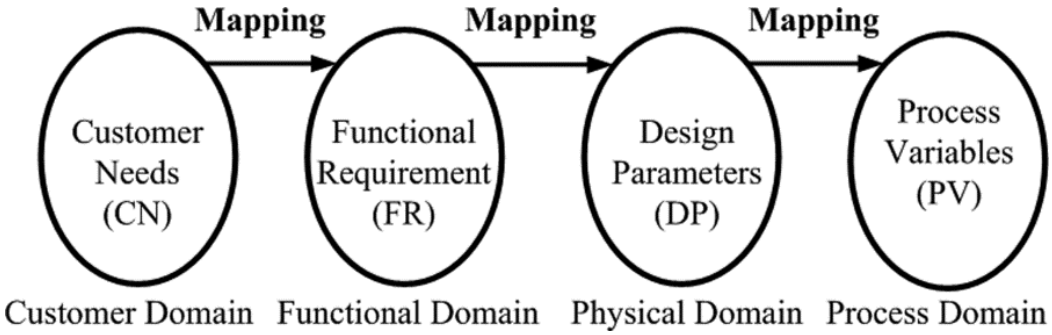


Figure 2-1: Four areas of the design process.

Suh has described the transitions between the areas of the design process mathematically with the help of a matrix formulation. For example, to determine the design parameters {*DPs*} of the system from the functional requirements {*FRs*}, the following "design equation" is applied:

$$\{\boldsymbol{FRs}\} = [\boldsymbol{A}]\{\boldsymbol{DPs}\} \tag{2-1}$$

The design matrix [**A**] uniquely associates the design parameters of a given solution with the functional requirements, and it takes the following three forms:

$$[\boldsymbol{A}] = \begin{bmatrix} \boldsymbol{A11} & \boldsymbol{A12} & \boldsymbol{A13} \\ \boldsymbol{A21} & \boldsymbol{A22} & \boldsymbol{A23} \\ \boldsymbol{A31} & \boldsymbol{A32} & \boldsymbol{A33} \end{bmatrix} \quad \text{“Coupled Design”} \tag{2-2}$$

$$[\boldsymbol{A}] = \begin{bmatrix} \boldsymbol{A11} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A22} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A33} \end{bmatrix} \quad \text{“Uncoupled Design”} \tag{2-3}$$

$$[\boldsymbol{A}] = \begin{bmatrix} \boldsymbol{A11} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A21} & \boldsymbol{A22} & \boldsymbol{0} \\ \boldsymbol{A31} & \boldsymbol{A32} & \boldsymbol{A33} \end{bmatrix} \quad \text{“Decoupled Design”} \tag{2-4}$$

They are the, "coupled design," which has a general matrix, "uncoupled design," which has a diagonal matrix, and "decoupled design," which has a triangular matrix. While in an "uncoupled design" each design parameter *DP* is uniquely linked to a functional requirement *FR*, "coupled design" and "decoupled design" have dependencies between the design parameters, which make the fulfillment of the functional requirements difficult or impossible.

"Axiomatic Design" appears like a rather independent and methodological approach. This is due to the mathematical forms that are built to describe the relationships between the domains.

However, it is not clear from the approach how the mathematical connection between the functional requirements and the design parameters are realized for a complex system.

2.2.3 Variations of V-Model

The difficulties in planning and implementing mechatronic development processes arise mainly from the large number of sequential and parallel dependencies that must be considered simultaneously. The Association of German Engineers (VDI) developed formal guidelines for the design of technical systems [48]. Different evolvements of the guidelines have been developed for specific tasks. For example, VDI 2221 [49] is intended only for mechanical systems, and VDI 2422 [50] is for mechatronic systems controlled by a microcontroller. VDI Guideline 2206 [50] has been developed to support the cross-domain development of mechatronic systems, especially in the early phase of the system design. Mainly three elements are used for this purpose; namely, the general problem-solving cycle as a micro-cycle, the V-model as a macro-cycle, and predefined process blocks for recurring work steps. V is meant to be used as a management tool, which shows the relationship between design activities and test activities. It is a practical way to represent the development process, and it is adopted from system engineering. However, simplistic straightforward use of V as a process model may lead to design defects, and inability to consider changes to customer needs during the product development. Therefore, Van Brussel has suggested an integrated (or concurrent) engineering approach that takes into account all involved physical disciplines from the beginning [51]. This basic idea is also found in [52] and in the VDI guideline 2206 [53].

The essence of the macro-level process of the V-model is generic and, therefore, some studies have suggested applying multiple macro-level cycles [54]–[58]. For instance, in [54] Gausemeier and Moehringer propose multiple macro-level layers that represent the degree of

product maturity. Each cycle has the same micro-level process with different level of detail. The first cycle represents "laboratory samples," the second cycle represents "early prototypes," and the third cycle represents "preproduction product." However, the number of macro-level cycles and degree of maturity can vary depending on the application. Vasic and Lazarevic [55] developed a different level of maturity representation, namely "laboratory specimen," "functional specimen," and "pilot-run product," [59] proposed the W-model, where two V-models are connected side by side forming a W-shaped structure, hence the name W-model. The macro-level development process includes five design phases: "system analysis," "specific solutions and dependency analysis," "virtual system integration," "Model analysis and detailed development," and "system integration." The extra element in this form that is not present in the regular V-model form is the addition of the central part, the "virtual system integration," which increases the capabilities of cross-domain integration and verification.

Several additional developments have been made to the micro-level processes of the V-models. [53] proposes a customized micro-level process of the guidelines for the development of mechatronic systems controlled by a programmable logic controller (PLC), which enabled the inclusion of further sufficient process guidelines. The developed method was tailored only for machines controlled by PLC. There, the domain-specific design neglected the electrical domain and was developed for modeling and design of 3D CAD, and for PLC programming environment only. [60] developed a micro-level process that comprised four phases: Requirement engineering, Functional design, Logical design, and Physical design – RFLP. However, since a separate specialized computational software manages each phase, the data exchange between different tools and the other domains remains a challenge.

2.2.4 Model-based System Engineering (MBSE)

An approach relevant to the development of mechatronic products in a systems engineering framework is "Model-based Systems Engineering" (MBSE). It has drawn increased attention recently as it is recognized as a powerful methodology for the design of complex mechatronic systems [61]. A model in the context of MBSE refers to an abstract representation of a system, sub-system, or component, to raise the level of understanding of the real system.

In MBSE, system models are placed at the center of development and is used for managing the SE process of all phases of system development (specification, development, integration, validation), which represents a view that is different from the traditional, document-based development. In the MBSE, the development process is an iterative sequence of activities for the development and creation of increasingly detailed models or an overarching system model over time [62].

The MBSE approach is becoming an increasingly important means in the development of mechatronic products [63]. There are a variety of approaches to the model-based development of mechatronic systems with different emphases such as the modeling of components [64], knowledge management [65], the support of architecture development [66], and the management of models [67]. Another significant benefit of the MBSE approach is its use for improving the coordination and communication between different disciplines in the domain-specific design phase of the development process. By working on the communication between the integrated system models, information about the current state of development is always available for all disciplines. Moreover, when changes occur, their effects on the other subsystem models are detectable, and hence the overall transparency and understanding of the system is increased. The current research

activities on MBSE are focused on the early development process, with additional support for functional requirement, and dependency modeling.

There exist a large number of process models, methods and tools for supporting model-based development of a system. The system is usually modeled using UML (Unified Modeling Language) and System Modeling Language (SysML). They have widely used modeling languages for the aid of the MBSE, which are provided by the Object Management Group's System Modeling Language [68]. MagicDraw [69] is a computational software that enables the modeling language UML, and SysML, which comes with extensions for system simulation [70]. However, there is still a lack of recognition and practice of MBSE in industry, which points to the need for further development of MBSE with regard to usability [71]. Also, there is a demand to integrate MBSE with analysis tools to support dynamic analysis.

UML and SysML

UML (Unified Modeling Language) is a semiformal, graphical language that is used in software development. Its use ranges from modeling to analysis of software programs. It facilitates map structures, architectures, system behavior, and the interaction with other systems through diagrams. Essential components of these diagrams are objects and classes. They contain specific attributes and methods and thus form the basis of the modeling language.

An object is generally a graphical representation of an object. It can be a model of real facts, things, or concepts. It has specific characteristics and reacts to defined requests with a given behavior. A class describes a collection of objects with the same properties (attributes), common functionality (methods), common relationships to other objects, and common semantics. All objects of a class have the same attributes but different attribute values [43].

UML is an object-oriented modeling language which that facilitates mapping of all the necessary aspects of a software program. In addition to the representation of all models, there are fixed definitions of all interfaces and connections. However, the reactive level of control engineering is not supported. Besides, UML cannot be regarded as a cross-domain language in system development.

While UML is intended for software development, SysML addresses system engineering and thus the holistic and cross-disciplinary modeling of technical systems. SysML is a semi-formal, graphical language for the modeling, analysis, and verification of systems. It is based on UML but different with regard to diagram types. Specifically, UML diagrams are reused and extended by it, as shown in Fig. 2-2 [68]. For this purpose, some adjustments and extensions to the UML have been made; for example, incorporation of classes called blocks.

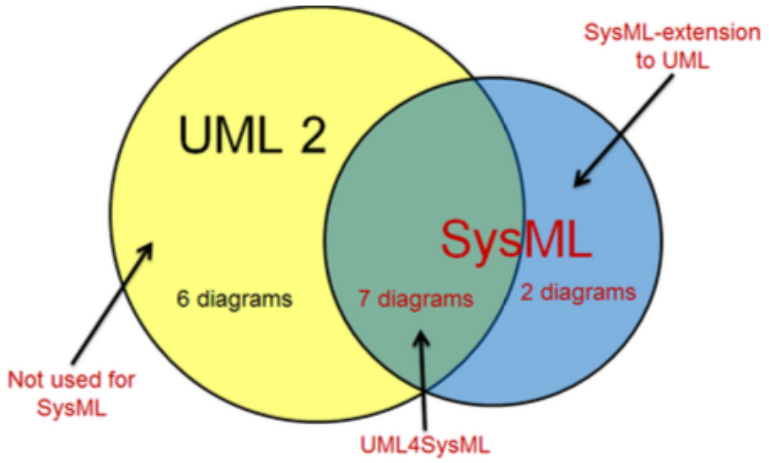


Figure 2-2: Relationship between UML and SysML.

SysML aims to provide a language that facilitates capturing of all different aspects of information about a system in an integrated model. It would increase the communication between different aspects of the model, and it decreases the ambiguity between the languages of the designers and the stakeholders. It aims to capture functional, behavioral and performance models,

and also capture the structural topology of the system, the parts of the system and how they are interconnected. Moreover, SysML incorporates requirements explicitly so that the extraction of information about the system can be realized.

SysML is composed of nine types of diagrams, which enable the description of various aspects of structure, requirements, and behavior. Fig. 2-3 shows different types of diagrams, some of which have been adopted unchanged from UML and others have been extended and partially renamed.

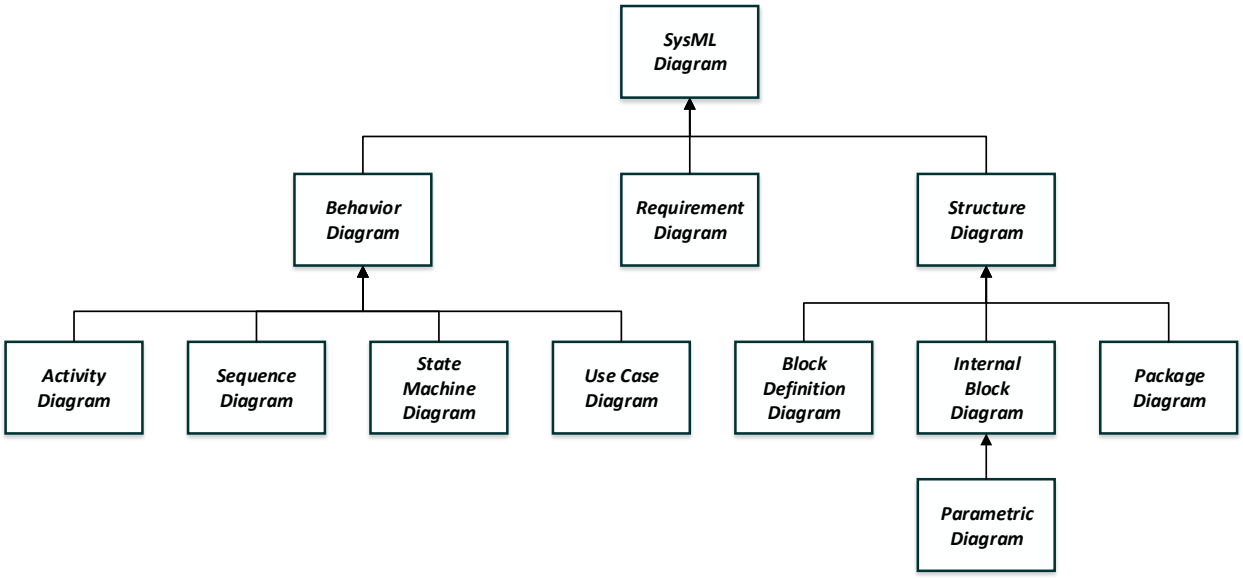


Figure 2-3: Taxonomy of SysML diagram

At the top of the hierarchy, there is a system diagram, and it is decomposed into four categories: Requirement diagram, Behaviour diagram, Structure diagram, and Parametric diagram. Requirement diagram allows the description of functional and non-functional requirements. In addition, the existing relationships between different requirements and the system can be specified and explicitly modeled. It can greatly help manage the requirements since they are represented as a part of the system. Moreover, constraints can be added, and requirements become constraints on the properties of the system.

The behavior can be modeled in SysML with four different diagram types. Use case diagrams can be employed to describe the interactions of all users or external devices with the developed system. State machine diagrams represent possible states of the system and state transitions. The criteria for transitioning between states and the causes or triggers of these transitions between states can be specified. Sequence diagrams concern logical ordering. They describe scenarios and sequences of events, and how the system components should interact with each other. An activity diagram describes the system processes including input and output data and represents flows of activities that can be tied to system elements in order to add a function or property to the element, to perform another operation on another part of the system. Furthermore, a decomposition of the activities is possible to derive a kind of functional hierarchy. These types of diagrams are very powerful in describing such aspects as the concept of operations. Use case analysis is mostly focused on early concept development or stakeholder requirements, addressing such questions as: how are they going to interface with the system, where do they drive value, and how does the user interact with the system?

A structure model is based on blocks that describe the structure and relevant structural configurations and properties of the developed system. Structure model also describes the decomposition of the system and the parts that make up the system; for example, logical decomposition and physical decomposition. Block definition diagrams define the structure of the system and the logical or physical decomposition. It describes the relationships between different blocks, their associations, generalizations, and dependencies. For example, if the system is a spacecraft and it has various sub-systems, then, it could be decomposed logically, as a thermal sub-system, a structure sub-system, ADCS sub-system or it could be decomposed physically, as solar arrays, instruments, thrusters, and so on. Internal block diagrams establish the relationship

and the ties with the interfaces between the blocks (components) through ports, connectors, and flows. They can have varying different levels of abstraction. Parametric diagrams are a sub-diagram type of the internal block diagram in order to impose mathematical or logical constraints on the interfaces and build up the infrastructure computation in the model. The parametric diagrams can be modeled to define the relationships between the properties of different blocks (e.g., physical laws). The packaging diagram is focused on the organization of the model and displaying the scope of the system.

These diagrams can be cross-connected into what is called "the four pillars of the system," which are based on four basic principles represented by the following diagrams: requirements, behavior, structure, and parametric. It is illustrated in the example shown in Fig. 2-4 [72].

The first cross-connector that joins these diagrams is "Allocation." Behavior diagrams, e.g., activity diagrams or sequence diagrams, can allocate a specific behavioral activity to a structure block and add behavioral constraints. These will be embodied within partitions called "Swimlanes." The partitions represent structural elements, and they have the responsibility of executing the behavior within the partition. The next cross-domain connecting element is the satisfy relationship. This relationship is performed between the requirement diagram and the structure diagram. It is shown on the structure diagram, e.g., block definition diagram, as a call-out notation, which indicates that this particular structural subsystem or element is meant to satisfy a specific requirement. As mentioned previously, the block definition diagram allows adding values, functionalities, and attributes to the block. Value binding property allows to link these values and bound them into a set of equations that are expressed in the parametric diagram and create parameter constraints. Finally, requirements can be verified through interactions. These

interactions would run a test experiment based on the values in the model and be able to conclude if the system's values still satisfy the given requirements.

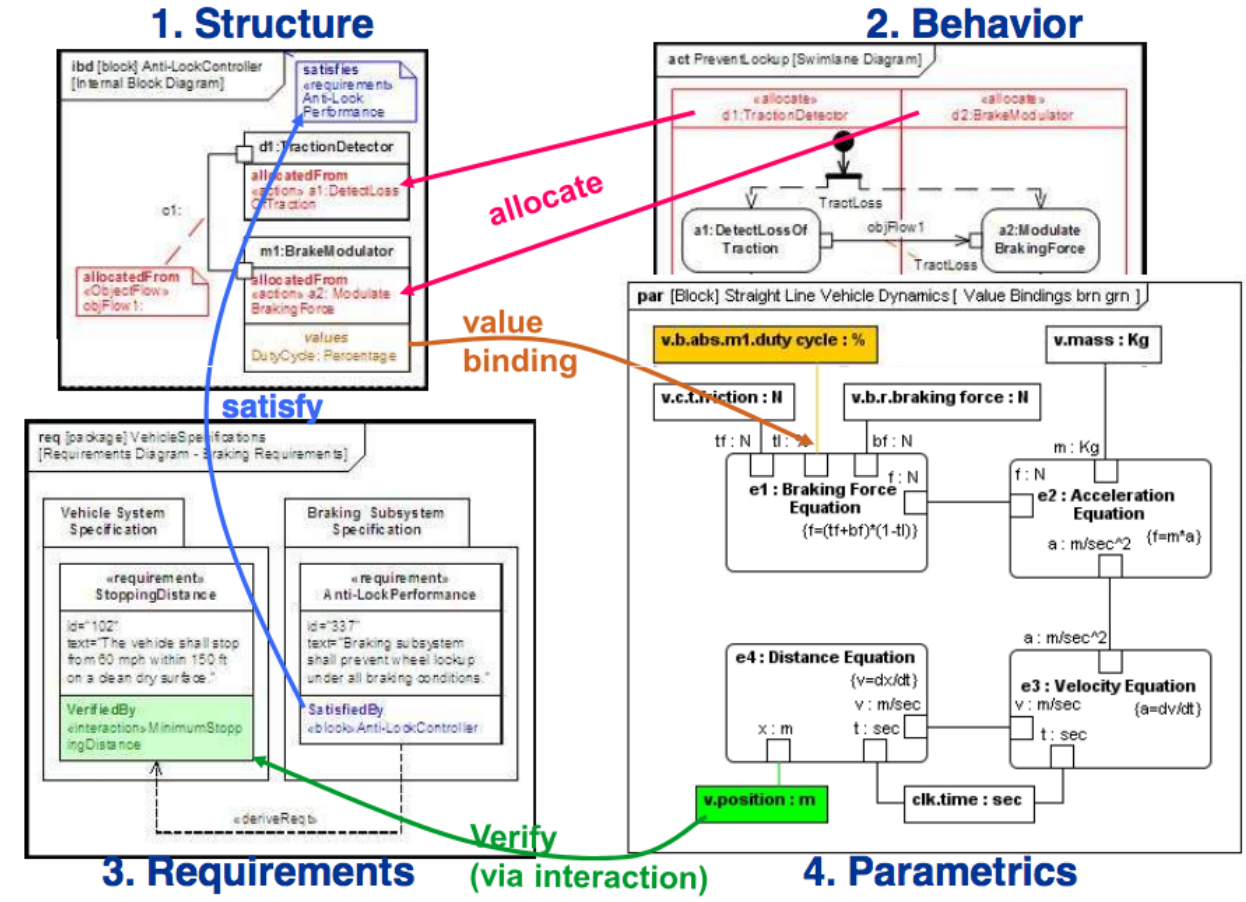


Figure 2-4: Relationships between different SysML diagrams.

Although, these diagrams are the way to define the system and its interfaces within the model, they are not the model themselves. They can create links between diagrams in such a way that if a change takes place in one diagram, it will propagate to the rest of the diagrams that have a connection to the first change. So, it is similar to a database for the whole model that encompasses all the information as opposed to having many isolated block diagrams.

SysML can map the complete design of a complex system, but it is limited to three aspects. A large number of diagrams and constructs are only roughly predefined and do not always allow

a clear or even intuitive use. Also, a large number of constructs can sometimes be used in any manner. The training is therefore done with much effort. The specification of an advanced mechatronic system and its information processing and reasoning, which lead to behavioral changes, is not directly addressed. Concerning the system design focus, functional decomposition is only possible with additional effort. However, the function hierarchy derived in this way is not primarily intended for solution finding.

2.3 Function-Behavior-State (FBS)

Although the terms function, behavior, and structure have been used in the past, it is not until 1990 when they were clarified and used to define a framework for modeling and representing the functionality of a system [73], [74]. In the FBS framework, function represents the functions that the system performs; structure represents the physical elements of the solution, and behavior acts as the relationship between function and structure. In the design synthesis, the behaviour is derived from a function in order to obtain a design solution. When a solution is defined, its behavior is determined to evaluate if it reaches the intended functionality. The FBS framework can also be used as a methodology for the analysis of the design process, through the representation of the evolution of the design state from the analysis of the design procedures [60].

In general, there are two approaches in the FBS design [75]. In the first approach, the functions are related to the behaviors of an element; next, these behaviors are related to the physical-structural descriptions of the elements. It was developed by Gero [60], who proposed the design model Function-Behavior-Structure (Function - Behavior - Structure), and by Umeda et al. [73], who proposed the design model Function-Behavior-State (Function - Behavior - State). This first approach considers behavior as a key concept and determines a clear ontological order: objects have their physical structure. This structure, in interaction with a physical environment, invokes

the behaviors of the objects, and then, the behaviors determine the functions of the objects [76]. In the second approach, the functions of the objects are modeled in terms of inputs and outputs, and these functions are directly related to the physical-structural descriptions of the objects [77]. It is also known as Functional Modeling because it considers behavior as a mathematical representation of the states of objects [26], [77]. Further discussion on FM is given in the next section.

2.3.1 Functional Modeling (FM)

In this dissertation, the term function is used analogous to the concept of function that represents a formulation of the design task on an abstract level. Here, the function is described as a statement that represents the general and intended relationship between input and output quantities of the system, without having to indicate any particular form [37]. A Functional Model (FM) in systems engineering and software engineering is a representation of functions (activities, processes, operations). The purpose of FM is to describe the functions and processes that help to discover the needed information and identify opportunities, and also to establish a basis for determining the product. Product function can be used as a link to the development of new innovative solutions [78]. Since the design process represents the solution to set tasks, the product function is considered the central structuring agent for the abstraction of the design task and the solution.

FM provides a high-level system view specifying the functionality of the product from the product description, where functional modeling is the specification of models that describe the function, and the functional relationships as objects and relations to the development process. Also, functional modeling can be used to describe the procedure in the design process for designing the sub-functions. Through functional product modeling, a solution-independent and abstract representation of a task to be created can be represented [79]. For this reason, function modeling

is considered when modeling in a conceptual level, where functions can be drawn from the realization of different customer requirements [80]. This abstraction of the basic concepts using the product function is used in many areas of engineering and is supported by suitable development tools, especially in electrical, electronic, hydraulic, pneumatic and software developments.

Although there are different definitions of functional modeling, this term can be defined as in system engineering, which states what the system must do, which typically specifies a function or behavior [81]. It is the designer's task to analyze the customer requirements and then incorporate the requirement into the main function. FM represents the link between the human design intention and the designed system. Therefore, there is a coexistence of different FM structures due to the subjectivity of the design's purpose, action, and behavior with external interactions [82].

2.3.2 Knowledge Base and Design Libraries

The design development process requires the provision and linking of different sources of knowledge, especially in the early stages of design development. In the context of the realization of innovative products, the provision and evaluation of related knowledge play a critical role. Effective use of knowledge resources is a key factor that influences design innovation. According to [83], the design knowledge may consist of: (a) an implicit knowledge, obtained through the acquired experience [84], and (b) an explicit knowledge derived from previous methods, design models, strategies or projects.

Design knowledge can support MBSE by incorporating the system knowledge into SysML as libraries [20]. These libraries can significantly support the modeling capability through the integration of the knowledge reuse into the united modeling representations of SysML [85]. Commonly in object-oriented software development, the designer is provided with a collection of libraries of basic software functionalities for reuse [25].

[86] claims that it is faster to adapt existing components or design elements in a new design than to start from scratch, and this will reduce the design time. The goal of design reuse is to make use of existing designs, either by reusing the whole design, which shortens the design time or by changing or updating the design, which takes less time. In addition to reducing the modeling time, design reuse has the potential of significantly saving time and cost savings in downstream assembly, engineering, manufacturing processes, and sales [87]. Existing designs have the advantage that they have already proven themselves in the market and has quality that is accepted by the customer [88]. It is estimated that the performance of the product can be improved by about 20% in this manner [89].

The evaluation and reuse of the mentioned solution modules require a high degree of multidisciplinary and up-to-date knowledge of the most important relationships. Therefore, the reuse of a third-party external knowledge base can be enabled, which is discussed next.

2.3.2.1 The Functional Basis (FB)

Functional Basis (FB) is important because of the need for a consistent approach to accurately represent and connect between abstract functions [90]. FB is a method that is widely used to increase the formality of the design process [91]. Functional modeling is created to reduce ambiguity in the level of modeling of an object, and it requires formal methods. Therefore, FB discriminates between the meaning of functions and flows. In FB, functions are represented by a graphical transformation between the input and the output flows. The flows are divided into three essential types: material, energy, and signal. The primary function is represented by a black box with inputs and outputs, which indicates the flow of the system. The inputs and outputs of a function/sub-function correspond to the boundary interactions. By splitting the overall function into sub-functions and assigning the input and output variables to the sub-functions, a functional

structure is formed by which different levels of abstraction can be represented in a hierarchal manner [92]. Taking into account the cause and effect relationship of individual sub-functions, an assignment of solutions to the respective sub-functions takes place in the subsequent construction steps [92]. Then, a distinction can be made as to whether solution components, which are already present can be used for the realization of a particular function, or first, the function has to be developed further to be captured by solution components [28].

FM contains controlled vocabulary in a design knowledge repository, which consists of 53 functions (in a verb form) and 45 flows (in a noun form). Each function and flow are structured in a three-level hierarchical taxonomy [91]. The most abstract forms are at the highest level, which is called "primary class," e.g., branch, channel, or convert for functions, and energy, material, and signal for flows. The next level is called "secondary class" and contains more detailed functions and flows. For example, separate is a secondary level of the branch, and solid is a secondary level of material. Caldwell et al. [93]–[96] investigated the use of FB and concluded, through an empirical evaluation, that the two levels of taxonomy can provide the most descriptive information. Moreover, free language can be incorporated with FB for a better description of the functional model, especially for the description of flows [97].

2.3.2.2 Simcenter Amesim

Simcenter Amesim [27] is an integrated, scalable simulation platform for mechatronic systems. It allows to virtually assess and optimize the mechatronic system's behavior and performance throughout the development cycle as well as during its usage. The modeling level is most suited for simulating the entire system by describing the power exchange between components, and it is well suited for design in the system level. Amesim includes more than 5000

ready-to-use multi-physics and multi-domain library models for mechanical, electric, thermal, control, hydraulic, pneumatic, and chemical simulation.

This level of modeling is adaptable, so it can cover different physical domains and different applications. The theoretical framework of this system modeling level is bond graph theory [98]. The modeling approach considers the power exchange between components while respecting the balance equations of physics. The modeling approach is based on the connection of elementary models by using power links. This method, which is called a multiport approach, makes it possible to associate different modeling assumptions in the same graphical representation, which affords good representation of the technology [98]. The analytical model has the form of ODEs (Ordinary Differential Equations) or DAEs (Differential Algebraic Equations).

For mechatronics, an integrated system-level performance simulation process is crucial. It is only by applying such a process that the development complexity of a mechatronic product can be handled in a timely, cost-effective and qualitative manner. Existing designs usually contain accurate calculations of the costs and the required design times. By providing up-to-date information about previously used design components, time can be saved and the accuracy of estimating new projects can be increased.

2.4 Simulation Model Transformation

Simulation-driven design can be defined as "*a design process where decisions related to the behavior and the performance of the design in all major phases of the process are significantly supported by computer-based product modeling and simulation*" [99]. In the DDP, different simulation methods are used for the digital validation of products, which can be classified according to their level of fidelity. In the late stages of DDP, for example, embodiment design and detail design, the computational support is widely integrated. For example, Siemens NX [100] and

CATIA [101] are well-known Computer-Aided Design software (CAD) that are used for part modeling and assembly, in which these models are subjected to further analysis. Electrical analysis is done through electronic design automation (EDA), for example by using AGNISYS [102]. It would cause a consistency disagreement between different units because of the design independence [58]. Therefore, data management is needed to govern the information between these different tools. Also, early stages of DDP lack computational support tools [103]. Moreover, systematic methods are needed to translate from a descriptive environment, for example, conceptual design, to a parametric environment, for example, detail design.

FM provides a functional structure that only represents a functional descriptive process of how the product will operate, in which solutions are independent of any physical forms. A major emphasis of the integration with external analysis tools is the integration of an informational and descriptive model with an analytical model and to enable the transformation of information from the descriptive model to the analytical model. SysML does not have built-in analysis capabilities, and it cannot run a model or calculate equations. However, the tools that are implemented in the system can provide those analysis capabilities built into the tools as opposed to the SysML language. For example, in the parametric diagram, a system of equations can be generated, which can then be solved.

The formalization of FBS (as addressed in section 2.3.1) has been established, where functions are transformed into behaviors. Then, with the use of computational support, simulation models are created. These high-level simulation models can roughly specify a system of linked parameters that satisfy the functional requirements. These parameters determine the superior and inferior designs within this range of parameters.

Accurate and detailed simulation models are becoming increasingly complex and time-consuming. An economic simulation is only possible through low fidelity models. Besides, the current practice is to evaluate several different high-level simulation models quickly and perform an early elimination process. At an early stage, a set of equations that describe the system's high-level behavior is established, with the support of model reuse integrated into the design environment, which requires a minimum of analysis [104]. It can facilitate the following aspects: significant evaluation of the design models at an early stage, high-level, cross-architecture, and multi-domain analysis, and consistent transition to the detail design [105].

2.5 Evaluation of Mechatronic Design

Conceptual design solutions must be achieved without violating the design constraints. Engineers encounter many challenges when making design decisions because design constraints might be conflicting.

In [106] Grabisch applied multi-criteria decision making by using a fuzzy integral. He showed that the traditional weighted average technique merely assumes independent criteria and therefore, a well-established methodology was needed to model interactions between criteria. He presented Choquet and Sugeno integrals to overcome this problem. However, a drawback of that approach is the growing number of coefficients. Marichal [107] introduced an axiomatic approach for criteria interaction and aggregation in Choquet integrals. He illustrated the behavioral analysis of aggregation with different interactions by using simple examples. De Silva [108] and [4] proposed a design objective function called Mechatronic Design Quotient or MDQ, which addresses the dynamic interaction between components and subcomponents of a mechatronic system by using separate design indices to express the design criteria for the subcomponents. MDQ is used to achieve optimal performance by aggregating these indices (design criteria) into the MDQ

and optimizing it. In [109] Labreuche and Grabisch addressed the preference of decision making over each criterion and interaction of criteria using a utility function. They introduced two new terms: intra-criteria and inter-criteria information. The difficulty of intra-criteria arises from the need for precise prior knowledge of the aggregation function. In [110] Labreuche addressed the construction of a Choquet integral and the value functions without considering multi-criteria decision-making. He investigated the value and capacity of the evaluation function. The basic idea was, if we were to alter criterion i, while the other criteria are kept fixed, then the weight of criterion j changes when its value is equal to the value of criterion i. This method enables the construction of a sequence of corresponding values on different criteria. Behbahani and de Silva in [111] used the aggregation of Mechatronic Design Quotient (MDQ) to develop a new mechatronic system methodology for the conceptual design stage. They used a nonlinear fuzzy measure through the Choquet integral, which has an advantage over the weighted average, as it takes into account interactions between criteria. To validate their work, they presented a case study of designing a manipulator of an industrial fish-cutting machine called the Iron Butcher. The exponential growth of the number of fuzzy measures was an issue. In [112] Xia et al. extended the work of Behbahani and de Silva where he addressed the problem of the growth of fuzzy measures. The 2nd order Choquet integral was introduced. It allows modeling of the interaction among criteria while remaining operational and straightforward. Later Gao and de Silva applied estimation distribution algorithms for constrained optimization problems [113].

Chapter 3: A Systematic Model-based Process of Conceptual Design Development

This chapter presents the approach of design process modeling for the present development. In the beginning, an overview of the design process methodology is introduced, followed by the details of the design process model. Next, the proposed macro-level design process model is presented, followed by a micro-level design process model. Both methodologies are based on Model-based System Engineering for the support of the integrated development process of mechatronic systems. In the former model, the multi-layer V-model design process is presented, and in the later model, the characteristics of the design tasks and activities within the macro-level design process are given.

Finally, a case study of an industrial fish cutting machine is given to demonstrate the presented design methodology, which the rationalization of the choices of the current solution is investigated.

3.1 Integrated Design Process Methodology

As discussed in Chapter 2, the challenges, which the designers encounter in the design of multi-disciplinary complex systems require integrated design methodologies. However, for an optimal and integrated design development process, the two main types of problems that must be overcome are “process-based problems” and “design-data related problems.”

After analyzing these problems, two aspects of the integrated design approaches are investigated. The first aspect is the “macro-level design process,” which describes the general development process guidelines of the design phases and activities. [10] classifies the “macro-level design process” into macro-level collaboration and macro-level interface. The macro-level

process focuses on the collaboration of different homogenous disciplines and guidance for the design activities. Macro-level interface explores the compatibilities of the components/sub-systems interfaces.

The second aspect of the integrated design approach is the "micro-level design process." This process is categorized into: micro-level collaboration and micro-level interface [10]. Micro-level collaboration examines the communications of the engineers and designers regarding different views of the design. It also includes the data exchange between the engineers and the stakeholders. The micro-level interface allows engineers from different disciplines to share information or data during the design process, through formal or informal interactions.

The present work utilizes the System Engineering (SE) approach to support the integrated design of complex mechatronic systems. A combination of approaches is used for the modeling of the "Macro-level design process." The first approach employs the V-model [48] as a comprehensive process model to guide and direct the general structure of the design process, and give a holistic view of the process. The second approach exercises the systematic design approach presented by Phal and Beitz [37] to represent the level of detail of the development process. Therefore, the "macro-level process" here is based on the V-model, but its phases are amplified based on the systematic design of Phal and Beitz to better manage the complexity. It is useful in managing the issues of the macro-level collaboration and macro-level interface, where the combination of the two methods provides a faster response to the dynamic changes of the design process, and the integration capabilities are enhanced. For the development of the "micro-level design process," SysML is utilized so as to represent all the aspects/activities of the system. Also, it describes the links, connections, and interfaces of the system. Therefore, it facilitates the

collaboration and coordination between different engineers. Moreover, it provides a common language, which enables traceability between the various components/sub-systems and functions.

In the next section the developed integrated design development process is introduced.

3.1.1 Macro-level Design Process

The macro-level DDP model that is developed in the present work reconciles the general structure of the V-model as described in VDI guideline 2206 [53], [54]. It represents a general flow of the activities of the design process of mechatronic systems, complementary to which are the existing guidelines, especially VDI 2221 [48], [49], and VDI 2422 [50]. The main reasons for selecting the V-model from the domain of software engineering for use in mechatronic systems are the following:

- The V-shape strictly enforces the top-down approach (system design) and the bottom-up approach (system integration).
- The presence of permanent verification/validation of the requirements and functions on the left side is essential during the system integration on the right side.
- It has usage practicality and acceptance when industries use it for the design of mechatronic products and systems [54].

The VDI guidelines establish a parallel design for each engineering discipline, and then the discipline-specific sub-systems are integrated into the overall system. Some issues arise as a result of the integration of the discipline-specific sub-systems/components. They include the lack of communication of data during the parallel development process [58]. Moreover, the general structure of the V-model suffers from the structure rigidity, which makes it difficult to implement any changes mid-way. Therefore, the present dissertation proposes an additional extension to the existing V-model. Specifically, it develops and adds more organization and transparency to the

DDP, increases its flexibility, reduces the design search space, reduces the number of design iterations, and supports sub-system integration [114].

Macro-cycles According to the Degree of Details

The macro-level DDP of the V-model may be defined as generic procedure steps for use as a management tool. The model provides an overview of the project cycle and the way to relate different development activities at several phases with the corresponding integration development activities throughout the system life-cycle. It is necessary for complex mechatronic systems to go through several cycles within the macro-level in a systematic manner in order to properly prevent design defects and accurately consider the customer needs during the development process. Therefore, the present work exploits the VDI 2206 for the macro-level and combines it with the design methodology by Pahl and Beitz [37] to represent the degree of detail. The design methodology of Pahl and Beitz is preferred because of its level of detail and its use in product design [115]. This combination increases the systemization and organization of the development process. It elevates the validation/verification capability, amplifies the design cycles, and responds more appropriately and accurately to rapid changes of the customer and market requirements.

Each cycle of the V-model represents a stage of the design process, as described by Pahl and Beitz. Fig. 3-1 shows the proposed V-model with different design cycle phases, which represent the level of detail.

The process model results from the combination of the V-model 2206 guidelines to symbolize the general structure. The Pahl and Beitz design methodology describes each cycle of the V's. Specifically, the breakdown of the layers is divided into four main phases: product planning, conceptual design, embodiment design, and detail design. The design process begins with the identification of the requirements. The design process activities then start with the inner

V-model, going down from the left-hand side and then up from the right-hand side. If the output of the inner V-model meets the corresponding requirements, then it goes to the next phase "Conceptual Design." Otherwise, an iteration process would be carried out until the output of the inner cycle meets the corresponding requirements. The verification and validation test enforced in each phase will significantly reduce the number of iterations in the later phase. Therefore, the overall development costs and the time-to-market can be greatly reduced. This cycle is repeated throughout all the sub-V-models until the final product is developed through the outer V-model.

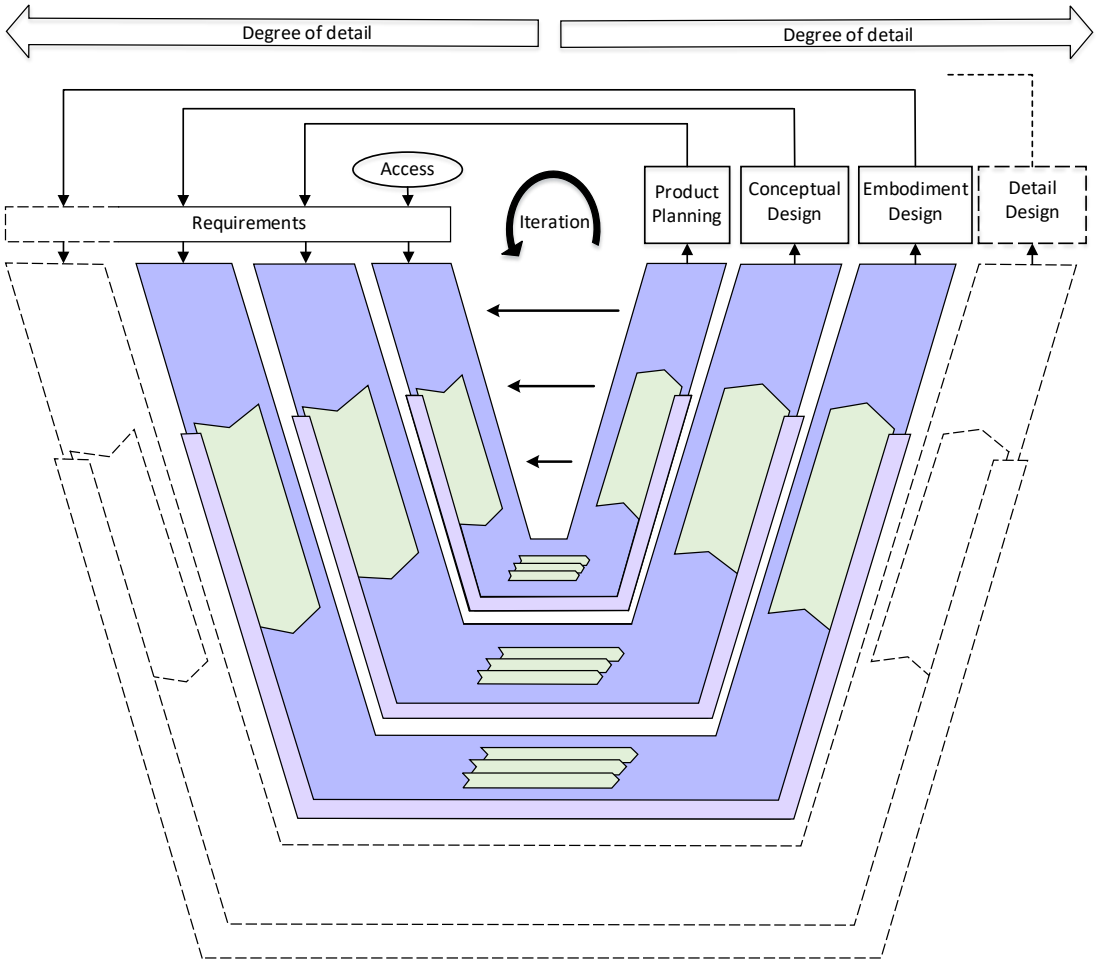


Figure 3-1: The proposed conceptual macro-level process.

Throughout the first cycle, the designed mechatronic product undergoes a market analysis to help find and select product ideas. The intended functionality and specifications are then defined. These specifications and functions are elaborated in detail in the second cycle, describing the product functionality, in order to identify solutions that fulfill the required functions. These solutions are then used to develop working structures for the product. In particular, combinations of solutions are developed into principal solution variants, which are evaluated before they undergo the third cycle. The form and layout are identified in the third cycle. Auxiliary functions are also defined, and design solutions that satisfy these functions are found. Then, detailed and compatible layouts for the primary and auxiliary functions are developed. This cycle ends with the evaluation and optimization of the design. In the final cycle, assembly drawings are developed and the production documentation is completed.

One advantage of this model structure is that each stage is subjected to rigorous tests of compatibility, verification, and validation. Furthermore, the model is adaptable to dynamic changes of the customer requirements, which resolve the problem of rigidity of the V-model. Additionally, the granularity of the requirements can be incorporated into the proposed model quite appropriately, as shown in Fig. 3-1, since the reality is that customers provide more accurate and detailed product requirements as the design process progresses. Moreover, the proposed enhancement of the V-model is developed based on a typical and accepted system engineering approach that is used by industries in the context of mechatronic systems.

The V-model and other tools and methods that are used for the support of the design development of a mechatronic system [116] primarily addresses analysis, and there is no specific support for the synthesis of the device components [115]. Hence, combining the VDI 2206

guidelines with a well-accepted design methodology such as that by Pahl and Beitz is desirable for extending the industrial application of the V-model methodology.

The next sub-section presents the macro-level process cycle of the proposed multi-layer V-model, for the conceptual design process.

3.1.2 Conceptual Macro-level Process

The details of the conceptual macro-level process cycle, which describes the procedure patterns of the conceptual design phase, is discussed in this section. Researches in the automotive industry [117] indicate that there exist issues related to the integration of the discipline-specific sub-systems. These issues result from addressing the integration of the sub-system in the later stages of DDP. As a consequence, the design process becomes inefficient with respect to cost and time. Also, the information provided from the domain-specific partial solution does not optimally satisfy system integration.

In an attempt to improve the integration of different disciplines associated with a mechatronic product, the proposed conceptual macro-level process cycle incorporates the virtual integration and simulation in an early phase of the design process – the conceptual design phase. Therefore, it enhances the capabilities of concept analysis and evaluation.

The early concept design process of a product involves three stages: a general descriptive stage, a virtual simulation stage, and an extensive analysis and evaluation stage. The enhanced V-model focusing on the conceptual design process cycle comprises three segments: the left-hand side of the V-model represents “Concept design and modeling,” the bottom of the V-model portrays “Run-time concept integration and simulation,” and the right-hand side of the V-model depicts “Concept analysis and evaluation.” The proposed conceptual macro-level process cycle of the V-model is shown in Fig. 3-2.

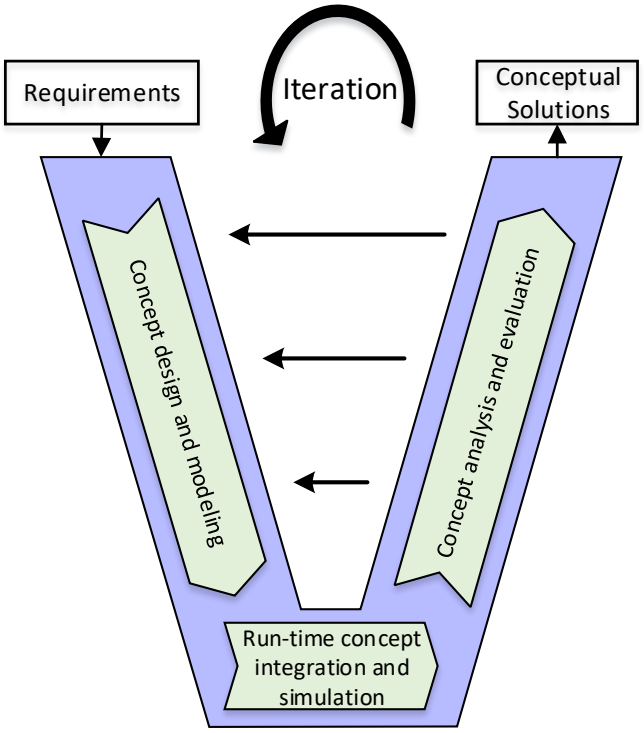


Figure 3-2: The proposed conceptual micro-level process.

3.1.2.1 General Descriptive Sub-process Phase

In the general descriptive sub-process phase, the concept design and modeling are conducted, where a description of domain-spanning solution concepts is defined. It utilizes the model-based design approach for the development of a CIM, in which functional, physical, logical, and structural characteristics are represented as different views of the designed product. The description of this sub-process phase uses flexible graphical modeling languages in order to thoroughly describe the solution concepts. This technique supports the integration of different disciplines in the domain-specific environment [118], by providing constructive information, i.e., multidisciplinary interface information. Other techniques include customizing the system into structured modules in order to reduce complexity [119], and early modeling and concept simulation [120].

Different phases of the concept design and modeling sub-process are modeled and described using schemes diagrams and semantics that include developed requirements, functions, or generic structures, each of which is modeled with a different graphical language. The product description models enhance clarity, since they should be understandable by all persons involved in the modeling process. The challenges arising from the links between different views of the product models are addressed, and relationships between different views are developed.

CIM contains all the information necessary to describe a product from different viewpoints, which can be stored, accessed and reused more conveniently and can replace paper-based methods. However, the models arising at the description level cannot be interpreted and executed in their entirety by the computer.

3.1.2.2 Concept Integration and Simulation Sub-process Phase

The second sub-process phase of the conceptual macro-level process comprises virtual integration and simulation. Those activities provide a consistency check for the system model, which is developed in an object-oriented or component/sub-system-oriented manner. For example, the physical laws of the system must be complete and consistent, and they are not only stored but also executed in the model.

The degree of detail is more abstract, from which generalized components can be constructed and simulated, for example using a component library. These simulation models are used to gain insight into the first set of conceptual ideas, which describe the product components and their behavior and provide a high-quality content. Also, they provide further understanding of the system. For example, some parameters need to be set, which might be neglected during the sub-process of concept design and modeling development.

The boundary between descriptive, function-oriented models and simulation models is abstract, and it is challenging to manage the virtual integration of them. Direct mapping to link the two sub-process models would be very complex due to the scope of the simulation models. The transformation of a concept model into a concept simulation model would require connecting the relationships between various components that are used, in order to complete the virtual integration. Then, a system of equations which constrain the properties of components has to be created to assure product properties, since these components carry the relevant domain information. These equations and constraints can roughly determine the set of parameter values that are required to satisfy the functional requirements. These values can be an upper limit, a lower limit or a range of acceptable values.

Additionally, the design sub-process phase enables the development of control system strategies in parallel with the development of the integration process.

3.1.2.3 Extensive Evaluation Sub-process Phase

The third sub-process phase constitutes the development of an extensive and comprehensive set of evaluation techniques. Generation and evaluation of a design solution are two tightly interconnected phases of the conceptual design. Effective evaluation of possible conceptual choices is the key condition in the conceptual design stage.

Design evaluation assesses how well the design solutions will function or behave against a set of non-functional requirements. In this context, non-functional requirements may include criteria, attributes, and constraints, for example, non-parametric constraints. Determining the non-functional requirements is done after reviewing the design tasks and customer needs and performing the requirement analysis, which is done during the first inner cycle of the V-model – product planning. The selected attributes and sub-attributes have to be operational in order to

identify how well each conceptual design solution meets the design requirements expressed by the attributes and sub-attributes.

The multi-criteria design evaluation index MDQ, which stands for Mechatronic Design Quotient [29], is used in the present work to evaluate conceptual design solutions. Cost, weight, quality, and flexibility are examples of non-functional requirements. Once the non-functional requirements are obtained, their relative importance is determined subjectively for the evaluation of the conceptual design solutions. The Analytical Hierarchy Process (AHP) and the pairwise comparison can be utilized to ensure the consistency of the weights. AHP is a Multi-criteria Decision Making (MCDM) technique that is used to assist the decision-makers in solving complex problems as well as handling multiple conflicting and subjective criteria.

Product development processes generally have an iterative character, since problem-solving initially requires a great deal of information, which is gradually reduced by repeating certain steps several times. The results of the conceptual design macro-level process cycle are solution concepts, which are analyzed and evaluated against a set of customer/non-functional requirements. The intended functions also have to be fulfilled. To help with the subsequent design cycle – the embodiment macro-level process cycle, requires results at a lower level of abstraction, and thus a network of specified parameters is identified. Therefore, a fewer number of iteration cycles are needed and consequently the design time-to-market, cost, and resources are reduced.

The micro-level process involves the development of the design model activities in the concept design and modeling sub-process and the relationships between them. Chapter 4 gives more details of the micro-level process model.

3.2 Case Study System

The reconfiguration and the design of the Iron Butcher (IB)—an automated fish cutting machine, is used in the present investigation as a case study to evaluate the validity of the proposed methodologies. This case-study system is described in the present section. The fish cutting machine has been developed in the industrial automation lab (IAL) of the University of British Columbia. The machine automatically cuts the head of fish accurately while minimizing the manual operation and the wastage of fish meat [121]. The Iron Butcher has many functions and sub-systems, including: an electromechanical conveying system, a hydraulic system, and a pneumatic system, as shown in Fig. 3-3.

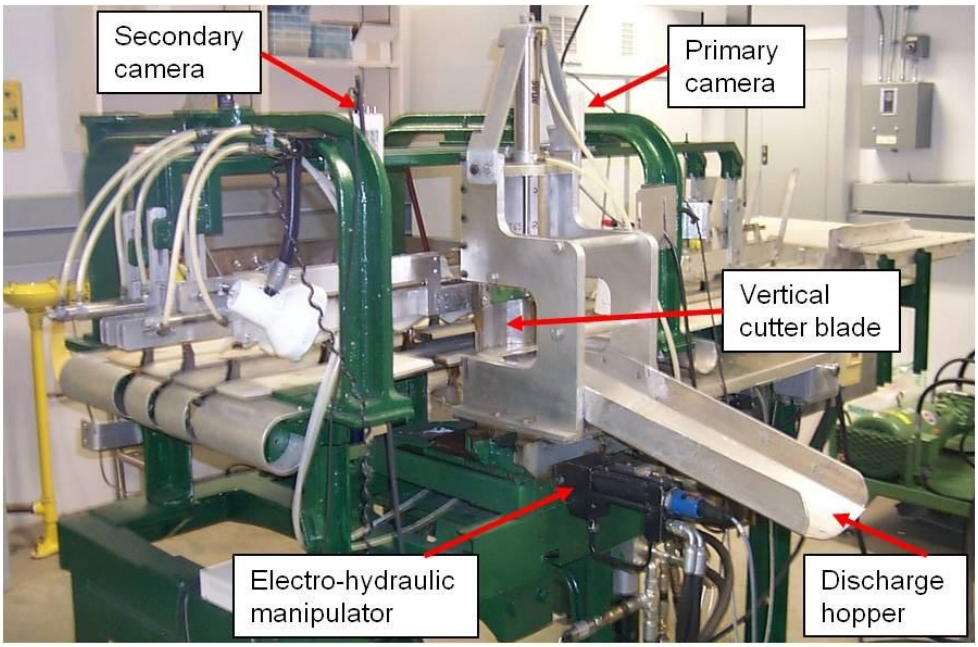


Figure 3-3: Iron butcher.

The electro-mechanical conveyor system falls into the category of a mechatronic system. In the present case study, this system, which is used to transport fish from the feeding station to the cutting station, is considered. Conveyor systems are widely used in fish processing machines in order to provide intermittent motion for the fish during transportation, inspection, and

processing. A 3-phase AC induction motor (Fig. 3-4 (a)) is connected to a variable-diameter pulley (VDP) to compensate for any speed variations of the motor. The VDP is connected to a gearbox to increase the torque and decrease the velocity. A mechanical linkage is used to convert the rotational motion to the translational motion of the conveyor, thereby moving its sliding mechanism horizontally. The intermittent motion profile is planned in such a way that the cutter has sufficient time to cut the head of each fish with minimum meat wastage, when the fish is stationary. Furthermore, the fish has to be held firmly during the transportation and cutting operations. It is done through several fixtures (fingers) that fold or stay open according to the cycle of motion (Fig. 3-4 (b)).

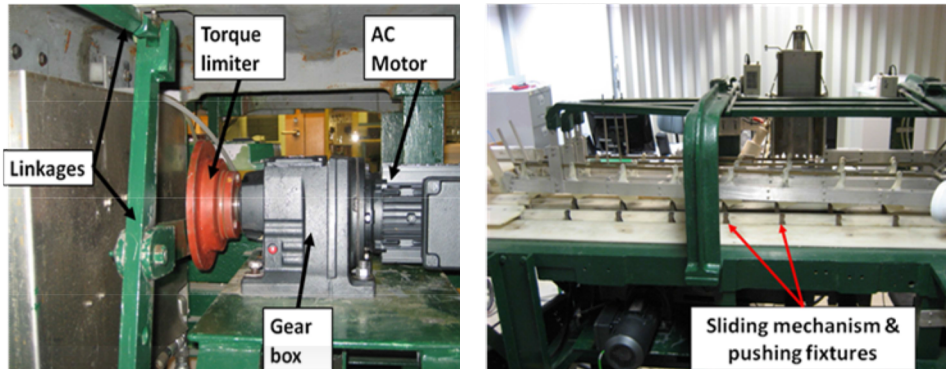


Figure 3-4: Conveyor system; (a) AC induction motor (left), (b) Sliding mechanism (right)

The motion cycle of the conveyor system has a continuous motion segment and a dwell (stationary period). Specifically, there is an acceleration from rest and deceleration to rest (and possible constant-speed motion in between these two segments). There are many acceleration profiles such as constant, trapezoidal, and sinusoidal. Sinusoidal acceleration/deceleration profile is commonly used for indexing systems because it provides minimal jerk (rate of change of acceleration) as shown in Fig. 3-5. The cost is mostly attributed to the drive system and the speed controller.

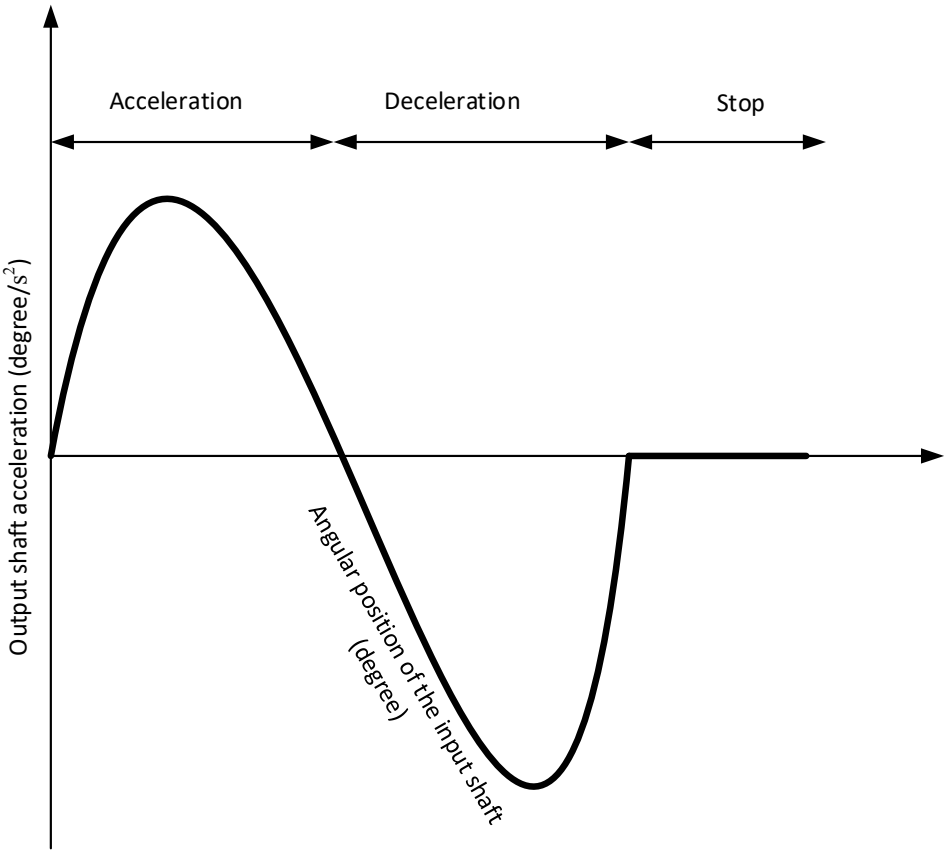


Figure 3-5: The acceleration profile of the conveyor system.

Chapter 4: Micro-level Process Model

The details of the micro-level process model are presented in this chapter. It defines the underlying organization of the different design activities in the phase of concept and modeling sub-process. Moreover, the relationships between the design activities are described. The design activities considered here incorporate specific model information that is derived from an overall model, to cover different aspects of the system. From the model-based design point of view, various design phases throughout the life-cycle of an engineering system need models with relevant objectives and details [122]. In the conceptual design phase, this model information is more general and abstract, which considers different characteristics of the design process, such as hierarchical decomposition and modularity.

An exchange of process models takes place between the macro-level process and the micro-level process, in order to maintain the relationships between different activities systematically and efficiently.

4.1 Concept Design and Modeling Sub-process Phase

The design activities that take place in the left wing of the V of the conceptual design will refine the task of conceptual modeling. The aim here is to generate innovative conceptual design solutions that satisfy the customer's functional and non-functional requirements. These activities can be represented as requirement modeling, functional modeling, and structural modeling, as shown in Fig. 4-1. Each of these represents a specific aspect of the stage of conceptual modeling and design. The next section describes these activities.

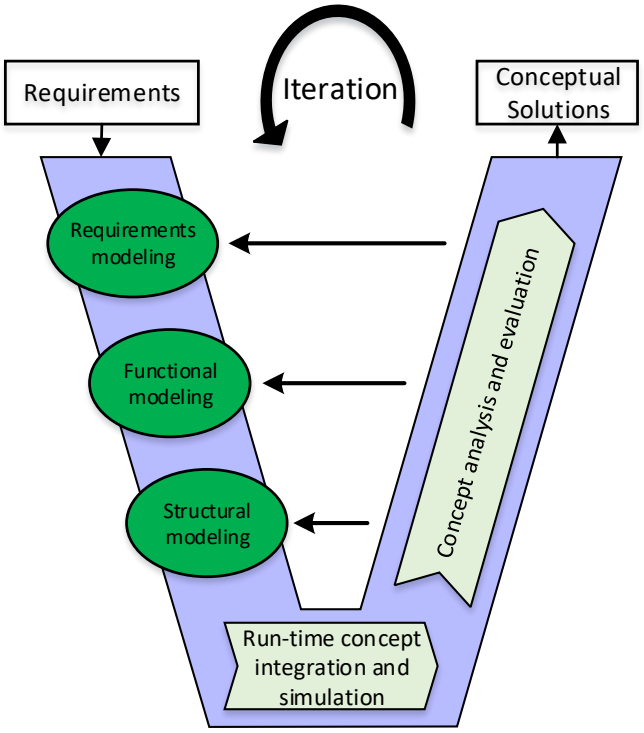


Figure 4-1: Details of the phase of concept design and modeling sub-process.

4.1.1 Requirement Modeling

For a mechatronic product, the technical and economical product requirements play a decisive role since they define the goal of the development task. Therefore, it is necessary to taken into account the prior-defined requirements for the product.

The requirements define the desired properties of the product to be developed, and they limit the available solution space to only what meets the requirements. The product task must be specified, in which the intended requirements are described in general terms, i.e., without anticipating specific future solutions. Also, they must be formulated in a solution-neutral manner. System modeling should ensure efficient management of the requirements to guarantee their fulfillment. Utilizing a modeling tool for requirement modeling is generally preferred. It can help in incorporating this information in the product model, which provides a formalized knowledge

base at this stage [43]. The first requirement can be derived from the overall description of the product task.

Requirements can be initiated both externally and internally, for example, by stakeholders and engineers, during the development process. However, usually, requirements do not contain all the required information at the beginning of the design. So, a continuous process to obtain information and coordination with the client is necessary. For example, detailed requirements may not be available at the conceptual design phase, and designers may obtain the needed requirements at another more detailed design phase.

The initial requirements will lead to more detailed requirements, and these requirements can be further decomposed to create a hierarchy of requirements. Every requirement must be properly clarified at the beginning so that additions and changes would be reduced to ae bare minimum, in order to save time, capacities and costs. Depending on the level of detail, the requirements may be applicable to the overall system, the sub-systems/components, or the interconnections between the subsystems/components.

Requirements can be categorized into four groups: Global requirements, Cumulative requirements, Specific requirements, and Interconnected requirements. They are clarified below.

1- Global requirements are those that must be applied to every single sub-system/component; for example, lead must not be used in any sub-system/component of the system.
2- Cumulative requirements are those that account for the degree of participation of various sub-systems/components in the overall system; for example, the overall cost of the system must be less than the sum of separate costs of the individual sub-systems/components.
3- Specific requirements are the requirements that are relevant to a particular sub-system/component; for example, electromagnetic pulse protection (or shielding)

requirement is relevant to the electrical sub-system, or lubrication requirement is relevant to the moving parts in the mechanical sub-system.

4- Interconnected requirements are those that are influenced by other requirements. For example, the energy consumption requirement, which is a measure of the system performance, is related to the dynamic properties of the system; for example, the maximum mass and the required driving force. The mass and the required force requirements can be determined independently. Therefore, through the physical relationships between the mass, the required force, and the energy consumption, the system performance requirement can be determined [122].

A hierarchical structure at different levels of detail should be established for requirement modeling. Consequently, the functional model can be constructed following the hierarchical structure of the requirement model. System Modeling Language (SysML) can be utilized for the modeling of the hierarchical Requirement structure, and for implementation of the relationships.

4.1.2 Functional Modeling

A function is a statement of the task or role of the system, which shows the relationship between the available input and the desired output, without having to indicate any particular form [32]. It represents the functions that the system performs. It also provides a formulation of the design task on an abstract level. It typically specifies a function or behavior; for example, water heating or energy conversion. Starting from the list of requirements, abstracting must formulate the essential problem and derive the overall function. This concept is adopted in the present dissertation, from past work [32], [35], [123], [124]. In most cases the overall function is very complex and must, therefore, be subdivided into sub-function, which creates a function structure.

A Functional Model (FM) is a representation of functions (activities, processes, operations). The modeling of the functions of the objects is followed in the present work according to [125], where the function description is in terms of inputs and outputs, and the functions are then directly related to the physical structure. The purpose of FM is to describe the functions so as to facilitate discovery of the needed information to help identify opportunities and to establish a basis for determining the product. Product function can be used as a link to the development of new innovative solutions [78]. Two aspects can determine the degree of functional integration for the formation of a product function: first, with regard to the functional structure, and second, with regard to the number of functions that are realized. The degree of functional integration improves the customer benefits, provided that the realization of the product, and at the same time, the given requirements are met. In products with a high degree of functional integration, there exists a great potential for creative innovation.

Functional structures are considered as graphical representation of functional models. Usually, a given product does not have a unique structural representation, which is one of the frequent criticism of the functional model even with the support of a systematic methodology [122]. As an attempt to help overcome this problem, a hierarchical structure for the functional model of the product under consideration is used. [126] developed a computer-based support for the concept development phase to facilitate the data management aspects of the product models. SysML was utilized to model the different types of ports of the functional model. [92] proposes a functional decomposition technique, which is based on the function interfaces. Moreover, the decomposition of the hierarchical structure must be derived in order to fulfill the desired requirements and help serve the intended design task. Therefore, proper relationships between requirements and functions must be established. Despite that different modeling methods have

been proposed to help engineers/designers in the modeling of the functional structure, for example, APTE, and IDEF. However, these methods are not suitable for representation of the hierarchical structure [127].

In the present work, the hierarchical functional structure of the functional model of the designed product is developed using the system modeling language SysML. More details are found in Chapter 5.

4.1.3 Structural Modeling

After performing functional modeling and decomposing the functional structure, the designer should come up with conceptual design solutions. First, a component, a set of components/sub-system, or sub-systems that represent instantiations of the physical forms or objects that satisfy each sub-function, should be identified [28]. These form solutions are then combined within the solution environment to form the complete structure of the final form of the product [128] - conceptual design solutions. A large number of conceptual design solutions are generated in this manner, which will be evaluated against the relevant requirements. A significant challenge in the conceptual design phase is the complexity of obtaining the sub-systems in the presence of incomplete information, such as the parametric values. For example, a sub-function "Convert Electrical Energy to Mechanical Energy" does not indicate a parameter value, size, or shape. Electric motors are sub-system instantiations that can describe the behavior of a sub-function. The transformation from the functional structure to the sub-systems requires engineering expertise, experience and judgment; however, the transformation may not be one to one. Several sub-systems or physical component instantiations may satisfy a specific function. So, depending on the case and the function type, one function may need the satisfaction of more than one sub-system. On the other hand, one sub-system might satisfy more than one function.

Any major component type should be as general as possible but sufficiently specific to allow the user to build a clear abstraction of the component that can be used during the conceptual design. [127] suggests that the structural model should exhibit the decomposed sub-functions, so that the consistency can be maintained. Therefore, a complete structural model is achieved by decomposing the sub-systems, in which the decomposition process is applied recursively.

Now, we encounter such questions as: how can we appropriately construct a structural model hierarchy, and at what level of detail? At first, the sub-system should be as general as possible, and at the same time, sufficiently specific to provide a proper yet abstract definition of what function the sub-system accomplishes. Then, the decomposition process is continued until two conditions are met: Completeness and Exclusivity [28]. Completeness ensures that a set of decomposed components completely provide all the necessary parts of the parent sub-system. Exclusivity indicates the independence of the functionality of the sub-systems. Based on these criteria, the lowest level of the structural decomposition can be obtained by the standard components from handbooks, past design solutions [127], or simulation components, as presented in Chapter 6.

4.2 Realization of Consistency between Macro- and Micro-level Processes

Design development of a system is based on an interplay between the conceptual macro-, and micro-level processes of the system. Generally, in the beginning, the design process has a high level of uncertainty because the most significant properties of the principal design solutions are unknown [122]. The macro-level process manages the development of the design and gives an overview of the design activities, where the multi-layer V-model represents the development process. The type of macro-level development process has a significant impact on the design

activities of the micro-level process [122]. Therefore, these two processes have to be developed simultaneously.

The micro-level process utilizes a model-based design approach for the development of the CIM, in which requirements, functions, and structures are represented as different model views of the designed product. CIM contains all the information that is necessary to describe a product from different viewpoints, where the information is stored, accessed and reused more conveniently. However, the exchange of information between different views of the model is essential in order to allow an adequate transition from system models to domain-specific models. This transition of information requires a collaboration between different model views of the product, described by the CIM, with the process model represented by CDDP. This exchange of information between the two processes is shown in Fig. 4-2.

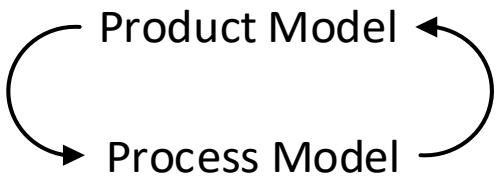


Figure 4-2: The development of the design process.

According to [129], two types of integrations are relevant in the design of mechatronic systems. An integration that is related to the process and the product, which needs a high-level collaboration between engineers and also among different design disciplines [129]. It is directly related to process-based problems – section 1.2.1. The present work contributes to the development of a design process that facilitates the involvement of various engineers. However, since the present focus is on the conceptual design phase, only the relationships between different model views of the CIM of the product are considered.

On the other hand, the integration associated with the product will require functional and physical integration. The present work presents an interface model that enables the exchange of information between different model views, which addresses the design data-related problems – see section 1.2.2.

The next section describes the development of the transitions between the macro- and micro-level processes.

4.2.1 Transition from Macro-level Process to Micro-level Process

The macro-level process is utilized to organize the exchange of information between different views of the CIM. Also, a hierarchical structure is established for each view of the model to maintain the consistency of the information, and to allow the exchange of information at different levels of abstraction. This process also determines the model granularity, which describes the level of detail that a model has been subdivided into. This entire process would enable an efficient evaluation of the transition from the customer model perspective to the engineer model perspective of CIM, which facilitates the progression from the conceptual design phase to the detail design phase.

This process of information exchange between different CIM views is represented in the macro-level process model in two ways. First, the functional model hierarchy should correspond to the requirement model hierarchy, for requirement tracking. Second, the structural model hierarchy should correspond to both requirement model hierarchy and functional model hierarchy for requirement and functional tracking, as shown in Fig. 4-3.

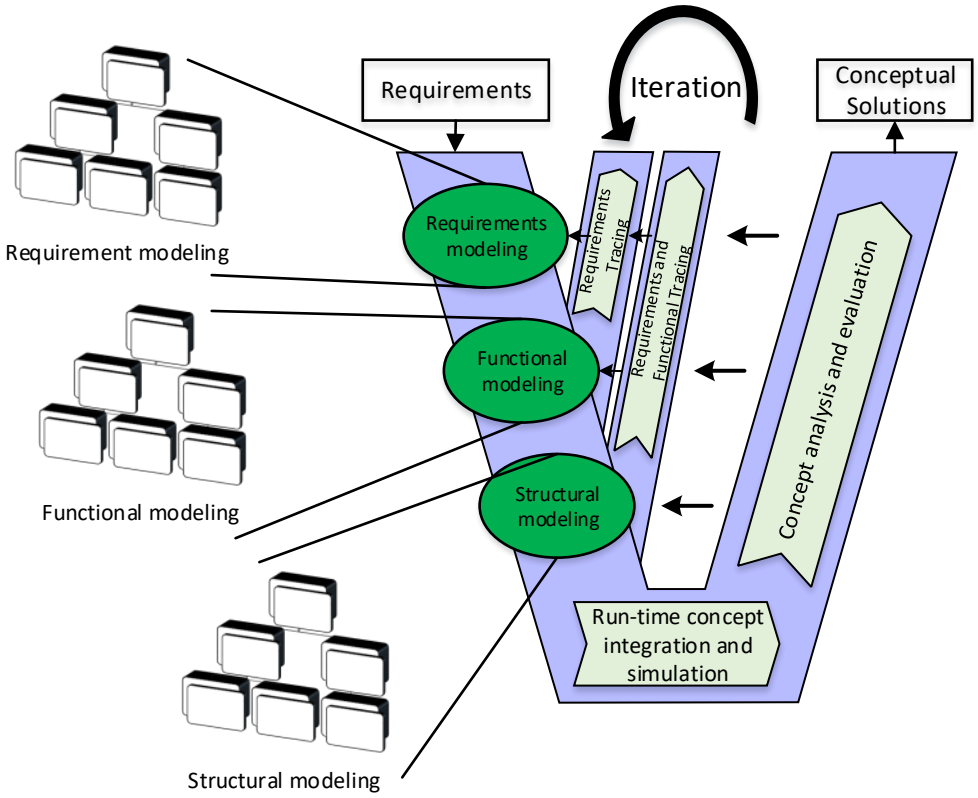


Figure 4-3: Information exchange between the product model and the process model.

This arrangement supports the exchange of information by enabling traceability between different model views, which can be visually depicted. For example, the functional model is extended to form a V-shape so it can exchange information with the requirement model. In the same manner, the structural model is extended to exchange information with both the functional model and the requirement model. Also, it facilitates the functional integration and the structural integration.

However, the exchange of information might not be one to one, or they must have the same level of abstraction. For example, the cost requirement from one abstraction level of the requirement model might encapsulate components at multiple levels of detail. Also, each model view is allowed to generate multiple levels of abstraction without the need to exchange information

with other model views. It can be concluded that different model views would make the exchange of information challenging and difficult. Therefore, an *Interconnection Model* is developed for the three model views. They are discussed in detail in the following section.

4.2.2 Transition from Macro-level Process to Micro-level Process

The micro-level process is developed to model different types of information. However, a common language for interconnections is required for effective communication between different model views. Therefore, the present dissertation develops an *Interconnection model* to facilitate modeling the process activities within and between different model views of the CIM. It also advances the synchronization between macro- and micro-level processes. Besides, customers and engineers from different domains benefit from the model by getting insight on the information activities. The Unified Modeling Language (UML) is used to graphically represent the process activities through *Classes*.

An interconnection is defined by two interfaces, and the interconnection starts from one interface and ends at the other interface. Interfaces are regarded as one of the most powerful tools in system management [130]. In mechatronic systems, interconnections are used to describe the interactions between the internal components of the model view, i.e., a function with another function, or components between different model views, i.e., a requirement with a function. Therefore, in order to represent the information with a shared language, interconnection classifications are presented now.

4.2.3 Interconnection Classifications

The interconnection classifications are: *Allocation*, *Type*, *Class*, *Conversion*, and *Confidence*. These classifications support the macro-level and micro-level processes simultaneously by providing standard representations of the interconnections throughout the

conceptual design and modeling sub-process. In addition, they enable easier collaboration among engineers from different disciplines.

4.2.3.1 "*Allocation*" Interconnection

An allocation interconnection indicates where the interconnection modeling takes place. There are two types of allocation interconnection:

1. *Internal* interconnection: It describes the interrelationships between different elements within a model view in CIM, i.e., the connection of a requirement element to another requirement element.
2. *External* interconnection: This interconnection occurs between the elements of different model views of the CIM and within the CDDP, i.e., functional elements and structural elements.

4.2.3.2 "*Type*" Interconnection

This interconnection classification is categorized as an *internal allocation* interconnection. It expresses where within different model views of the CIM, the interconnection modeling activities occur. There are three *internal types* of interconnection: *Requirement inner* interconnection, *Functional inner* interconnection, and *Structural inner* interconnection.

4.2.3.3 "*Class*" Interconnection

The *Class* interconnection shows which type of information is transferred through each *type inner interconnection*. Therefore, different *class* interconnection models are proposed for the three *type inner* interconnections. They are described as follows:

4.2.3.3.1 "*Class*" Interconnection for *Requirement Inner* Interconnection

The classifications of interconnections describe the information exchange in the requirement model. They assist proper handling of the requirements and the relationships between them. The interconnection *classes* in the requirement modeling activity are listed below.

1- *Containment* interconnections: These express the relationships of parent requirements with two or more child requirements. They help in managing the decomposition process of the requirements until all the child requirements are satisfied with a corresponding function or a group of functions. As a result, the parent requirements are satisfied as well.

2- *Derive* interconnections: These interconnections correspond to the requirements at the next level of the requirement hierarchy, or the same level of the requirement hierarchy but at a different level of abstraction. The relationships developed by these interconnections are assigned between two requirements generated by different engineers from different disciplines. For example, requirements developed by domain-specific engineers are connected through the derive interconnections to the requirements developed by system engineers. This interconnection incorporates additional considerations of detailed implementation.

4.2.3.3.2 "*Class*" Interconnection for *Functional Inner* Interconnection

According to [26], [91], different flows in the functional model are prescribed by the functional Basis (FB), in which the flows are divided into material, energy, and signal. They represent the essential information that is exchanged between different functions. Therefore, a proper interconnection should govern these different types of flows. Three *classes* of the interconnection in the functional modeling activity are:

1. *Material* interconnections
2. *Energy* interconnections
3. *Signal* interconnections

These interconnections maintain the flows of the FB and enforce the consistency rules. For example, a mechanical energy interconnection has to have similar beginning and ending ports in order for the interconnection to take place.

4.2.3.3.3 "*Class*" Interconnection for *Structural Inner* Interconnection

In structural modeling, graph-based modeling tools such as bond graphs and linear graphs, are used due to their structural nature and the characteristics of an object-oriented language. They use a declarative language, in which objects contain equations and, therefore, the structural model is defined by its set of equations. The objects are interconnected in order to establish relations between the states, using state variables. These interconnections can take the form of *classes* of:

1. *Power* interconnection: It represents the flow of power (i.e., rate of change of energy) between the structural elements. It uses the power variable, which is the product of the two variables: *effort* e and *flow* f. Using a unified language, *Power* interconnections enable energy flow between different engineering domains (e.g., mechanical domain, electrical domain, thermal domain, fluid domain).
2. *Control* interconnection: It represents a signal connection, which requires much less power than a *Power* interconnection, because it transmits communication signals. It transmits a communication signal, from one structural element to another element, providing information for control, drive, and so on.

4.2.3.4 "*Conversion*" Interconnection

The other category of interconnections is the *external allocation* interconnection. They describe how the model view is converted into another view within the CDDP. Since the source and the target of a flow through interconnections are not defined, the directions of the interconnections are important to determine the source and target of the flow between the different model views within the CDDP. These interconnections are directional because they are assigned according to their operational directions.

1. R to F: The first *conversion* class concerns interconnections between requirements and functions. These relationships describe which requirements are fulfilled by which function or group of functions. For example, a sub-function "Measure breakdown voltage" may be related to the "Security requirement" in order to resolve a breakdown condition. As another example, a sub-function "Provide a translational/rotational motion" may be related to the "maintenance requirement" to satisfy the usability condition. These relationships help keep track of the requirements throughout the design process.
2. F to S: This *conversion* interconnection is used to describe how a function or a set of functions are related to structural components, which are simulation components and are described in detail in Chapter 6. The basis of this interconnection is port-modeling of energy and signal flows of the functional model, and the abstraction description of its functions. On the other hand, the structural components and their ports are developed and dissected to make feasible the interconnection matching.
3. F to R: This *conversion* relationship describes the interconnections of functions to requirements. Specifically, it describes how an implementation of a function satisfies one

or more requirements. In some cases, a group of functions might be clustered to describe how to satisfy one or more requirements.

4. S to R: This *conversion* of the interconnections is between requirements and forms. These relationships are mapped directly from the requirement modeling activity to the structural modeling activity, where they by-pass the functional modeling. This *conversion* of the relationships is not always formulated. It is only shown when more detailed descriptions such as constraints, need to be specified. For example, "The heating unit should have low power consumption," can remove "Gas Burner," "Electric element," and "House heat supply" selections and keep "Solar." Furthermore, this interconnection is used to determine the network of parameter values of components, for example, maximum and minimum values or a range of values. Also, it eliminates the candidate components that violate any requirements.

Note: The two *conversion* interconnections R to S, and S to F were discarded since they do not have a significant impact on the conceptual design process.

4.2.3.5 "*Confidence*" Interconnection Classification

Confidence interconnection provides a description of the degree of certainty of the interconnection. It can be expressed in three levels: High, normal, and low. When an interconnection with high *confidence* is made by an engineer, for example a domain engineer, this high certainty can be taken into account when reviewed by another engineer, for example a system engineer.

Classifications are defined in a constraint language, where the description of the language has the characteristics of *Object Constraint Language* (OCL). OCL is typically used as a navigation language for a graph-based model. It provides precise expressions that are free of

ambiguities of a natural language, and the complexity of mathematics. The representation of *Conversion* interconnection classification in OCL is as follows:

inv ConversionDetail:
self.requirement_link->size()=1 then self.function_link->size()=1 implies Interconnection.conversion=Conversion::R_to_F and
self.function_link->size()=1 then self.structure_link->size()=1 implies Interconnection.conversion=Conversion::F_to_S and
self.function_link->size()=1 then self.requirement_link->size()=1 implies Interconnection.conversion=conversion::F_to_R and
self.structure_link->size()=1 then self.requirement_link->size()=1 implies Interconnection.conversion=conversion::S_to_R

In this language, *requirement_link, function_link, and structure_link* describe the *conversion* characteristics between requirement, functional, and structure models, respectively. The operation *size()* represents the multiplicity of the *conversion* interconnection.

4.2.4 Interconnection Model in UML for Complex Mechatronic Systems

The proposed interconnection model is developed using UML, and OCL syntax, and is used as shown in Fig 4-4. It facilitates the representation of process activities between different models throughout the development process. There it provides a standard description for the interconnection that does not provide room for ambiguous interpretation between the project team and the discipline engineers.

The attribute **name** stores the name of the interface to differentiate from others, which makes it easy to trace changes. The attributes **allocate**, **type**, **class**, **conversion**, and **confidence** represent the interconnection classifications. The **visibility** attribute defines the accessibility of the information of the interconnection by other users, domain engineers, or stakeholders. There are three modes are given: *Public*, *Private*, and *Protected*. **Parameter** is a separate UML class that defines the parameter values in an interconnection. Different cases of parameter definition can be applied. The exact value can be applied; however, in many cases, the application of the parameters

is based on intervals of values, for example, between a maximum value and a minimum value. In other cases, only limit values are applied. **Configuration** exhibits the operation approach, and OCL defines it. Moreover, an interconnection can be decomposed into sub-interconnections. Therefore, an interconnection can be an aggregate of separate interconnections.

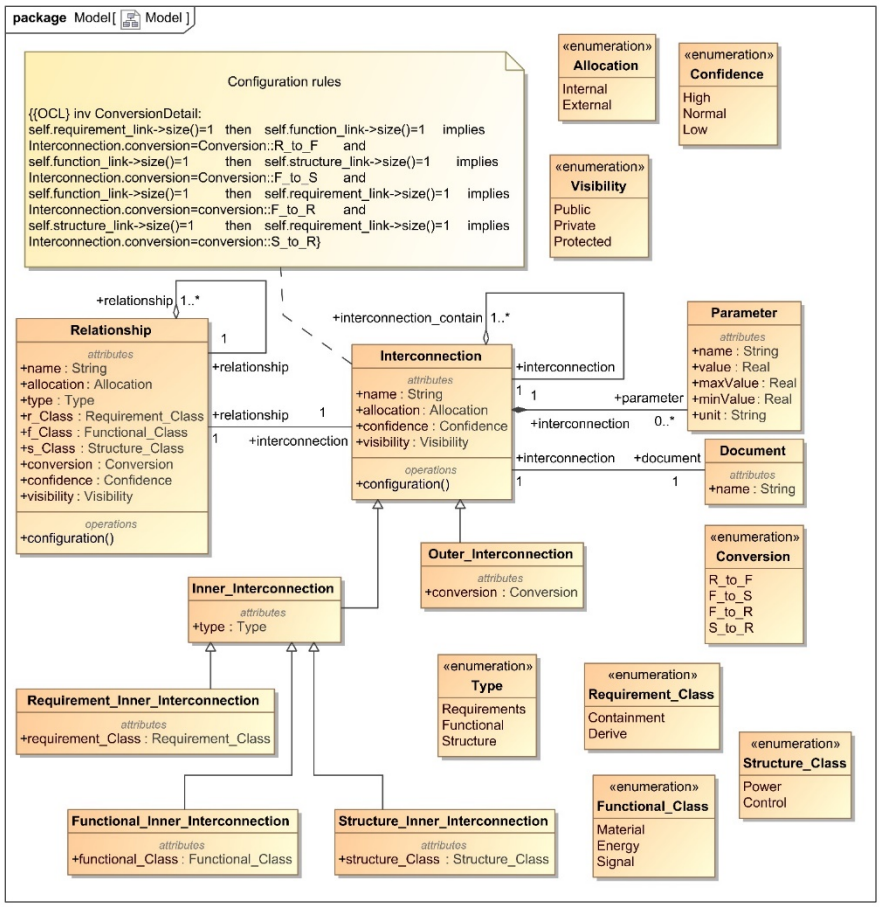


Figure 4-4: Interconnection model in MML class diagram.

4.3 Case Study Implementation

This section investigates the implementation of the proposed interconnection model on an electro-mechanical conveyor system. A sub-system of an industrial fish processing machine. The development of the conceptual phase of the system takes place in three stages; requirement

modeling, functional modeling, and structural modeling. A practical implementation of the proposed model is to be examined and evaluated for all the conceptual development phases.

4.3.1 First Stage

Requirements are presented in the *Requirement Diagram* [20]. The requirement elements are identified as modeling objects. A hierarchal structure is established to facilitate the interconnection between requirement elements and elements from other models, as presented in section 2.2.1.

The main requirements are allocated at a higher level of the hierarchy, where they are decomposed into more detailed sub-requirements. Depending on the level of abstraction, more than one hierarchy can be structured to improve flexibility. However, this would increase informality, ambiguity, and complexity. Therefore, the *interconnection model* is established between these elements. An advantage of the *interconnection model* as represented in Fig. 4-4 is that it is standardized across all the modeling views of the sub-process phase. An example of an established implemented *interconnection model* in requirement modeling is given in Fig. 4-5.

The left side of Fig. 4-5 gives an excerpt of the requirement model of the *Conveyor System*. One example of a requirement element is system **Operation**, which is broken into the sub-requirements: **Operator**, **Input Power**, and **Driving System**. Instances of the *Relationship* UML object are generated to model this relationship. These instances have all the classifications presented in Fig. 4-4. However, only the relevant classifications are defined for each corresponding interconnection. For example, in Fig. 4-5, two instances of the *Relationship* UML object are shown. These instances have similar classifications, such as *allocation*: **Internal**, *confidence*: **Normal**, *type*: **Requirement**, and *visibility*: **Public.** On the other hand, two classifications are

dissimilar: *name* and *r_Class*. The *name* is described in an editor with free language, whereas the *r_Class* is a selection between **Containment** and **Derive**.

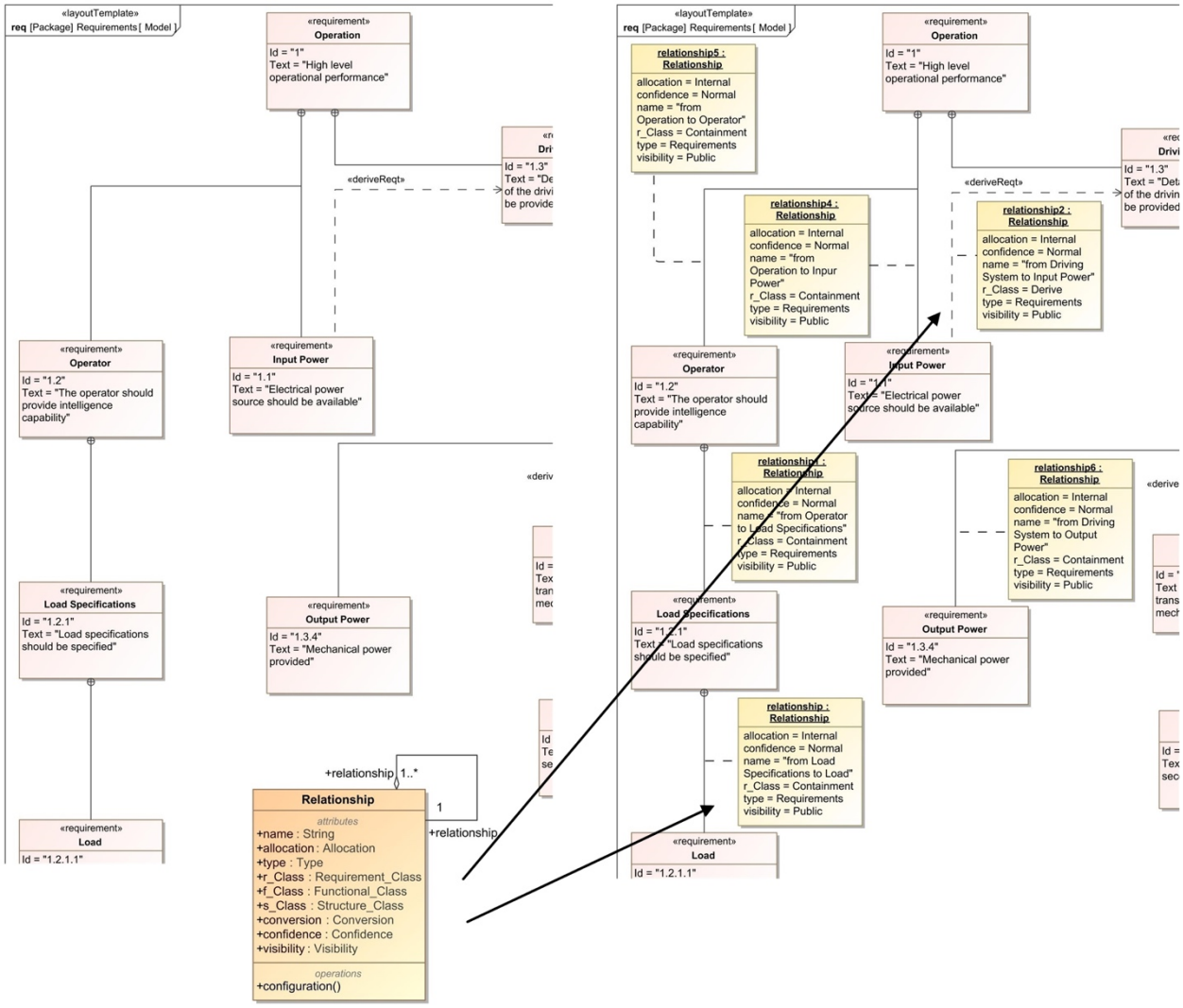


Figure 4-5: The implementation of the interconnection model in the requirement model.

In comparison to other modeling representations, standard Requirement tables can provide valuable information and arrangement for the requirement elements. However, such representation of requirements does not benefit from the support relations among the requirement elements that are integrated into the same model [20].

4.3.2 Second Stage

The development of the hierarchical structure of the *Functional Model* is done according to the hierarchical structure of the *Requirement model*. This development process requires the generation of the intended functions with multiple levels of abstraction, in which the exchange of information is necessary. The relationships between different functions in the *Functional Model* can be modeled with the *interconnection model,* as shown in Fig. 4-6.

The *functional model* is represented in two different diagrams in SysML. Details are given in section 5.3.2. *Block Definition Diagram* (*BDD*) demonstrates the relationships between the parent functions. Also, it supports the hierarchical construction of parent functions with their child functions. On the other hand, *Internal Block Diagram* (*IBD*) supports the nested-network structure, where the relationships between the parent functions and the child functions are established. Fig. 4-6 shows the *interconnection model* is implemented in both diagrams of the functional model of the *conveyor system*. For example, the *BDD* shows the function **Translate**, which is obtained in correspondence with requirement **Speed**. An instance of this relationship UML object is developed, which defines all the information of this relationship, such as *allocation:* **External**, *confidence*: **Normal**, *name*: **from Speed to Translate**, and *visibility*: **Public**. Similarly, the *interconnection model* is established in the *IBD* between the two functions: **Translate** and **Export**. The *interconnection model* has the following information: *allocation:* **Internal**, *confidence*: **Normal**, *name*: **from Translate to Export**, *Type*: **Functional**, and *visibility*: **Public**.

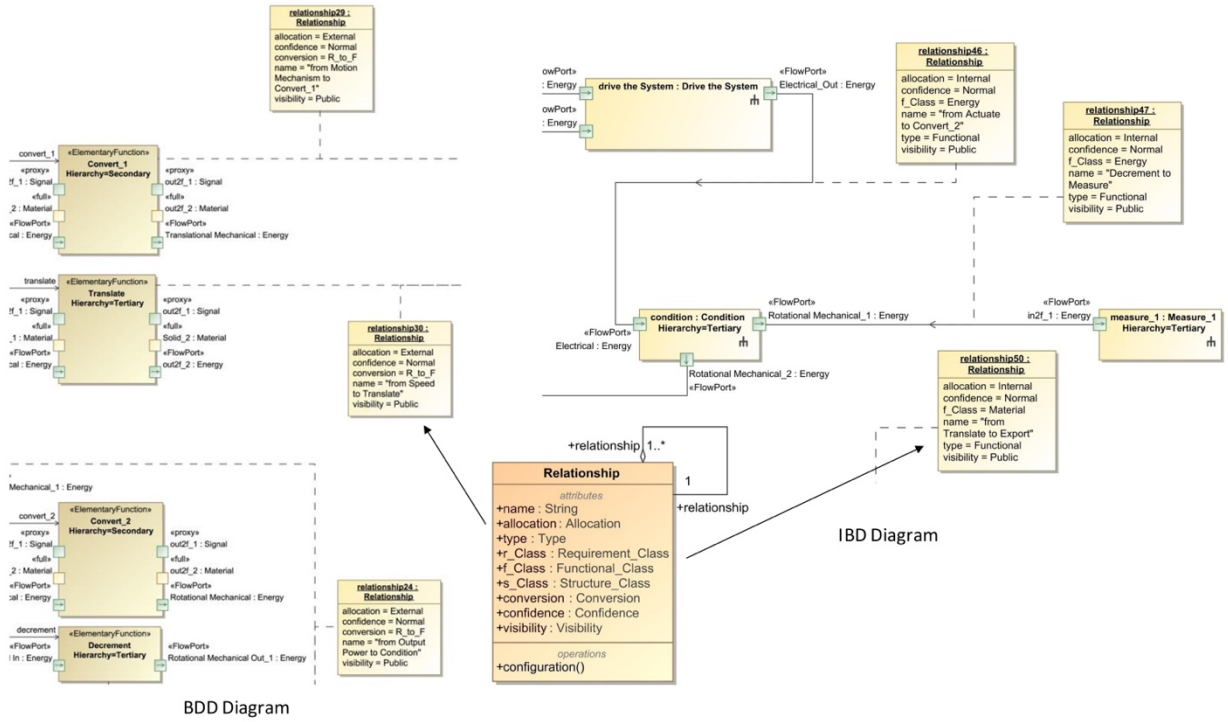


Figure 4-6: The interconnection model in the functional model – BDD diagram (left), and IBD diagram (right).

4.3.3 Third Stage

This stage involves the establishment of structural models according to the *requirement*, and *functional* models. It entails the generation of various system solutions that satisfy the customer needs. The simulation models are created from a collection of a large library of Amesim simulation components [27]. The conditions of the integration are discussed in detail in Chapter 6. These structural models are a part of the concept and model sub-system, where they contribute to the development of the CIM. Therefore, the application of the *interconnection model* is considered, which facilitates the traceability capability of the process.

Fig. 4-7 illustrates an excerpt of the structural model of the electro-mechanical conveyor system, showing the implementation of the *interconnection model*. One example is the relationship between a Geneva wheel and a mass, in which power information is exchanged. Such an

interconnection model develops the following: *allocation*: ***Internal***, *confidence*: ***Normal***, *name*: ***from Geneva Wheel to Mass***, *s_Class*: **Power**, *Type*: **Structure**, **visibility**: *Public*.

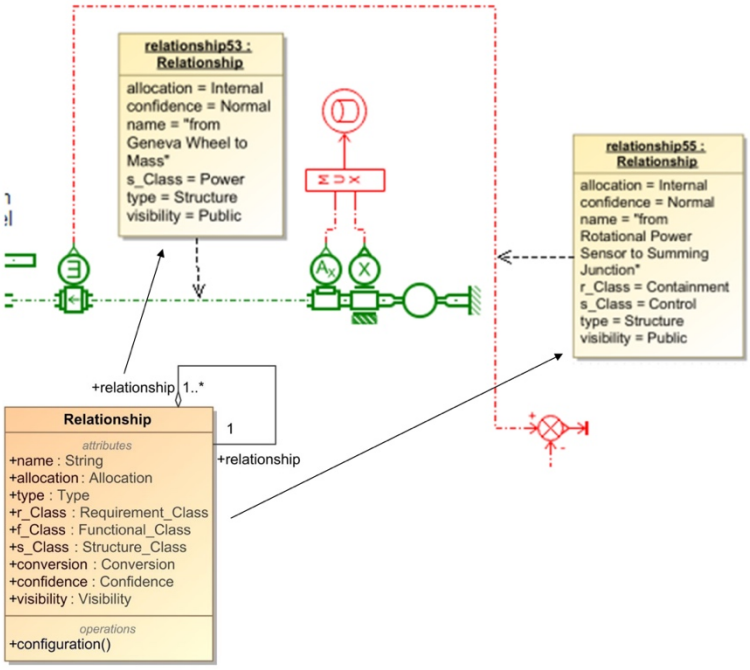


Figure 4-7: The implementation of the interconnection model in the structural model.

4.4 Discussion

The increased complexity of mechatronic systems poses a challenge in terms of model-based and computational support for the conceptual design phase. Interconnection modeling helps to facilitate the complexity of the modeling and they need to be defined formally for all the model views of the conceptual development process. The implementation of the proposed approach was demonstrated in a case study. Different aspects need to be considered for a justifiable evaluation of the approach.

The syntax used classes classes from the Unified Modeling Language (UML), which can model different model views, i.e., requirement modeling, functional modeling, and structural modeling. Therefore, the scope and scalability of the approach can cover situations from simple to

complex applications. Also, the approach gives a good overview representation of the interconnections. Moreover, since the approach uses a drag-and-drop method of the classes, it allows flexibility to modify and adapt to address new problems.

The degree of interaction with the designer is an additional aspect to be considered. The unified language shared between the different designers from the different domains gives the approach a very high degree of freedom. In addition, it contributes to enhance the understanding of the product, where all the details of the interconnections are visually represented.

Finally, this approach has the capability to represent state transformation between the different model views. For example, *conversion* interconnection identifies they type, and direction of element transformation, i.e., from **Requirement modeling** to **Functional modeling**.

Future work includes the investigation of developing interconnections library elements for a wide range of application to be re-used when modeling. This will greatly reduce the modeling effort and time.

Chapter 5: A Library-based Concept Design Modeling Approach in SysML

5.1 Introduction

A rapidly increasing complexity of the mechatronic system correspondingly increases the complexity of the design processes. It led to the need for evolved methods to support the phase of conceptual design development of a mechatronic product. Computational methods are necessary to enhance the necessary support, particularly through model-based conceptual design. For example, [20] expresses the need for further improvements in model-based and computational support for conceptual design development.

The functional modeling provides additional reinforcement to the conceptual design development phase, which enable the understanding and representation of the product functionality. It is a way to describe the product based on its goal, and not on the technical implementation [131]. The nature of functional modeling, as a domain-independent and solution-independent approach, assists in enhancing the communication and understanding between different engineering domains, and across the entire conceptual development process. However, since functional modeling has not been introduced for the computational support, but rather for paper-based methods [20], the introduction of a suitable modeling language is necessary. Furthermore, generating a functional model is a highly subjective and challenging process. This capability is provided through SysML, which is a standardized, graphical, and general-purpose modeling language [68]. It addresses systems engineering through a holistic modeling approach. Therefore, it has a high potential in providing a standard language for functional modeling and facilitates the implementation of computational support. Nevertheless, due to the language informality of SysML, different modeling interpretations are available for computational functional modeling. Various modeling approaches have their strengths and weaknesses, which

depend on the particular application and design intent. Different functional modeling has been suggested, including flow-oriented, relational, and network structure [131].

The functional Basis (FB) is an accepted and widely used supplementary library for the functional modeling that increases the modeling systematism, consistency, accuracy, and formality. Therefore, the modeling approach in SysML is able to integrate the Functional Basis (FB) library into the modeling implementation.

The present work takes advantage of SysML for the functional modeling of complex mechatronic systems, and to integrate it into the CDDP to allow traceability. A flow-oriented modeling approach is developed in SysML through structural diagrams. In addition, a compatible FB library is modeled to enable re-use of functions and consistency checking. Functional benchmark protocols are conducted to express the strengths and weaknesses of the developed approach compared to other existing approaches, particularly related to the network structure. The followed comparison criteria are: representation characteristics, modeling and cognition dimensions, and enabled reasoning activities as suggested by [132]. It contributes to the development of the CIM, where the focus is on the functional modeling activity and its implementation in the concept design and modeling sub-system.

5.2 Background and Related Work

Object-oriented system engineering (OOSEM) [72] is an extension of the system engineering approach, as indicated in section 2.2. It makes use of an object-oriented method for system modeling processes. Functional modeling (FM) has been integrated into system modeling as a way to guide the logical structure. In system engineering, FM is a representation of functions to describe the process of identifying innovative solutions [78]. Functional structure is the graphical representation of functional models, where different functional and structural

representations can be developed for a given product [122]. A widely used representation is the hierarchical structure, in which functions are decomposed into sub-functions. The work in [133] clarifies the ambiguities in the representation of functions. In this case, two representations of functions were identified: Semantics, and syntactic. Kruse *et al.* in [134] have performed a comparison between two structural representations of functions in SysML for the modeling of a *Hydrokeratome*. Computational support for FM has been developed in [20], where SysML language was used for FM in the *Activity Diagram*. The description of the diagram was based on the flow of *Energy, Materials*, and *signals* in a network manner. [131] proposed another functional representation, where the hierarchical structure was maintained for FM in the *Block Definition Diagram*. This way, the tracking ability connection between different model was clearly expressed. In [135], Wölkl has developed a computational knowledge base in SysML to support model-based conceptual design. The functions and flow taxonomies of the National Institute of Standards and Technology (NIST) Functional Basis were defined and transferred in a SysML compatible language and a model library was built. [23] refined the FB library of Wölkl, in SysML, where additional extension *packages* and *containments* were set-up.

The present work formalizes the preliminary work of [131]. Consistent syntax and semantics are provided, and compatible FB libraries are developed to enable model reusability. This work addresses the functional modeling aspect of CIM.

5.3 Function Modeling Approach in SysML

This section introduces a functional modeling approach in SysML, which describes the designed product from a function-oriented point of view. Design libraries are developed for the functional modeling phase of CIM. The general approach followed here is based on the MBSE principles, where the design principles of CDDP and CIM, as presented in detail in Chapter 2, are

followed. In previous studies such as [20], [23], [80], the functional modeling was described by the *Activity Diagram*, as these diagrams are powerful in representing the concept of operation. However, considering the nature of FM, different modeling representations can co-exist [82]. Taking advantage of the capability of the computational modeling language SysML, *Structure Diagram* can also be utilized for the representation of FM [131].

Extending the work to formalize the representation in SysML is needed. The incorporation of a knowledge-base to support the representation, in the form of reusable libraries, has an added value in facilitating the process. The aim here is to provide a structural, comprehensive, and efficient functional representation of FM to support the conceptual design phase that can be integrated into the overall proposed CDDP and CIM. Case tool MagicDraw is a computational modeling tool from NoMagic, Inc. [69], which is used for modeling UML/SysML in the present work.

5.3.1 Structural Syntax Representation

The model representation that is exercised here for the functional model structure is a combination of a tree-based and nested-network structures. This model can be integrated into the CDDP, where the *Requirement Modeling* is used to develop the *Functional Model* [134]. In SysML, the main diagrams in *Structure diagram* are *Block Definition Diagram* (BDD) and *Internal Block Diagram* (IBD). IBD can be developed into a *Parametric Diagram*. *BDD* is composed of *Blocks* and *Connectors*.

For the functional model context, *Blocks* in BDD are used to model system functions, which provides a unifying concept of a function element (also known as; nodes, verbs, or transformation). Whereas, *Connectors* represent a flow element (also known as; edges, or nouns).

The hierarchical structure of the FM is developed in the BDD, in which the relationships between the *Requirement Modeling* and the *Functional Modeling* can be explicitly represented.

At this point, the FM flows are not yet modeled. Only the relationships between the parents or more abstract functions with the children functions are described through *Compositions*. After the establishment of the function elements, the relationships between these function elements are established. Each parent, in the FM element in BDD, can develop an internal view of its components or children elements via the IBD. A nested-network structure of the FM function elements is developed in IBD, where these elements are interconnected relative to each other. These interconnections may reveal additional FM elements for any missing functions in the network structure. Furthermore, the connections between these function elements in IBD represent the FM flows, which have different characteristics.

Ports and *Connectors* are used to model the **Functions** and **Flows** in IBD. The consistency of the flows between any adjacent FM elements should be ensured. Therefore, *port* representations for the function elements in IBD have to be developed in order to identify these connections and to ensure consistency.

Port Representations

The flow representations here are according to the definitions of the Functional Basis (FB), as given in section 2.3.2.1. Three representations of the flows are: **material**, **energy**, and **signal**. Therefore, three types of *ports* in SysML are introduced to model the interaction of **flows** and **functions** according to the FB. *Connectors* are used to connect *parts*, and they need *ports* to specify the details of the interconnections. These *ports* are represented as a black box interface on the *parts*.

The first type of *ports* in IBD is **flow ports**, which is employed to model the **energy** flow. **Flow ports** contain the flow direction and the flow properties, in which more description can be used for any additional information about the incoming or outgoing flows. **Full ports** can represent the flow of **materials** between the connected function elements. **Full ports** can characterize a part of the system, which can handle incoming and outgoing items. Finally, **proxy ports** are used for the modeling of **signals**. They provide access to/from features of its associated *Parts*, which serves as a proxy for the internal parts. The representation of FB functions in IBD is shown in Fig. 5-1.

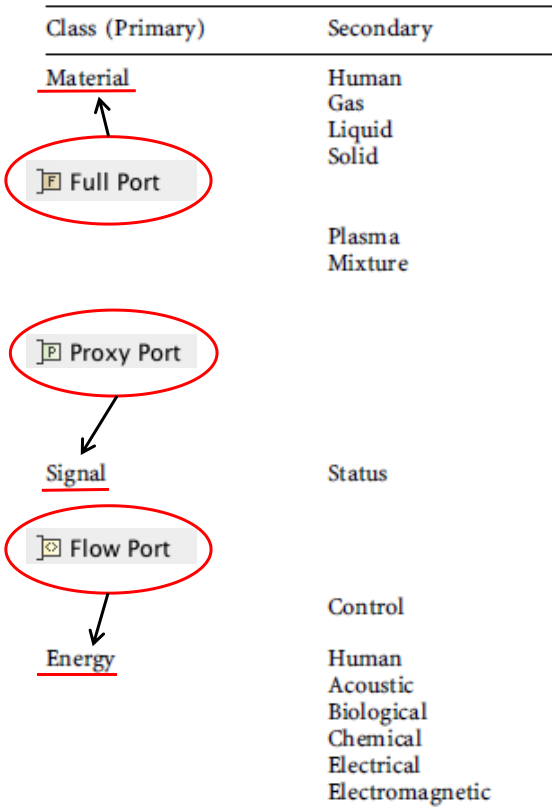


Figure 5-1: Representations of FB Function Class in IBD SysML.

5.3.2 Development of Functional Library Modeling

Based on the work of [23], [135], different modeling directions are developed here, with more details and refinement. Functional Basis taxonomies as presented in [26], are incorporated

into SysML. The approach followed here defines two libraries, one for **functions** and the other for **flows**. The functional model structure is then established by combining elements from the two libraries, in which the elements are ready for reuse. For the proposed functional library modeling, an extension of SysML in the form of stereotypes is established. A stereotype is a specialized SysML block that allows the expansion of SysML by introducing new vocabulary to add more flexibility. Fig. 5-2 shows an overview of the defined stereotypes.

Two *Metaclass* are used for the definition of the stereotypes, *Block* and *Ports*. The *Block Metaclass* defines <<ElementaryFunction>> and <<BasicFlow>> stereotypes. In the present work, these stereotypes are determined by using *SysML Blocks*. There is an additional <<User-definedFunciton>> stereotype, and it can be used for functions, which are not elementary [23]. This stereotype is intended for functions that require more descriptions to satisfy the requirements. <<User-definedFunciton>> can be broken down into any number of <<ElementaryFunction>> or <<User-definedFunciton>>. On the other hand, <<FlowPort>>, <<FullPort>>, and <<ProxyPort>> stereotypes are used for modeling the points where **Functions** and **Flows** interact.

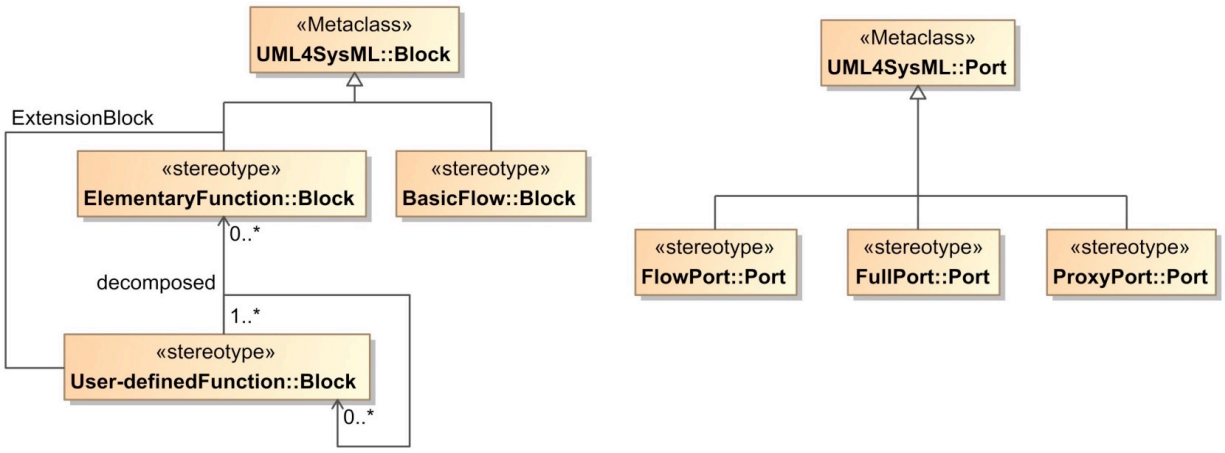


Figure 5-2: Defined *stereotypes* in the function library.

The first step in the implementation of the functional basis in SysML is to define the library content, where a hierarchal structure is adopted in SysML BDD similar to the structure of FM. Figure 5-3, shows the implementation of the flows from the FB, where *SysML Blocks* are used to model the corresponding FB flows [23]. The cascading of the *SysML flow blocks* follows a tree-like arrangement, with each *SysML Block* having a stereotype <<BasicFlow>>.

The top level of the hierarchy has the element *RootFlow*, which is represented as a parent element. The names of *SysML Blocks* are similar to those of the flow FB. In addition, the level of the hierarchy is also indicated from the tag value of the stereotype <<BasicFlow>>. These *SysML Blocks* are connected together with the links *generalization.* It means that each FB flow in the lower level of the tree inherits the properties of its parents. For example, the FB flow "**Liquid"** is transferred into *SysML* structure in the secondary hierarchical level, and it inherits the **Material** properties.

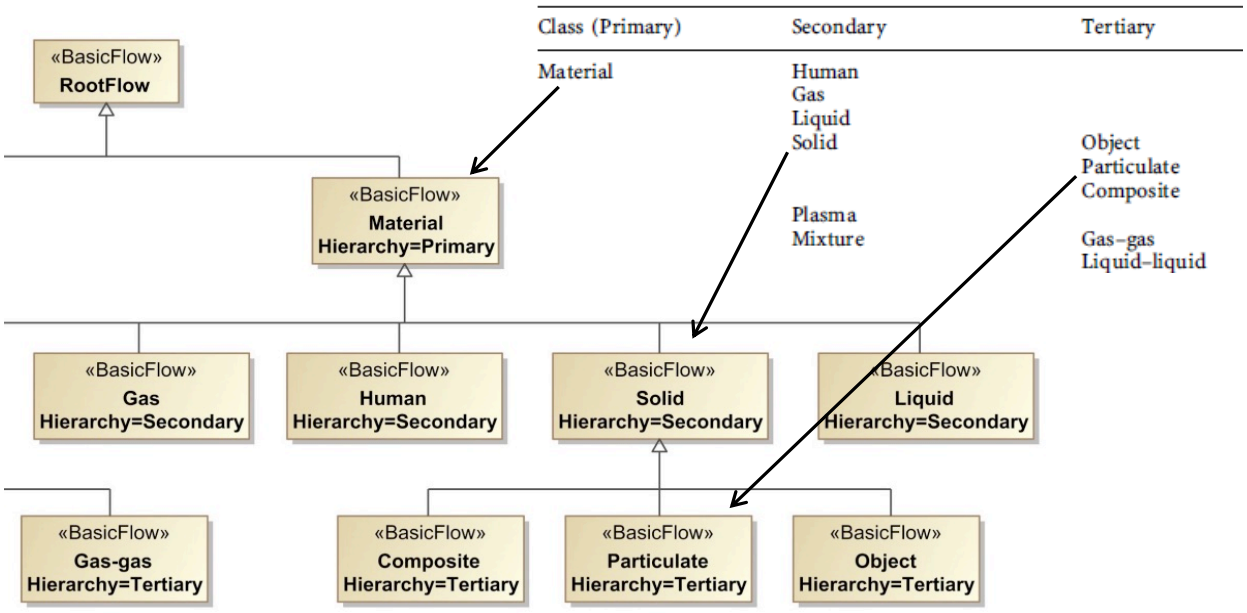


Figure 5-3: Transformation of FB flow into SysML Library.

Similar to the FB flows, the transformation of the functions of the FB is also presented in Fig. 5-4. The same tree-like structure is followed for the modeling of the FB functions in SysML, where the top level of the hierarchy is defined by the element *RootFunction* and the elements in the structure are connected with *generalizations* to inherit the properties of the parent elements. Each FB function transferred to *SysML Block* shows the stereotype <<BasicFunction>> with a tagged value representing the hierarchy level in the structure. For example, the FB function **Remove** has a hierarchy level of *Tertiary*, which inherently carries the properties **Separate** and **Branch**.

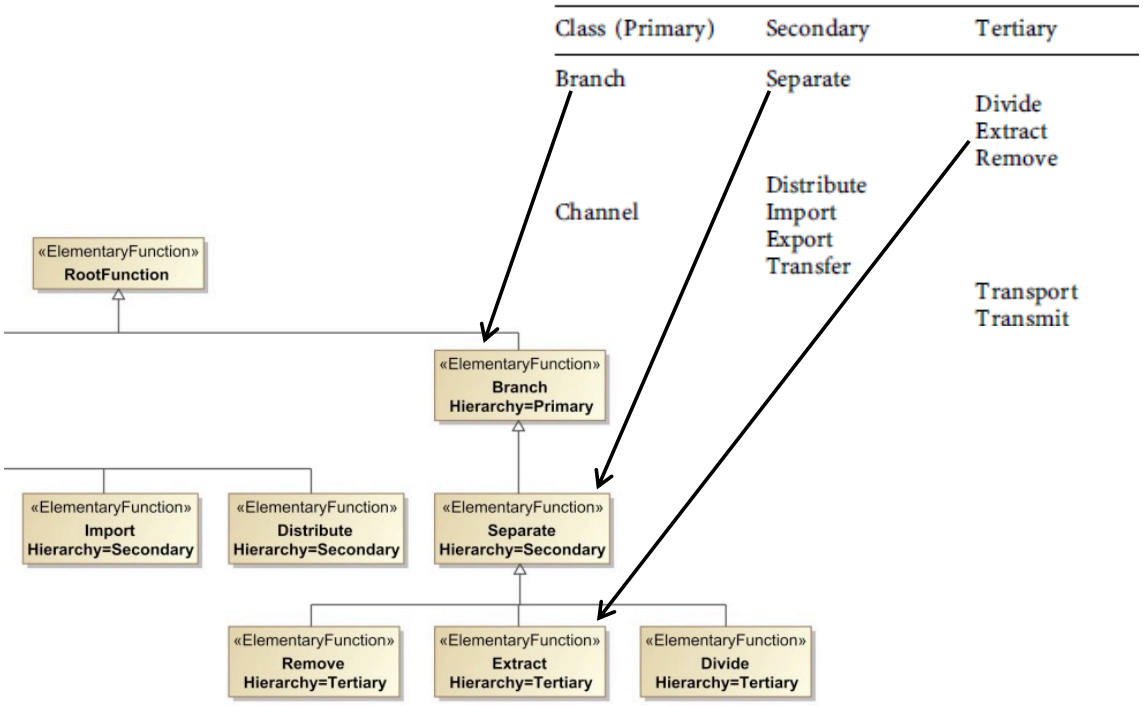


Figure 5-4: Transformation of FB functions into SysML.

Further refinements for the modeling of the FB functions in *SysML* are developed, with reference to prior work [135]. The interfaces of each FB functions are also modeled in *SysML Block*, where SysML Ports represent them. The information about the number and types of all the

incoming and outgoing flows in each FM functions are implicitly provided in the FB. This information is given in the form of a descriptive language.

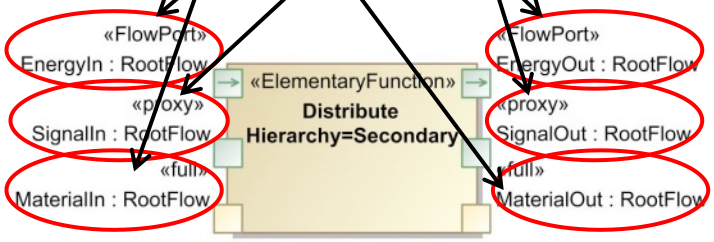
Fig. 5-5 shows two examples, where the ports of the FB functions “Distribute” and “Transfer” are derived from the description. “Distribute” is a secondary-level FB function, as indicated in the stereotype tagged value. The associated **flows** can be derived from the FB description, which allows the flow of any *material*, *energy*, or *signal.*

Fig. 5-5 shows how the three types of *flows* are modeled using the three introduced *Ports* <<FlowPort>>, <<FullPort>>, and <<ProxyPort>>. In the same manner, “Transmit” is modeled in the tertiary hierarchical level, and is described as “To move energy from one place to another.” Therefore, only the <<FlowPort>> stereotype is used for modeling the associated *flows*, as shown in Fig. 5-5.

Standard Functional Modeling (FM) technique is well established for the support of systematic product development [132], and it covers all the points of modeling principles. Therefore, all the views that are identified as modeling principles should also be covered with an appropriate SysML syntax. Table 5-1 presents a comparison between the functional model, modeling principles, and the SysML representation.

b. Distribute

To cause a flow (material, energy, signal) to break up. The individual bits are similar to each other and the undistributed flow. *Example*: An atomizer *distributes* (or sprays) hair-styling liquids over the head to hold the hair in the desired style.



c. Transfer

To shift, or convey, a flow (material, energy, signal) from one place to another.

i. *Transport*. To move a material from one place to another. *Example*: A coffee maker *transports* liquid (water) from its reservoir through its heating chamber and then to the filter basket.

ii. *Transmit*. To move an energy from one place to another. *Example*: In a hand-held power sander, the housing of the sander *transmits* human force to the object being sanded.

Figure 5-5: Examples of how definitions of functions are implemented in SysML.

Table 5-1: Comparison of Functional, Modeling, and SysML modeling representations.

Functional modeling	Modeling principle		SysML representation	
Function structure	Tree-based	Network-type	BDD	IBD
Function	Node		Block	Part
Flow	Edge		Directed composition	Connectors
Differentiated flows for material, energy and signal	Not available	Typed edge visually: (double, plain and dashed)	Not available	Information about the flow
Decomposition	Sub graph principle	Not available	BDD	Not available
Black box	Black box principle		Block with ports	Part with ports
Predefined flow	Interaction point		Port	Port
Scope of function structure	Control volume		Diagram frame	
Flow direction	Directed edge		Directed composition	Connectors

5.3.3 SysML Functional Model Usage with the Library Support

The use of SysML to support a systematic design approach is illustrated now, in the form of a case study. The modeling steps and the method of utilizing the function library are shown. First, the main system functions are defined based on their satisfaction of requirements. The general functions are decomposed into further sub-functions as more detailed requirements need to be satisfied. Each of these functions and sub-functions is modeled as a black box in the form of *SysML Blocks.* More abstract functions are defined as user-defined functions. More detailed functions are modeled from the <<ElematryFunction>> library. The inputs and outputs of each

functions or sub-functions are defined from the <<BasicFlow>> library. Fig. 5-6 provides a visualization of the definition of the **main function** of the present case study, where an excerpt of the *requirement* is shown and it is used to create the user-defined function "Develop a Conveyor System."

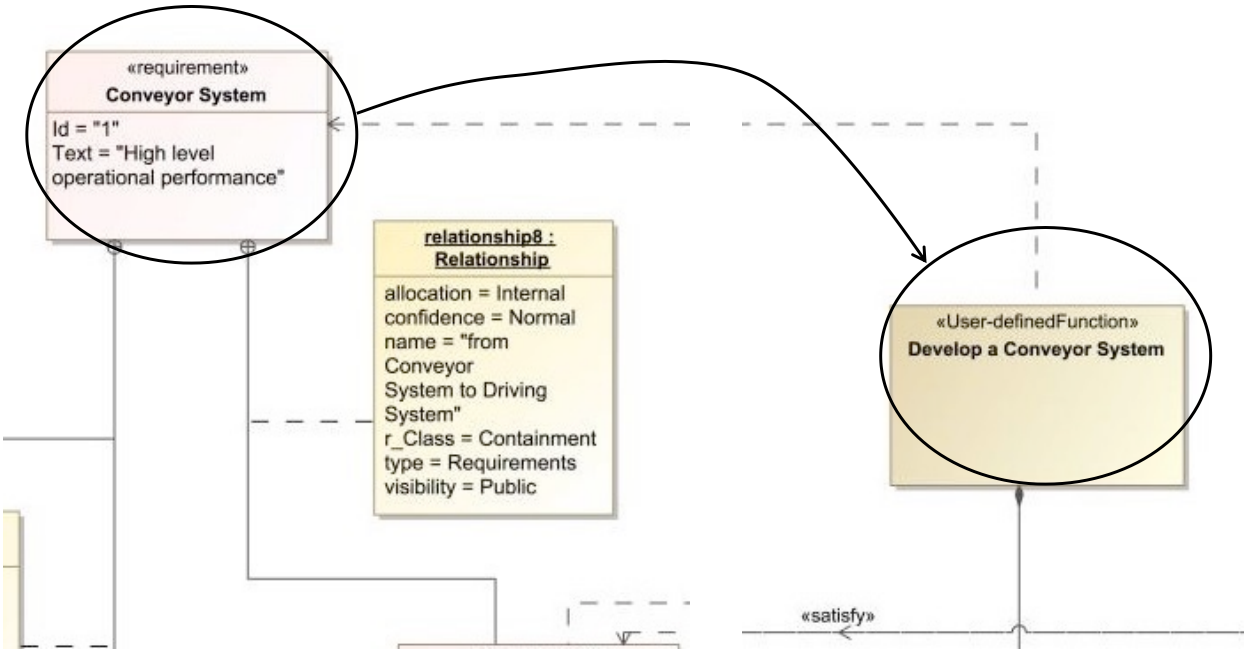


Figure 5-6: Excerpt of developing requirements in BDD.

The main function is, then, decomposed into sub-functions, and some are further decomposed into more sub-functions. This decomposition process is determined by the *requirement* satisfaction. However, the creation of sub-functions should follow the general rules of *verb* and *noun*, where the flow of energy, material, and signal should be maintained as in [92]. The representation of the flows in the <<ElementaryFunction>> is based on the FB description of functions, where additional flows can be added.

The functions "Supply_1," "supply_2," "Drive the System," and "Characterize the Motion" are sub-functions of the main function "Develop a Conveyor System." It is seen that "Develop a Conveyor System," "Drive the System," and "Characterize the Motion" are <<User-

definedFunction>>, whereas “Supply_1” and “supply_2” are <<ElementarFunction>>. Therefore, “Drive the System” and “Characterize the Motion” are decomposed into further sub-functions. In SysML, *ports* are represented from the <<BasicFlow>> library to << ElementaryFunction>> elements. Each << ElementaryFunction>> *SysML Block* in BDD is expanded into IBD, where the energy, material, and signal *ports* are assigned. Fig. 5-7 also shows the implementation of <<BasicFlow>> to the function. Here, it is named “Human Energy,” which is transferred to “Convert_3” <<ElementaryFunction>>.

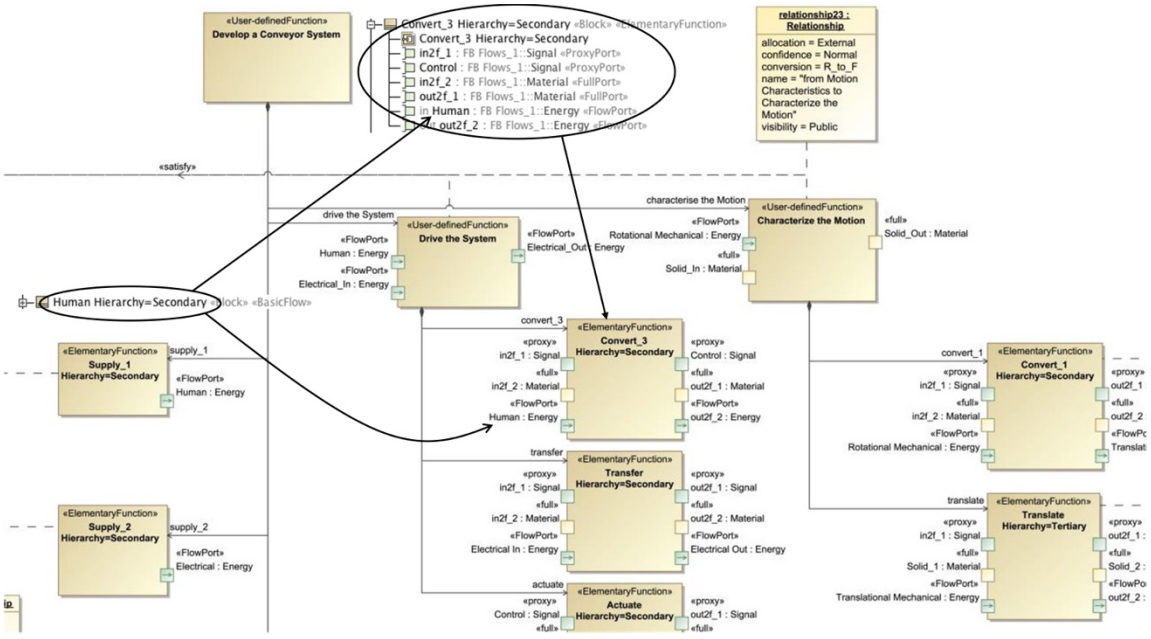


Figure 5-7: Excerpt of the flows representation based on the FB description.

After the completion of the hierarchical structure, the details about the interconnections between these functions or sub-functions are determined. The hierarchy is established based on a nested-network structure. First, parent <<User-definedFunciton>>/<< ElementaryFunction>> *SysML Blocks* develop an IBD diagram inside it. All the associated children *SysML Blocks* of the parent are present inside the IBD with their *ports* already defined. The interconnections are then

established honoring the **material**, **energy**, and **signal** flow type relationship between the children *SysML Blocks*. An example elementary function is described in Fig. 5-8.

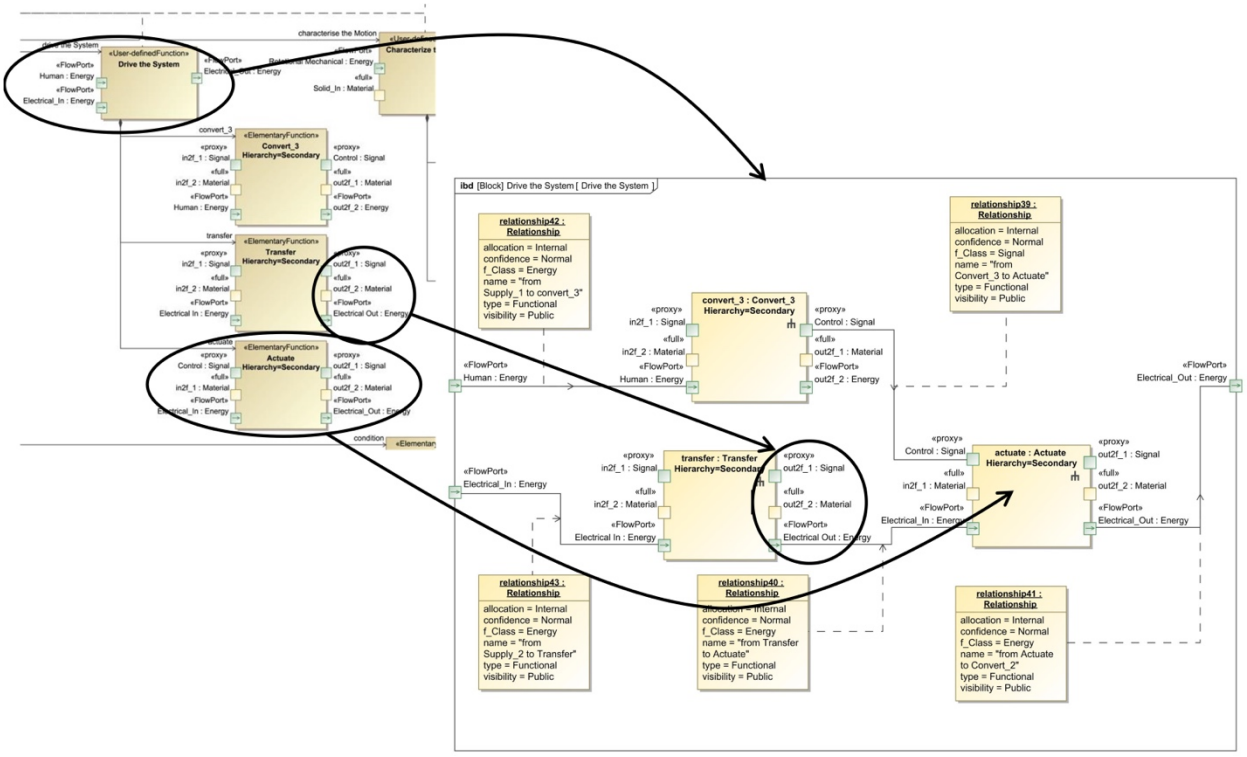


Figure 5-8: The nested-network of the function *Drive the System* is shown in IBD.

An IBD diagram is established inside the parent BDD. All the children sub-functions with their <<BasicFlow>> *ports* are automatically instantiated inside the IBD diagram in the form of *SysML parts*. Then, the interconnections between different *SysML parts* with each other, and the interconnections between *SysML parts* with their environment are created. Here, an IBD diagram is opened inside "Drive the System" *SysML Block*. The <<ElementaryFunction>> "Actuate" is shown inside the IBD diagram. Also, the <<BasicFlow>> ports "Transfer" is presented in the IBD diagram. Finally, Fig 5-8 displays the interconnections between different *parts* in the IBD diagram and with the environment, where the <<BasicFlow>> *ports* of the parent function "Drive the System" are determined.

A special case can be discussed, where the pre-defined number of *ports* are not adequate to describe the complete network of interconnections. In other words, some *SysML parts* have no connection with any of the *SysML parts* in the IBD. For example, a function has a defined pneumatic output flow port, but there are no pneumatic input flow ports defined within the IBD diagram. Two solutions are suggested. One is to define a custom <<User-definedFunction>> in the IBD diagram to extend the library of flows. The other is to interconnect any un-defined ports to the environment, in which the parent <<ElementaryFunction>> is re-defined as a <<User-definedFunction>>. The process of defining the nested-network type structure of each parent takes a bottom-up approach. The main function, therefore, should include all parents and children functions in its IBD diagram, each of which describing the interrelationships.

Fig. 5-9 shows how "Drive the System," "Condition," and "Supply_1" are identified inside the IBD diagram of the main function "Develop a Conveyor System." Also, it is indicated that the sub-functions of "Drive the System" and "Condition" are already stored inside the corresponding sub-function of the IBD diagram.

In addition to *port* definition constraint, SysML IBD provides the diagram constraint, which can be advantageous during modeling. The proper port type must be satisfied when the structure interconnection takes place in IBD, where the port type of one interconnection must be consistent with the port type of the other end. Fig 5-10 shows an indication of inconsistency when the "signal" port type is connected with "material," and "material" port type is connected to "energy" port type, in the form of warning messages.

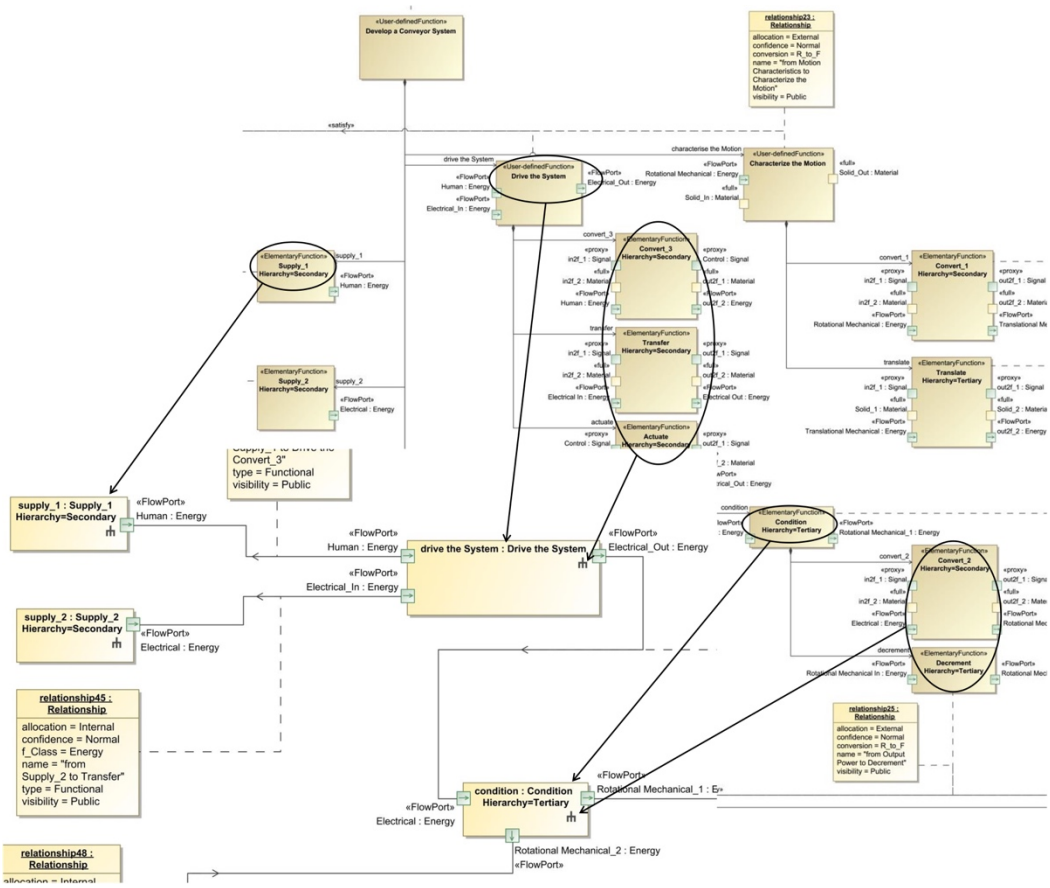


Figure 5-9: The development of FB functions from BDD to IBD.

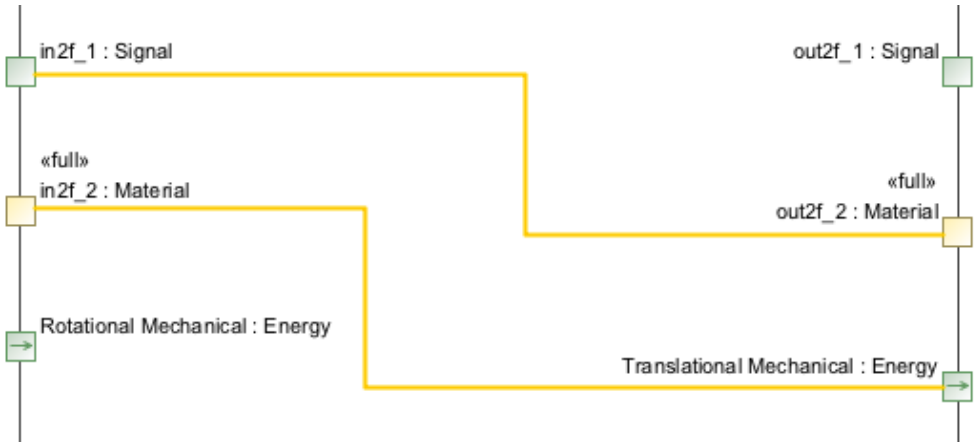


Figure 5-10: *Port* definition constraints.

5.4 Discussion

This chapter developed a computational functional model in SysML to solve some problems of conventional models. In particular, the development and integration of the design library into the SysML functional model, as presented by [131], was conducted to add the library-reuse capability. Other work that utilized the activity diagram for the functional model representation (e.g., [135]), are not included in the present work.

Due to the free nature of the SysML modeling language, some evaluation criteria for the modeling approach in the present work are considered as in [132]. For the representation characteristics, the hierarchal structure in BDD provided a comprehensive functional structure overview and also the representation of the model within the CDDP, which allowed the demonstration of the relationships with other aspects of the CIM model views. Moreover, the presentation of the IBD showed the interconnection between different function elements. However, the introduction of the library would limit the usage of the models to mechatronic systems, which limited the model scalability when addressing complex problems. Flexibility, on the other hand, is improved, where the library element reuse can help in the adaptation of the model to further modification.

Model consistency and model validation were expanded. For example, the support for the introduction of pre-defined *ports* with their defined number and the presence of flow type checking would improve the model consistency and model validation. Alternatively, the indexing ability, which is a description of model binding, and knowledge accessibility are decreased because of the development of two diagram representations. The cognitive dimension characteristics represent the degree of interaction with the designer, which is high in the presented model since the functional and requirement structures are represented in one model. Also, the development of the

process model conventions, as presented in Chapter 4, together with the "pick-n-drop" functions from the library, increases the criterion **closeness of mapping**, which is related to the intuitive nature of the resulting model, reduces the criterion **error-proneness**, and increases the criterion **hidden dependencies**.

From another point of view, **error-proneness** increases with the introduction of the library and *ports*, since the tracking of all the changes of the *ports* types and function decomposition becomes more challenging. Furthermore, the **abstraction gradient**, which indicates the maximum and minimum levels of abstraction, is limited to the <<ElementaryFunction>> being selected from the library.

Finally, in enabling the reasoning characteristics, *state transformations* criterion is discussed. SysML model development in BDD, and IBD can be incorporated in functional modeling. However, they do not fully support dynamic modeling and state transformation [72]. Therefore, the model presented here does not have any reasoning capabilities.

Chapter 6: A Library-based Concept Design Approach

6.1 Introduction and Motivation

Once the functional modeling is completed, structural modeling has to be developed. A structure is a description of how the components or sub-systems of a system are interrelated. These components or sub-systems represent the instantiations of the physical forms or objects of the system, which are introduced according to their fulfillment of the corresponding sets of functions. Therefore, preliminary topologies of the designed product or system have to be generated, and this process is called structural modeling. This contributes to the development of the CIM, where better management of the design date-related problems can be achieved.

These topologies should be simple but sufficiently complete to provide the necessary information to allow model evaluation and evolvement. Consequently, the need for system-level simulation support is crucial, which can shorten the design cycle significantly [104]. Developing a structural model that represents the behavior of the system is required to facilitate system-level simulation. System-level simulation serves as a virtual prototype, where the behavior of the equivalent physical prototype should be modeled as accurately as possible. In this way, the designer is able to perform simulations instead of physical experiments, which makes the design process less expensive.

However, complex mechatronic systems require advanced capability of computational support tools, and these tools must be established to advance the CIM process. Support for model-based design and simulation has been developed mostly in the detailed design phase [136]–[138] since domain engineers determine the exact engineering specifications for a design. In addition, systems are modeled more precisely for discipline-specific domains, and this process is supported by discipline-specific simulation tools. For example, Computer-Aided Design (CAD) and

Computer-Aided Engineering (CAE) tools are particularly intended for mechanical engineers. Electronic Design Automation (EDA) and wire harness design tools are mainly intended for electronic engineers.

Unfortunately, these tools of modeling and simulation cannot be just combined to perform system-level simulation [139] because of the different nature of different engineering domains and also the compatibilities of different simulation tools. Existing system-level simulation tools for mechatronic systems are general and stand-alone, and are not integrated with DDP or CDDP [104]. The present work develops a system-level synthesis methodology for the CDDP of mechatronic systems. It automatically transforms the view of the CIM from a technology-independent model to a structural model that represents the behavior of the system, because the functions are intimately related to the system structure [139]. The developed model view has the following characteristics:

Port-based modeling standards: Port-based modeling standards are developed, in which a designer selects the candidate simulation components by mapping between the inputs and outputs of the functions or a set of functions and components of the structure. The interactions between the interconnecting components can be defined by connecting the ports of the subcomponents.

Physical modeling paradigm: In the physical modeling approach, components are representable by mathematical-graphical objects. These objects are topologically interconnected by rules based on the exchange of power through the connections to form the governing equations.

The present modeling paradigm supports multi-physics simulation, in which different engineering domains are modeled and simulated in an integrated manner under a single simulation environment. This results in lower costs in the physical tests through of using more simple and fewer hardware prototypes.

Design catalogs: Designers are required to use design knowledge that provides the details of object functionalities, from available behavior simulation libraries [140]. Proper development of a simulation library is essential in order to facilitate the mapping between the solution object candidates to functions, via object ports.

Reconfigurability: Granularity of the system-level simulation model is defined at a limited level of accuracy. As the CDDP develops through iterative cycles, the complexity of the model grows with components, and the model accuracy increases without the need to remodel the complete system. Additional components for the control system may have to be introduced and integrated in order to fill the gaps of the system mechanics.

Automatic synthesis: A high-level system synthesis methodology is introduced. It advances the development of the transformation between two different model views of the CIM, which is integrated within the CDDP by utilizing the functional model, to create the structural model. In this way, the synthesizer is able to develop a system of equations automatically, which describe the behavior of the system.

A system-level simulation model development algorithm is introduced. It allows an automatic generation of model synthesis that represents the topological structure of the simulation model. Also, the values of the network of parameters that fulfill the functional requirements are roughly determined; particularly the lower and upper bound values and the acceptable range of values.

Traceability capability: The algorithm enables the capability of traceability during the synthesis process following the presented methodology (see Fig. 4-1). It allows automatic information exchange between different model views of the CIM through the CDDP (see Fig. 4-3), which is the tracing of the structural model to functional and requirement models.

6.2 Background and Related Work

The development of structural-based design, also called architecture-based design [138] and platform-based design [141], in CDDP provides a high-level of structural abstraction, in which the functional model is utilized to transfer the model view into representations of multiple behavioral architectures. The shortcoming in the existing structural modeling developments is that they are locked to a particular implementation in making the design decisions [142]. Also, the modeling capabilities are unable to represent physical components and their interactions as computational components [138].

The process of transforming a functional model of a product into its structural model is a development process, as described in [32], [123]. The developed methodology is based on characterizing the product by three levels of abstraction: form, function, and behavior [37], as shown in Fig. 6-1.

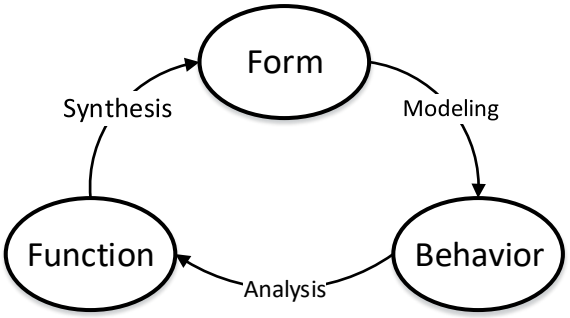


Figure 6-1: Relation between form, function, and behavior.

The first level of abstraction is the function level, where the design tasks are formulated into functions. These functions are decomposed and modeled into a different level of detail – see section 2.3.1. In the second level of abstraction, those function descriptions are directly related to the structural descriptions of the object. They describe the form of the physical embodiment of the object, particularly the structural level, as indicated in section 4.1.1.3. The third level of abstraction

characterizes the behavior of the system, which is derived from the forms of the objects. This behavior is verified to see if it matches the developed functional and non-functional requirements, where mathematical models represent the behavior. The design verifications are established by carrying out simulation and analysis of these models.

Simulation tools and languages have been utilized heavily in many industries for the development of complex systems through improved design processes [104], [139]. They have proven to be effective and economical in design, validation, and testing the systems. Virtual analysis of the system will reduce the need for physical prototyping, where the dynamic behavior of the system can be predicted at a high degree of certainty and under a variety of conditions. However, many ongoing research activities still address the development of activities such as model validation, different specific-domain simulation tool integration, and utilization of virtual reality technology to enhance the visualization. The present work seeks to simplify the model formulation process, in which system-level simulations can be conducted to facilitate the identification of system-level problems.

Modeling and simulation packages can differ according to their characteristics such as graph-based or language-based, multi-domain or single-domain, and declarative or procedural modeling. Modelica [143] is a language-based modeling methodology based on the bond-graph approach, where the language can be described as typed, declarative, equation-based, and textual. The modeling methodology is non-casual and uses object-oriented construct, which was inspired by software development for modeling physical systems to describe their dynamic behavior. It has the following characteristics:

Multi-domain: The modeling language is adequate to model different engineering domains such as electrical, mechanical, thermal, and fluid. Therefore, it is suited for modeling complex mechatronic systems

Object-oriented: General class concepts are defined with a strongly typed language.

Declarative language: Equations and mathematical functions allow acausal (non-causal) modeling, in which the internals of sub-models can be completely encapsulated. The acausality makes Modelica library classes more reusable than traditional classes that contain assignment statements where the input-output causality is fixed.

Modelica language has been implemented in several open-access and commercial simulation tools such as Simcenter Amesim from LMS [27]. Amesim has a built-in graphical editor that can graphically represent different mathematical equation systems, which describe the component behavior (see section 2.1.3.2.2). Amesim is based on a 1D, lumped-parameter, time-domain simulation platform software. This means the geometry of each component is not directly resolved. Hence, it can be used as a unified environment of system-level simulation for complex mechatronic systems, because it provides a systematic means of cataloging and classifying component design knowledge that is organized in libraries. The individual components are represented by standard symbols used in the engineering field; for example, ISO symbols for hydraulic components. Also, the libraries are classified according to the engineering domains. Amesim is still widely used for the modeling and simulation of multi-domain systems [144]–[147]. However, the identification of model components has to be done according to the design specifications represented by functions. Therefore, the selection of components that have correspondence with functions, is practiced

The work of [28], [148], [149] used the identification of structural components by structural mapping. Kurtoglu [28] used graph grammar for automated generation of a configuration flow graph. This means components are found in a library structure parallel to the functional basis. [148], [149] utilized Triple Graph Grammar (TGG). With structural mapping, having unique, redundant-free models is not guaranteed, whereby a component embodying two functions can be placed in more than one location in the library structure. An alternate component selection from a library would be achieved through the identification of inputs and outputs from a mapping matrix.

[23], [150] developed a system-level model integration based on SysML to overcome the SysML inability of not being able to support the simulation of behavioral models. Therefore, extensions are made to transform between the hybrid models in SysML to executable simulation models in Simulink/Simscape, and Amesim simulation environment, respectively. The capability of automatic simulation model generation was not developed in their work, and therefore required a manual selection of components.

[138], [139] established algorithms for automatic and semi-automatic simulation model generation of cyber-physical systems. The transformation of the simulation models was done according to the functional model development, and two levels of synthesis were presented: architecture synthesis algorithm and simulation synthesis algorithm. This method increased the computational cost, and the complexity of the algorithm is $O(N * M * K)$,

where:

N: the total number of architecture components in the architecture library

M: the total number of architecture templates in the architecture library

K: the number of functions in the functional model

Their work primarily focused on the development of the architectural library for the automotive industry. Therefore, a goal of the present work is to develop an algorithm for automatic simulation components synthesis, to reduce the computational cost and also to broaden the application to other industries such as the food processing machinery industry.

6.3 Functional Model

The development of an algorithm for transforming the functional model into a simulation model requires an algorithmic description of the model. Therefore, the functional model and the simulation model are algorithmically described next.

Definition 6-1. A functional model is a labeled, directed, multigraph $F = (V, E, s, t, L_V, L_E, P)$. Each node $v_{i_{(n,m)}} \in V$ represents a function, and has a defined input port $p_n \in P$ and output port $p_m \in P$. Each edge $e_{(i,j)} \in E$ represents a flow from a function $v_{i_{(n,m)}} \in V$ to a function $v_{j_{(l,k)}} \in V$. The mappings of the node source and node target of each $e_{(i,j)} \in E$ are represented by $s: e_{(i,j)} \in E \rightarrow p_m \in P \; of \; v_{i_{(n,m)}} \in V$ and $t: e_{(i,j)} \in E \rightarrow p_l \in P \; of \; v_{j_{(l,k)}} \in V$, respectively. L_V, and L_E are the vocabulary from the Functional Basis Language.

The functional basis (FB) – see section 2.3.3.1 – defines vocabulary, syntax, and semantics of the functions, constituting the functional library, which is encoded into V, E. For example, a node $v_{j_{(n,m)}} \in V$ is labeled $l\left(v_{j_{(n,m)}}\right) = convert$ and a flow $e_{(i,j)} \in E$ is labeled $l\left(e_{(i,j)}\right) = electrical\ energy$. By checking the compatibility of the target mapping of $e_{(i,j)} \in E$, t_j and the input mapping port of $v_{j_{(n,m)}} \in V$, p_n, we can achieve a $convert\ electrical\ energy$ signature.

Definition 6-2. A set of requirements $R_{set} = \{R_1, R_2, R_3, \dots\}$ represent the user-defined set of requirements, where the properties are described in general terms.

6.4 Simulation Models

Functions are allocated to candidate simulation components from Simcenter Amesim [27] simulation library, which constitutes a simulation model.

Definition 6-3. A simulation library S_{lib} is represented as a collection of simulation components S_i, $S_{lib} = \{S_1, S_2, S_3, \dots\}$. The simulation components are classified into up to 40 sub-libraries, such as Control, Electrical, Mechanical, Fluid, and Thermal sub-libraries. These sub-libraries are developed through engineering services in partnership with customers.

Definition 6-4. A simulation component is $S_i = \{P, I, Par, CE, SL, Cons\}$. Here $P = \{P_E, P_s\}$ includes energy ports P_E and signal ports P_S. The energy ports are input ports and output ports: $P_E = \{P_{E_{in}}, P_{E_{out}}\}$. P_E is also classified according to the energy domains: $P_E = \{P_{EE}, P_{MRE}, P_{MTE}, P_{HE}\}$. They incorporate the conservation of energy law, which is defined by conjugate variables that represent effort e and flow f. Also,

P_{EE} = ports for electrical energy = Current $i(t)$ and Voltage $V(t)$

P_{MRE} = ports for mechanical rotational energy =Angular velocity $\omega(t)$ and Torque $T(t)$

P_{MTE} = ports for mechanical translational energy = Velocity $v(t)$ and Force $F(t)$

P_{HE} = ports for hydraulic energy = Volume flow rate $\varphi(t)$ and Pressure $P(t)$

The signal ports are input ports and output ports: $P_S = \{P_{S_{in}}, P_{S_{out}}\}$. Here, P_s are signal ports that require minimum power, and are used to communicate a drive action to a component. Signal ports may include data buses and embedded controllers. Each simulation component S_i contains: parameter values Par, constitutive equations CE, internal parameters I, a list of constraints $Const$, and is classified within a sub-library SL.

Definition 6-5. A simulation model S_{mod} has strongly typed simulation components $s_{i_{(n,m)}}$, from the simulation library S_{lib} and connectors $c_{(s_{i_{(n,m)}}, s_{j_{(l,k)}})} \in C$. The simulation components are connected through their defined power ports from $p_m \in P_{E_{out}}$ of simulation component $s_{i_{(n,m)}}$ to $p_l \in P_{E_{in}}$ of simulation component $s_{j_{(l,k)}}$, where the connectors impose algebraic constraints on the port; for example, Kirchoff voltage and current laws in electrical circuits.

6.5 Synthesizer Principles

The simulation components need to be established for the model. Each component embodies an equation and a corresponding code, which represents the mathematical description of the component. A set of equations are developed in order to complete the model. These equations define the behavior of the system, by interconnecting the simulation components. Modeling is based on bond graph principles [98], [151], where the power transmission is defined as the product of effort and flow.

To achieve the automatic generation of a complete model, the synthesizer algorithm identifies components from a model library to satisfy the associated functional model. The selection of the simulation components from the library requires input and output direct mapping to match the input and output of a function/group of functions. The challenge here is defining appropriate inputs and outputs for the contents of the model library [135]. Therefore, the synthesizer is able to capture the energy and signal port information of the functions that are stored in the object ports in the IBD idb of SysML functional model. This information contains the description of the incoming and outgoing flows from and to the functions from the FB reconciled

flow set [152]. The synthesizer matches and stores the corresponding power conjugate complements of each type of flow, as given in table 6-1 [152].

Table 6-1: Power conjugate complements for the energy class of flows.

Class (Primary)	Secondary	Tertiary	Power Conjugate Complements	
			Effort analogy	Flow analogy
Energy			Effort	Flow
	Human		Force	Velocity
	Acoustic		Pressure	Particle velocity
	Biological		Pressure	Volumetric flow
	Chemical		Affinity	Reaction rate
	Electrical		Electromotive force	Current
	Electromagnetic		Effort	Flow
		Optical	Intensity	Velocity
		Solar	Intensity	Velocity
	Hydraulic		Pressure	Volumetric flow
	Magnetic		Magnetomotive force	Magnetic flux rate
	Mechanical		Effort	Flow
		Rotational	Torque	Angular velocity
		Translational	Force	Linear velocity
	Pneumatic		Pressure	Mass flow
	Radioactive/Nuclear		Intensity	Decay rate
	Thermal		Temperature	Heat transfer

Embodiments of the component behaviors that satisfy the functions are, then, identified by matching the input and outputs of the correspondent power conjugate complements. These components are then connected through their energy ports by using connectors. Since the functional model consists of functions V and flows E, port matching solves the flows E compatibility. Also, it generates many simulation components, which leads to a growth of the solutions space. This creates difficulty in identifying good or optimal solutions. In order to limit the search space, the functions V; for example, convert, store, should also be addressed.

Commercial simulation libraries provide an extensive source of re-usable, pre-existing, system-level simulation components. However, synthesizing the simulation components based on matching the flow of power ports will lead to the challenges mentioned earlier. Therefore, developing the synthesizer to incorporate compatibility matching between the functions V of the functional model to the simulation components, is necessary. The library of heterogeneous simulation components have to be developed to enable a systematic classification of the functions V. This would link the gap between the functions V and behaviors, support the formalization of the simulation component library, and facilitate better and reliable mappings.

6.6 Behavioral Component Library Development

The behavioral component library is organized according to the functional basis-reconciled function set L_V [152]. It contains elements from Simcenter Amesim [27], which are defined based on abstraction ports, similar to [153]. These abstraction ports are established between the bond graph elements and the functions from the functional set L_V. The underlying modeling principles of Simcenter Amesim modeling environment are bond graphs [98], [151], which are based on the energy flow of different domains. Table 6-2 presents the state variables in different domains, which may include effort e, flow f, their time integrals, momentum p, and displacement q, depending on the physical element, in different engineering domains.

Table 6-2: Domain-specific state variables.

Energy domain	Effort e			Flow f			Momentum p			Displacement q		
	Name	Symbol	Unit	Name	Symbol	Unit	Name	Symbol	Unit	Name	Symbol	Unit
Mechanical translational	Force	F	N	Velocity	v	m/s	Linear Momentum	P	Ns	Linear displacement	x	m
Mechanical rotational	Torque	T	Nm	Angular velocity	ω	$1/s$	Angular Momentum	L	Nms	Angle	α	rad
Electrical	Voltage	U	V	Current	I	A	Flux Linkage	λ	Vs	Charge	Q	A

The mathematical operations that represent the relationships between the state variables and their time integrals represent the physics of different bond graph elements, as shown in Fig. 6-2 [154].

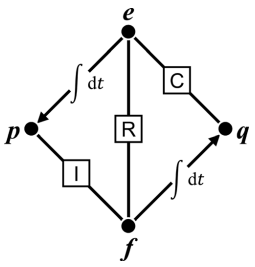


Figure 6-2: Tetrahedron of state.

These elements, specifically, Resister (R), Capacitor (C), Inertia/Inductance (I), Transformer (TF), and Gyrator (GY), will embody functions, and they provide the basis for the matching of functions V and simulation components S_i.

The structure and organization of the library are based on the functional set L_V. A function $v_i \in L_V$ constitutes a bond graph element, which represents a relationship between two state variables. They are used to identify a family of simulation components. Consequently, all of the component families from different sub-libraries are stored in the function v_i. The complete list of the assigned ports according to bond graph elements is found in [153].

Definition 6-6: A simulation component family S_f represents a group of components from the same domain that share the same behavior and interfaces but they have different levels of detail. For example, a function v_i that represents a decrease from the functional set L_V is assigned to bond-graph element Resistor (R), since R dissipates power. Also, R provides the relationship between effort e and flow f using equation (6-1), where k is a lumped parameter that describes the linear relationship:

$$\boldsymbol{e = k.f} \tag{6-1}$$

From the simulation library, friction component family from the Pneumatic sub-library is identified and stored in the *decrease* library, as shown in Fig. 6-3.

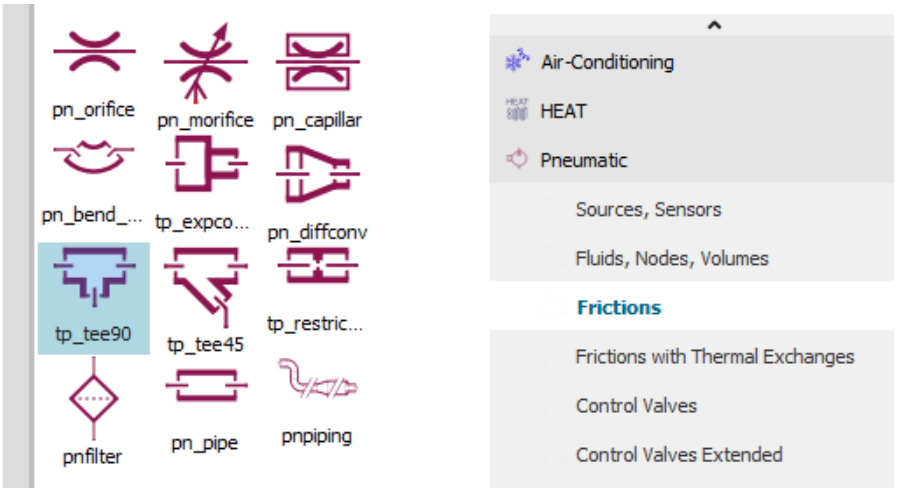


Figure 6-3: *Frictions* family from *Pneumatic* sub-library is to be stored in *decrease* library.

For illustration, the input and output power flows of a simulation component *pn_capilar* (the top right component) has the following description:

In the pneumatic capillary, the flow path is assumed to be between parallel plates. The length of this clearance flow path is assumed constant. The pressure in Pa *and temperature in* K *are input at each port and the mass flow rate in* g/s *and the enthalpy flow rate in* J/s *are computed and output at both these ports.*

This description is illustrated in Fig. 6-4, which shows that the input power is *Pressure (effort e)* and the output power is *mass flow rate (flow f)*, and they satisfy equation (6-1).

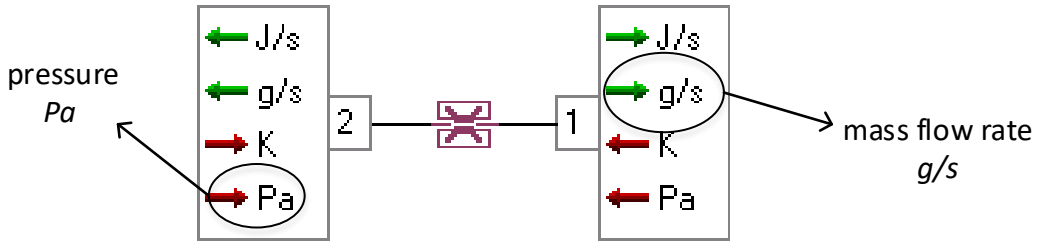


Figure 6-4: The energy flow of a pneumatic capillary.

Synthesizer Algorithm

Definition 6-7: A mapping from a functional model F to simulation component family S_f is a set of 2-tuples $m_i = \{< v_{i_{(n,m)}}, s_{f_j} >\}$, where $v_{i_{(n,m)}} \in V(F)$ and $s_{f_j} \in S_f$. When $m_i = \{< v_{i_{(n,m)}}, \emptyset >\}$, S_{f_j} is defined as is $\emptyset$ does not exist in m_i. $m_{set} = \{m_1, m_2, \dots\}$ represents all combinations of mappings between $v_{i_{(n,m)}}$ and S_f.

The synthesizer algorithm 1 generates simulation models S_{mod} for the functional model F. It follows four steps: the dissection of the component simulation families $s_{f_i} \in S_{lib}$, mapping between the functions $v_i \in V(F)$ and component simulation families $s_{f_i} \in S_{lib}$, identification of the range of parameters for $s_{f_i} \in S_{lib}$, and generation of simulation models S_{mod}. First, lines 4-18 of algorithm 1 examine the ports of the simulation component families $s_{f_i} \in S_{lib}$ and classify the energy and signal ports. They also identify whether the flows and signals are incoming or outgoing, in which the energy incoming flows are decided based on the effort state variables. Furthermore, the synthesizer links each incoming and outgoing energy port to its corresponding FB flow set, as described in definition 6-4. The synthesizer takes into account the presence of more than one incoming and outgoing signal and energy port. Lines 19-29 of algorithm 1 utilize the information stored in each simulation component families, i.e., the internal state equations, to derive FB function set in each component, similar to [153]. Second, lines 30-39 of algorithm 1 create full mapping m_{set} between all the functions v_i from the functional model F, and the the simulation component families S_f. This process takes two steps: first, the incoming and outgoing ports of each function $v_i \in V(F)$ are examined and a search in the component simulation library S_{lib} is conducted to look for the same port types. Second, all possible candidates are subjected to a pruning process, where their FB functional set should match FB functional set of the function v_i.

This is important in order to achieve a correlation between the functions and simulation components. Third, in lines 41-45 of algorithm 1, all the mapped simulation component libraries are investigated against the set of customer requirements R_{set} to identify the network of parameters, specifically the range value or the upper and lower limits. Also, a component mapping is eliminated together with its corresponding function v_i if it contradicts any requirement from the requirement set R_{set}. Fourth, in lines 47-59 of algorithm 1, the generation of different simulation models takes place. Each simulation component library in each mapping m_i is subjected to compatibility port matching with each simulation component library in each other mapping m_j. The incoming and outgoing ports of a component are connected to compatible ports of another component. The unconnected ports of the connected components are then connected to other compatible ports of the other components. Finally, a simulation model is generated and stored in the simulation model S_{mod}.

Synthesizer Algorithm 1

Input: F: A functional model imported from SysML Internal Block Diagram IBD
Input: R: User requirements from Requirement Diagram RD R_{set}
Input: S_{lib}: Simulation library with abstraction ports identification
Input: m_{set}: A set of mapping from $V \in F$ to $S_f \in S_{lib}$
Output: S_{mod}: A set of simulation models

1 $S_{set} = \emptyset$
2 $m_{mod} = \emptyset$
3 For each $s_{f_i} \in S_{lib}$ do
4 For each $p_j(s_{f_i})$ do
5 If $p_j = P_E$ then
6 If $effort(p_j)$ is entering then
7 $p_j = e(p_j)$
8 $p_j = p_j \cup P_{in}$
9 Else
10 $p_j = e(p_j)$
11 $p_j = p_j \cup P_{out}$
12 Else $p_j = P_s$ then
13 If P_s is entering then
14 $p_j = P_s$
15 $p_j = p_j \cup P_{in}$
16 Else
17 $p_j = P_s$

$p_j = p_j \cup P_{out}$

For each $p_{E_{in_j}}(s_{f_i})$ do

For each $p_{E_{in_k}}(s_{f_i})$ do

If $p_{E_{in_j}} = p_{E_{in_k}}$ then

$s_{f_i} \rightarrow \emptyset$

Else

If $e(p_{E_{in_j}}) = e(p_{E_{in_k}})$ then

$s_{f_i} \rightarrow "decrement, prevent, inhibit, decrease, store, supply"$

Else

$s_{f_i} \rightarrow "convert"$

For each $p_{E_{out_k}}(s_{f_i})$ do

$s_{f_i} \rightarrow "increment, decrement, convert, "$

For each $p_{E_{out_j}}(s_{f_i})$ do

For each $p_{E_{out_k}}(s_{f_i})$ do

If $p_{E_{out_j}} = p_{E_{out_k}}$ then

$s_{f_i} \rightarrow \emptyset$

Else

If $e(p_{E_{out_j}}) = e(p_{E_{out_k}})$ then

$s_{f_i} \rightarrow "store, supply"$

Else

$s_{f_i} \rightarrow "increase, decrease, convert"$

For each $p_{E_{in_k}}(s_{f_i})$ do

$s_{f_i} \rightarrow "increase, decrease, convert"$

For each $v_{i_{(n,m)}} = V(F)$ do

$m_i = \emptyset$

$m_i = m_i \times \{< v_{i_{(n,m)}}, \emptyset >\}$

For each $s_{f_j} \in S_{lib}$ do

For each $p_{in_k}(s_{f_j})$ do

If $p_{in_k} = P_n$ then

For each $p_{out_l}(s_{f_j})$ do

If $p_{out_l} = P_m$ then

If $v\left(v_{i_{(n,m)}}\right) = s_{f_j}$ then

$m_i = m_i \times \{< v_{i_{(n,m)}}, s_{f_j} >\}$

$m_{set} = m_{set} \cup m_i$

For each $m_i \in m_{set}$ do

For each $s_{f_j} \in m_i$ do

Store the network of parameters from the requirement set R_{set}

If $Const_k(s_{f_j})$ contradicts with any of the requirement set R_{set} then

$m_i = m_i < v_{i_{(n,m)}}, s_{f_j} >$

$S_{mod} = \emptyset$

For each $m_i \in m_{set}$ do

For each $s_{f_j} \in S_{lib}(v_i)$ do

For each $p_{out_k}\left(s_{f_j}\right) = \emptyset$ do

For each $m_l \in m_{set}$ do

For each $s_{f_m} \in S_{lib}(v_l)$ do

```
                                        For each p_in_n(s_f_m) = ∅ do
                                            p_out_k(s_f_i) → p_in_n(s_f_k)
            For each p_in_k(s_f_j) = ∅ do
                For each m_l ∈ m_set do
                    For each s_f_m ∈ S_lib(v_l) do
                                        For each p_out_n(s_f_m) = ∅ do
                                            p_out_k(s_f_i) → p_in_n(s_f_k)
    S_mod = S_mod ∪ s_f_i
Return S_mod
```

6.7 Case Study

The case study of an electro-mechanical conveyor system is used now to demonstrate the utilization of the function-to-simulation model synthesizer. Conveyor systems are widely used in fish processing machines in order to provide an intermittent motion for the fish during transportation, inspection, and processing. The intermittent motions are commonly used in indexing and sequencing [155]. Indexing motion is mostly needed in the industry to move products in step-by-step patterns and processing by a stationary device. This motion allows an automated line to stop the product in a predetermined location for defined time, in order to complete a specific task within a specified time period while the product is kept stationary. The assigned tasks and operations performed on the object include cutting, inspection, and assembly. For the particular case study that is provided in this section, the motion profile is planned in such a way that a cutter is used to cut the head of a fish, while the fish is kept stationary. Therefore, different kinematic behavioral concepts are developed for the indexing system, which present how to capture the behavior aspect of the FM through multi-domain simulation model representations. The present case study is not intended to provide a detailed kinematic analysis but, rather, develop different design configurations through the synthesis of simulation components.

6.7.1 Synthesis of Simulation Models

The simulation models enable domain engineers to validate their sub-systems in software- and hardware-in-the-loop simulations, and to perform performance comparisons across various architectures. Since the synthesizer is based on the functional model, the functional description is essential in this process, where it describes the product from the functional point-of-view. Two functional sub-systems are considered: the driving system and the motion mechanism system. These sub-systems are represented as shown in Fig. 6-5, according to definition 6-1.

$$F_{sub_1} = \left\{ \begin{bmatrix} v_{1.1}{}_{(p_{1n_{1.1}}, p_{1n_{1.2}}, p_{1n_{1.3}}, p_{1m_{1.1}}, p_{1m_{1.2}}, p_{1m_{1.3}})} \\ v_{1.2}{}_{(p_{1n_{2.1}}, p_{1n_{2.2}}, p_{1n_{2.3}}, p_{1m_{2.1}}, p_{1m_{2.2}}, p_{1m_{2.3}})} \\ v_{1.3}{}_{(p_{1n_{3.1}}, p_{1n_{3.2}}, p_{1n_{3.3}}, p_{1m_{3.1}}, p_{1m_{3.2}}, p_{1m_{3.3}})} \\ v_{1.4}{}_{(p_{1n_{4.1}}, p_{1n_{4.2}}, p_{1n_{4.3}}, p_{1m_{4.1}}, p_{1m_{4.2}}, p_{1m_{4.3}})} \end{bmatrix}, \begin{bmatrix} L_{v_{1.1}} = Convert_3 \\ L_{v_{1.2}} = Transfer \\ L_{v_{1.3}} = Acuate \\ L_{v_{1.4}} = Convert_2 \end{bmatrix}, \begin{bmatrix} e_{1.1}{}_{(v_2, v_{1.1})} \\ e_{1.2}{}_{(v_3, v_{1.2})} \\ e_{1.3}{}_{(v_{1.1}, v_{1.3})} \\ e_{1.4}{}_{(v_{1.2}, v_{1.3})} \\ e_{1.5}{}_{(v_{1.3}, v_{1.4})} \\ e_{1.6}{}_{(v_{1.4}, v_5)} \end{bmatrix},$$

$$\begin{bmatrix} L_{e_{1.1}} = Human\ Energy \\ L_{e_{1.2}} = Electrical\ Energy \\ L_{e_{1.3}} = Control\ Signal \\ L_{e_{1.4}} = Electrical\ Energy \\ L_{e_{1.5}} = Electrical\ Energy \\ L_{e_{1.6}} = Rotational\ Mechanical\ Energy \end{bmatrix}, \begin{bmatrix} s_{(e_{1.1})} = p_{m_2} \\ s_{(e_{1.2})} = p_{m_3} \\ s_{(e_{1.3})} = p_{1m_{1.1}} \\ s_{(e_{1.4})} = p_{1m_{2.3}} \\ s_{(e_{1.5})} = p_{1m_{3.3}} \\ s_{(e_{1.6})} = p_{1m_{4.3}} \end{bmatrix}, \begin{bmatrix} t_{(e_{1.1})} = p_{1n_{1.3}} \\ t_{(e_{1.2})} = p_{1n_{2.3}} \\ t_{(e_{1.3})} = p_{1n_{3.1}} \\ t_{(e_{1.4})} = p_{1n_{3.3}} \\ t_{(e_{1.5})} = p_{1n_{4.3}} \\ t_{(e_{1.6})} = p_{n_5} \end{bmatrix}, \begin{bmatrix} p_{1n_{1.1}} \\ p_{1n_{1.2}} \\ p_{1n_{1.3}} \\ p_{1n_{2.1}} \\ p_{1n_{2.2}} \\ p_{1n_{2.3}} \\ p_{1n_{3.1}} \\ p_{1n_{3.2}} \\ p_{1n_{3.3}} \\ p_{1n_{4.1}} \\ p_{1n_{4.2}} \\ p_{1n_{4.3}} \\ p_{n_5} \end{bmatrix}, \begin{bmatrix} p_{m_2} \\ p_{m_3} \\ p_{1m_{1.1}} \\ p_{1m_{1.2}} \\ p_{1m_{1.3}} \\ p_{1m_{2.1}} \\ p_{1m_{2.2}} \\ p_{1m_{2.3}} \\ p_{1m_{3.1}} \\ p_{1m_{3.2}} \\ p_{1m_{3.3}} \\ p_{1m_{4.1}} \\ p_{1m_{4.2}} \\ p_{1m_{4.3}} \end{bmatrix} \Bigg\}$$

$$F_{sub_6} = \left\{ \begin{bmatrix} v_{6.1}{}_{(p_{6n_{1.1}}, p_{6n_{1.2}}, p_{6n_{1.3}}, p_{6m_{1.1}}, p_{6m_{1.2}}, p_{6m_{1.3}})} \\ v_{6.2}{}_{(p_{6n_{2.1}}, p_{6n_{2.2}}, p_{6n_{2.3}}, p_{6m_{2.1}}, p_{6m_{2.2}}, p_{6m_{2.3}})} \end{bmatrix}, \begin{bmatrix} L_{v_{6.1}} = Convert_1 \\ L_{v_{6.2}} = Translate \end{bmatrix}, \begin{bmatrix} e_{6.1}{}_{(v_5, v_{6.1})} \\ e_{6.2}{}_{(v_8, v_{6.2})} \\ e_{6.3}{}_{(v_{6.1}, v_{6.2})} \\ e_{6.4}{}_{(v_{6.3}, v_7)} \end{bmatrix}, \right.$$

$$\left. \begin{bmatrix} v_{6.1}{}_{(p_{6n_{1.1}}, p_{6n_{1.2}}, p_{6n_{1.3}}, p_{6m_{1.1}}, p_{6m_{1.2}}, p_{6m_{1.3}})} \\ v_{6.2}{}_{(p_{6n_{2.1}}, p_{6n_{2.2}}, p_{6n_{2.3}}, p_{6m_{2.1}}, p_{6m_{2.2}}, p_{6m_{2.3}})} \end{bmatrix}, \begin{bmatrix} L_{v_{6.1}} = Convert\_1 \\ L_{v_{6.2}} = Translate \end{bmatrix}, \begin{bmatrix} e_{6.1}{}_{(v_5, v_{6.1})} \\ e_{6.2}{}_{(v_8, v_{6.2})} \\ e_{6.3}{}_{(v_{6.1}, v_{6.2})} \\ e_{6.4}{}_{(v_{6.3}, v_7)} \end{bmatrix}, \right\}$$

Figure 6-5: Driving system functional sub-system (top) and motion mechanism functional sub-system (bottom).

Each function v_i in the sub-functional model is represented with its associated ports p_i and the defined vocabulary L_{v_i}. Moreover, each flow e_i is expressed with the corresponding outgoing and incoming functions v_i, and defined vocabulary L_{e_i}. The ports p of each source s_{e_i} of a flow e_i must match the ports p_m of the outgoing function v_i. Similarly, the ports p of each target t_{e_i} of a flow e_i must match the ports p_n of the incoming function v_j. For example, the function $v_{1.4}$ is defined from FB $L_{v_{1.4}}$ as **Convert**: "*To change from one form of a flow [...] to another.*" Therefore, three incoming ports $p_{1_{n_{4.1}}}, p_{1_{n_{4.2}}}, p_{1_{n_{4.3}}}$, and outgoing ports $p_{1_{m_{4.1}}}, p_{1_{m_{4.2}}}, p_{1_{m_{4.3}}}$ are assigned. These ports can be compatible with any type of source or target flows. Notice that the synthesizer identifies the information stored in the simulation component, as shown in Fig. 6-6.

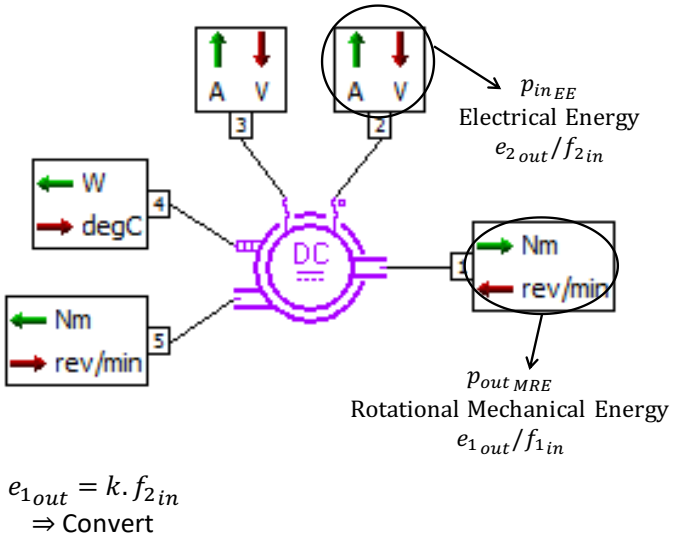


Figure 6-6: The simulation component of a DC motor.

First, in the simulation component s_i, each port p_i is examined, which represents the power conjugate complements. Thus, from table 6-1, the ports are defined according to the FB. Furthermore, the input, and output flows are determined with respect to the direction of the effort variable. The ports of the simulation component are then described as follows: $s_{i_{p_1}}$ expresses the

effort state variable **torque** leaving the port, and the flow state variable **angular velocity** entering the port. Hence, the port can be defined as a **Rotational mechanical energy** leaving the port $p_{out_{MRE}}$. Similarly:

$$s_{i_{p_2}} = p_{in_{EE}}$$

$$s_{i_{p_3}} = p_{in_{EE}}$$

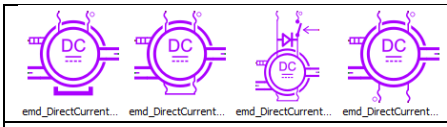
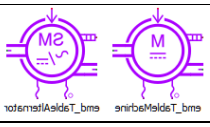
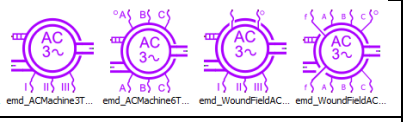
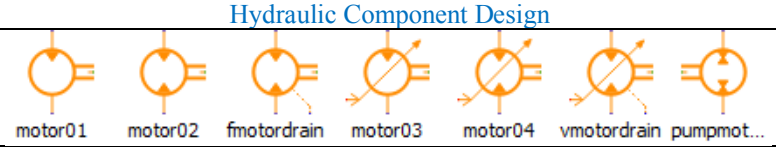
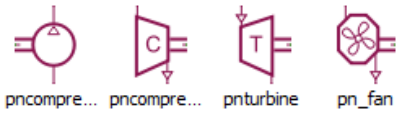
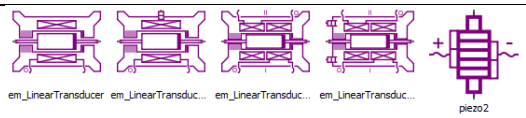
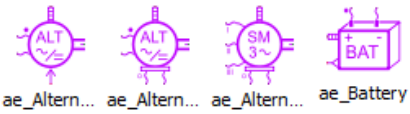
$$s_{i_{p_4}} = p_{in_{TE}}$$

$$s_{i_{p_5}} = p_{out_{MRE}}$$

The ratio of the output to the input *effort variable* is determined from equation (6-1), which dictates the embodied function, here, **Convert**. Full mappings between the function $v_{1.4}$ and the simulation component library s_{lib} is developed, and they are shown in table 6-3.

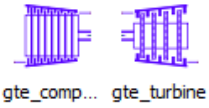
The targets of the incoming flows to the function $v_{1.4}$ should then match the input ports p_n of the simulation component family s_{lib_i}. Equivalently, the sources of the outgoing flows from the function $v_{1.4}$ should match the output ports p_m of the simulation component family s_{lib_i}. Here, the flow $e_{1.5_{(v_{1.3},v_{1.4})}} = Elctrical\ Energy$ is entering the function $v_{1.4}$ through the port $t_{(e_{1.5})} = p_{1_{n_{4.3}}}$. On the other hand, the flow $e_{1.6_{(v_{1.4},v_5)}} = Rotational\ Mechanical\ Energy$ is leaving the function $v_{1.4}$ through the port $s_{(e_{1.6})} = p_{1_{m_{4.3}}}$. Therefore, the mapping contains only the simulation component families, with their incoming and outgoing ports matching with $p_{1_{n_{4.3}}} = Electrical\ Energy$ and $p_{1_{m_{4.3}}} = Rotational\ Mechanical\ Energy$.

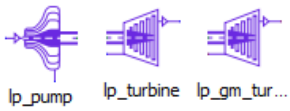
Table 6-3: Mapping generation of the function convert from the simulation component library.

		
s_{lib_1} = DC Machines	s_{lib_2} = Functional Devices	s_{lib_3} = Synchronous Machines
Electro Motors and Drives		
		
s_{lib_4} = Pumps		
Hydraulic Component Design		
		
s_{lib_5} = Pumps, Motors		
Thermal-Hydraulic		
		
s_{lib_6} = Specific components		
Thermal-Hydraulic Component Design		
		
s_{lib_7} = Compressors and Motors		
Pneumatic		
		
s_{lib_8} = Electro-mechanical Actuator family		
Electro-mechanical		
		
s_{lib_9} = Generators		
Automotive Electrical		
		
$s_{lib_{10}}$ = Compressor, Turbine, and Pump		
Two-phase Flow		


$s_{lib_{11}}$ = Specific components
Pneumatic Component Design

$s_{lib_{12}}$ = Pump, Compressor, Turbine
Gas Mixture

$s_{lib_{13}}$ = Electrodes, Electroyles
Electrochemistry

$s_{lib_{13}}$ = Compressors, Turbines
Gas Turbine

$s_{lib_{14}}$ = Power Generator
Aeronautics and Space

$s_{lib_{15}}$ = Liquid propulsion

At this point, the simulations components $\boldsymbol{s_{lib_1}}$**= DC Machine**, $\boldsymbol{s_{lib_2}}$**= Functional Devices**, $\boldsymbol{s_{lib_3}}$**= Synchronous Machines**, and $\boldsymbol{s_{lib_9}}$**= Generators** are selected. On the left of Fig. 6-7, the mappings of the simulation components $s_1 \in s_{lib_3}$(left) and $s_6 \in s_{lib_{10}}$ (right), with the function $v_{1.4}$ are shown.

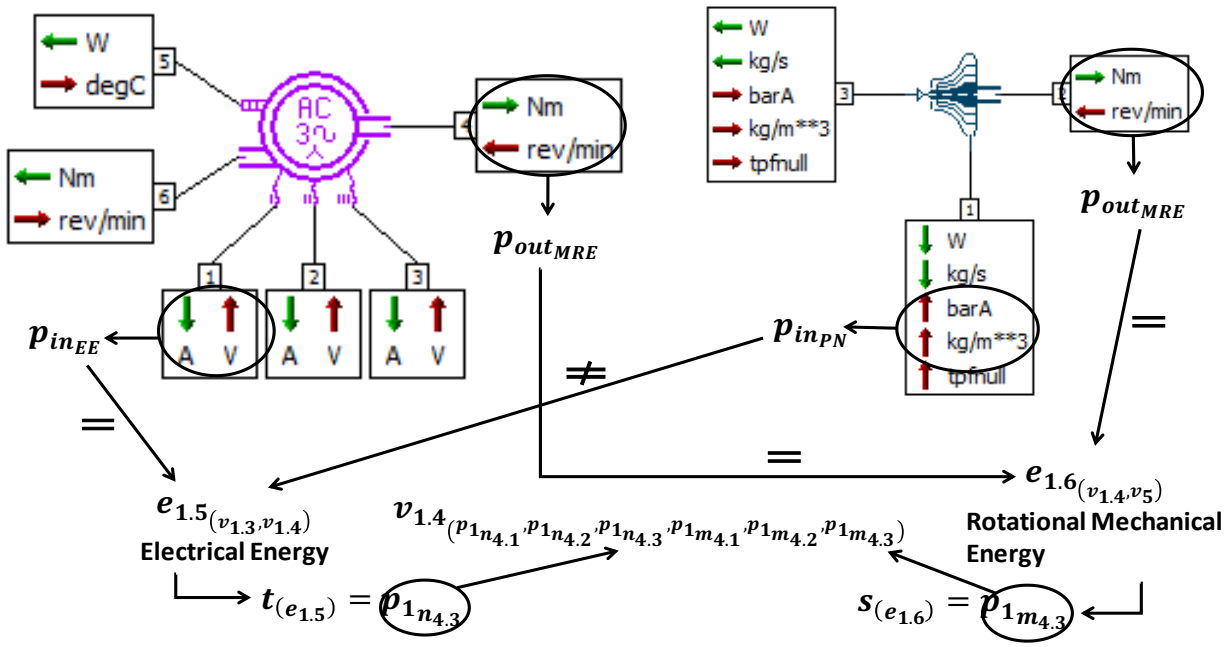


Figure 6-7: Port mappings of two of the simulation components - Synchronous Machines (left), and Liquid Propulsion (right) - with the function *convert*.

6.7.2 Kinematic Behavior

In the present case study, different solutions are developed using the synthesizer. The sub-functional model F_{sub_6} describes the kinematic behavior of the motion mechanism of the conveyor system, as shown in Fig. 6-5 (bottom). The function $v_{6.1}$= *Convert* has an incoming flow $e_{6.1}$= *Rotational Mechanical Energy* and an outgoing flow $e_{6.3}$= *Translational Mechanical Energy*. Similarly, the function $v_{6.2}$= *Translate* has two incoming flows, $e_{6.2}$= *Solid Material* and $e_{6.3}$= *Translational Mechanical Energy*, and one outgoing flow $e_{6.4}$= *Solid Material*. The expected kinematic behavior of the sub-model is a continuous cycle motion that goes through periods of acceleration, deceleration, and dwell, with the assumption that the input rotational motion speed is kept constant. The net energy of the indexer is also estimated to be zero, since the energy required by the system in the acceleration period should, theoretically, equal to the energy removed by the

system in the deceleration period, assuming there is no frictional loss. Fig. 6-8 shows three Amesim simulation models generated based on the functional design intent.

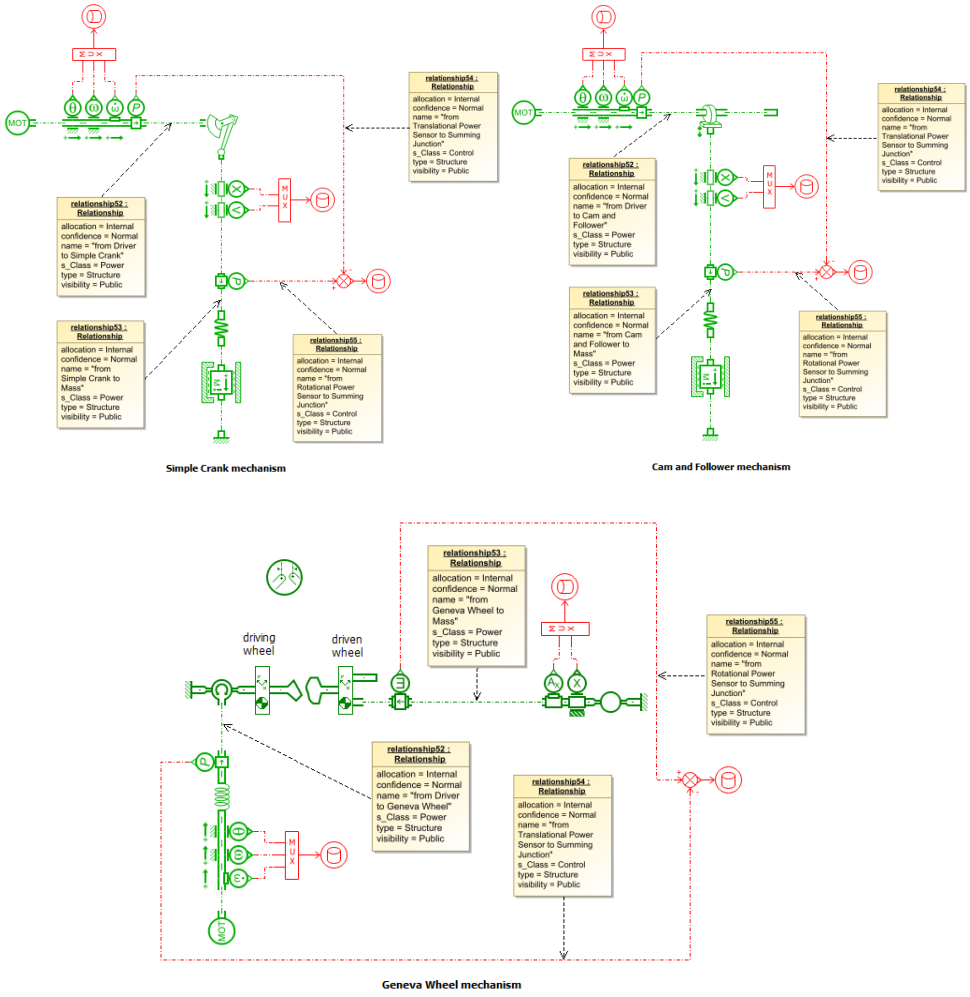


Figure 6-8: Generated Amesim simulation models of Simple Crank (top left), Cam and follower (top left), and Geneva Wheel (bottom).

These models represent the conversion of the rotational motion into linear intermittent motion through three different mechanisms: an ideal crank, a cam and follower, and a Geneva wheel. These models assume there is no frictional losses or inertia, and they have almost perfectly efficient motion transformation. Also, in the case of the cam and follower, the contact between the cam with the follower is ideal, providing continuous contact. These kinematic concepts meet the

target functionality with its incoming and outgoing flows. In two scenarios shown in Fig.6-8 upper left and upper right, springs are used to accelerate and decelerate the drive shaft. When the spring is fully pressed, the potential energy is stored. Its release causes a motion acceleration. When the spring starts to extend, it decelerates the motion until the spring is fully extended, and this cycle of energy exchange continues. The irregularity of the mechanical element, specifically the cam, can provide the required intermittent motion of the indexer. In Fig. 6-8 (bottom), the circular motion of the Geneva Wheel is represented by a rotation of two rotating bodies, in which one is the driving body and the other is the driven body. The driven body has four slots, where a pin in the driving body is to be inserted inside the slots, one at a time, causing the intermittent motion. Fig. 6-9 shows the dynamic motion behavior of the indexer for the indicated three concepts.

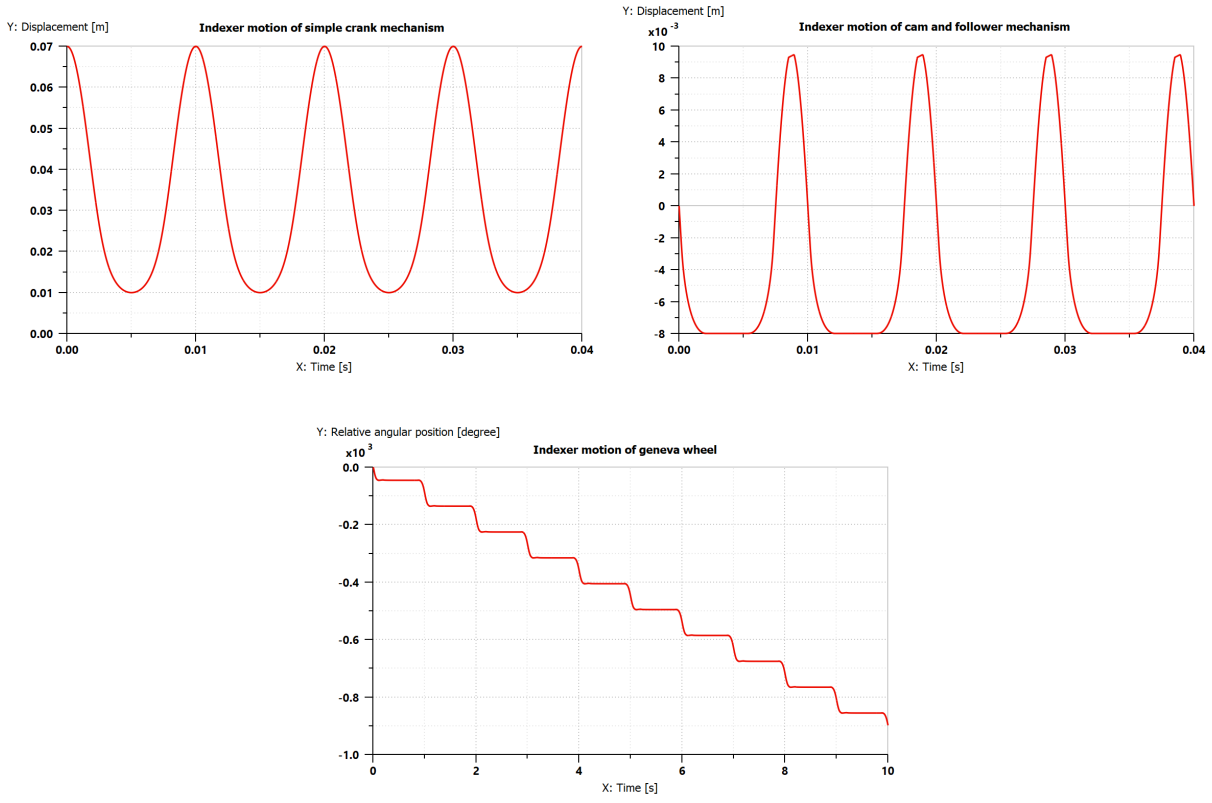


Figure 6-9: The dynamic motion behaviors of the indexer for the three concepts.

When evaluating these behaviors against the expected motion behavior, we find the motion profile of the indexer with the simple crank mechanism has no dwell motion, where the predetermined operation is minimal. However, a sufficient dwell period can be obtained with the cam and follower mechanism, and the Geneva wheel mechanism. The motion profile of the Geneva wheel mechanism shows the circular motion of the driven body in degrees, which exhibits a stair-like motion behavior. The net energy of the system is shown in Fig. 6-10.

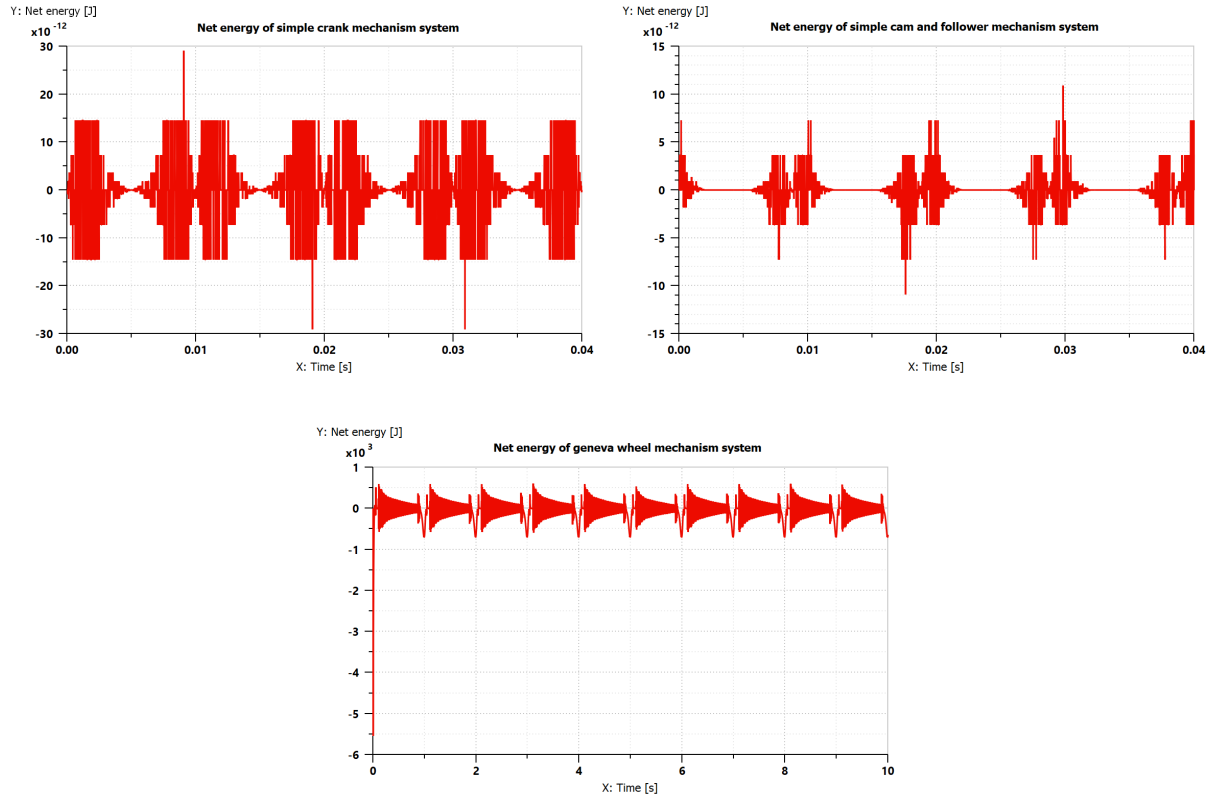


Figure 6-10: The net energy of the indexer for the three concepts.

These obtained patterns represent the net energy consumptions of the three models. In the case of the simple crank mechanism, and the cam and follower mechanism, it should be noted there is almost an identical net energy consumption in the motion of the indexer in one direction and also in the opposite direction. However, in the case of the Geneva wheel, we observe that there is a higher energy consumption at the beginning of the operation and the energy consumption then

decreases. This results from the extra energy that is required to move the driven body at the start of the motion. The simulation results of the generated conceptual models provide a preliminary insight into their dynamic behavior, which corresponds to the first evaluation of these concepts.

Chapter 7: Process Design Methodology Using the Mechatronic Design Quotient (MDQ)

7.1 Introduction

The applications of multi-criteria optimization have roots in many fields such as business, management, economics, logistics, and engineering. A key objective in them is to obtain optimal decisions in the presence of two or more possibly conflicting objectives and possibly involving multiple physical domains. The term Pareto optimality concerns an optimal solution that considers all objectives of optimization simultaneously.

An important engineering application of multi-criteria optimization is the design of physical systems, where the design evaluation requires taking into account the correlation between various system requirements. For some practical, and technical considerations, the traditional, sequential design approach for multi-domain (multi-physics) systems separates the overall system into several sub-systems that are treated sequentially according to the different domains. Therefore, the optimization process is conducted using only one perspective at a time. This has serious drawbacks, particularly due to the dynamic interactions (or coupling) that exist among different domains [4]. For example, Valdez et al. [156] did a comparison study between concurrent and sequential optimization methodologies for the design of serial manipulators with the objectives to find the best geometrical and control parameters. Three metaheuristics were compared. Specifically, Ominioptimizer, BUMDA, and CMA-ES were used for three types of optimization problems: Statics, Kinematics, and Dynamics, with the focus on comparing a dynamic optimization model. They observed that the sequential method finds optimum lengths for the static model, which are not optimal for the dynamic model; therefore, the overall design is not optimal

even if the optimal control gains are found by modifying the arm length leads or modifying the model masses and the manipulator dynamics.

Hence, a high potential exists for developing integrated (concurrent) design and evaluation schemes for mechatronic systems, resulting in lower efficiency and cost, and improved component compatibility. Furthermore, the performance quality of the design solutions in the early stages of design development of a mechatronic system should be evaluated in an integrated manner, where the presence of some incomplete information, particularly concerning different physical domains, is recognized.

7.2 System Model Evaluation

Determining the functionality of a design should be objectively evaluated by comparing the performance capabilities of the system with the corresponding system specifications and customer requirements. The performance description of the system should incorporate product components and their behavior. At this point of the CDDP, the concept analysis and evaluation sub-process take place on the right side of the V-model, as shown in Fig 3-2, for the evaluation of the CIM. The degree of detail of the CIM is abstract, due to the incorporation of instantiations of generalized integrated components from different disciplines.

The V-model sub-process of the CDDP incorporates virtual integration and simulation in an early phase to enhances the concept analysis and evaluation capabilities, and incorporate different disciplines with respect to their functional and structural specifications, as shown in Fig. 4-1. After the development of the integrated system, where different structures are presented as conceptual design solutions – see Chapter 6 -- further understanding of each solution is required. Each design solution is described by a set of interrelated instantiated physical forms of objects. The *behavior variables* of the considered design solution $Y(t)$ is a function of *environment*

variables $U(t)$ and *design parameters/variables* $X(t)$, as shown in Fig. 7-1. Performance evaluation through modeling and simulation involves describing the dynamic behavior of the system computationally, while considering the interactions of the interrelated components.

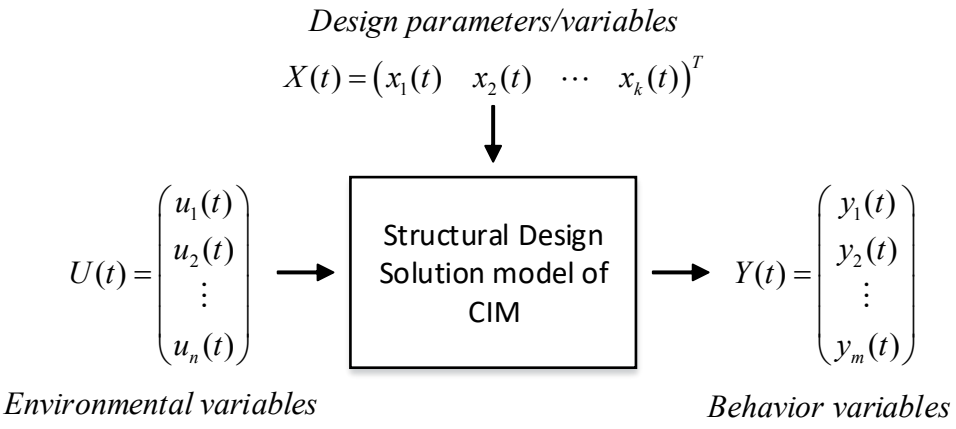


Figure 7-1: : Conceptual design solution model.

7.2.1 Individual Performance Indicator

In order to present the performance of a system, indicators are required that represent the behavior of the time-varying system variables. These variables can be characterized using system parameters that describe generalized constant properties such as max/min values, averaging, and variances, over a defined period of time [3]. The use of mathematical concepts is necessary to appropriately represent a wide range of variations of these properties over entire time histories. The required parameters can be experimentally chosen later, during the *system validation*. An underlying mathematical concept is expressed here as an **individual performance indicator (IPI)**, where the derived parameters of a conceptual solution are predicted. These parameters are represented using a generalized distance metric, which assigns a non-negative value to all pairs of elements in a metric space [3] as follows:

Definition 7-1: Individual performance indicator

Let Y be any set of behavior variables of a system. An individual performance indicator is a map $I: Y \times Y \rightarrow \mathbb{R}$, and it has the following properties of $y_1, y_2, y_3 \in Y$:

$$\textbf{Identity metric: } \boldsymbol{I(y_1, y_2) = 0 \Leftrightarrow y_1 = y_2} \tag{7-1}$$

$$\textbf{Symmetry: } \boldsymbol{I(y_1, y_2) = I(y_2, y_1)} \tag{7-2}$$

$$\textbf{Triangle inequality: } \boldsymbol{I(y_1, y_2) \leq I(y_1, y_3) + I(y_3, y_2)} \tag{7-3}$$

Therefore, it can be verified that:

$$\boldsymbol{I(y_1, y_2) \geq 0} \tag{7-4}$$

7.2.2 Aggregated Performance Indicator

Non-functional requirements can be defined to indicate how well the system should perform or how the system should behave. They determine the design criteria or constraints on the system functions or behaviors. These criteria or constraints are identified by carrying out a requirement analysis based on the customer needs and a thorough understanding of the design tasks. Some design sub-functions might share the same criteria or constraints and others might not, depending on the design goals. Typical design criteria or constraints contain cost, efficiency, reliability, size, complexity, speed, weight, payload, and so on. Some of these measures may take an analytical form while some others may be qualitative and fuzzy and may involve human perception. The interaction may be present among these criteria or constraints. In order to find relationships and correlations between the design criteria or constraints, multi-criteria decision making that may involve both qualitative and quantitative criteria is required. The next step after obtaining the non-functional requirements is to calculate the weights for different criteria, as a means to evaluate the conceptual design solutions.

These criteria are, then, related to some of the variables/components of the considered system. an engineer with the necessary expertise and experience will properly address and

determine these relationships. Also, the design process methodology might dictate the kind of relationship the designer is required to follow. Specifically, some criteria are associated with the input parameters, where the design specification can be determined accordingly. Another form of this association is the selection of the design components according to the criteria, where these components come with their parametric values [92], [112], [157], [158]. Usually, these input values are stationary and, therefore, do not represent the dynamic behavior of the system. On the other hand, in [159], the design criteria were associated with the behavior variables of the dynamic system. The advantage of this association is that the interactions between different multidisciplinary components/sub-systems are taken into account.

Mechatronic Design Quotient (MDQ)

Mechatronic Design Quotient (MDQ) was first proposed by de Silva [4], [108]. It is an aggregated performance indicator for the evaluation of multi-domain (Multiphysics) and multi-criteria systems possibly involving both qualitative and quantitative considerations, which is utilized for the concurrent (integrated) design of mechatronic systems to optimize the overall design, using unified (analogous) approaches for various physical domains. This indicator is also used as an evaluation scheme of different design alternatives of different proposed conceptual solutions that are generated in the conceptual design development process.

The weights of the criteria express the degree of satisfaction of each one, and accordingly, a partial score would be assigned. For n criteria, and m constraints, the MDQ indicator aggregates the scores of each n criterion as:

$$\boldsymbol{MDQ(s) = A(w_1, w_2, \dots, w_n).\prod_{i=1}^{m} r_i(s)} \tag{7-5}$$

where A is an aggregator operator, s is a design solution, w_i is weight represented by a partial score between zero to one of the i^{th}criterion, and $r_i(s)$ is a function indicating whether a constraint has been met, which is equal to 1 if the i^{th}has been satisfied, and zero, otherwise.

For a finite set of criteria $C = \{c_1, c_2, \dots, c_n\}$ in a multicriteria evaluation of a design solution, a key consideration in the aggregation of these criteria is the interactions that may exist between them. The nonlinear Fuzzy integrals such as Choquet and Sugeno integrals have an advantage over the traditional method of linear weighted average, particularly because the fuzzy integrals can model the criteria interactions [51]. Therefore, not only the weighting of criterion is considered, but also for each subset of criteria, there are $g: 2^C \rightarrow [0,1]$ weighting factors that must satisfy:

$$\boldsymbol{g(\emptyset) = 0} \tag{7-6}$$

$$\boldsymbol{g(C) = 1} \tag{7-7}$$

$$\boldsymbol{if\ A \subseteq B \subseteq C, then\ g(A) \leq g(B) \leq 1} \tag{7-8}$$

Different types of interaction take place between these criteria. They are indicated in Table 7-1.

7.3 Approach Description

The non-functional requirements determine the design criteria or constraints that are imposed on the behavior of the system, which requires the understanding of the basic requirements of the design goals. The evolution of the design criteria formulation therefore encapsulates a wide range of features. For the mechatronic design of systems, some general objectives may include the following [160]:

- Component matching
- Efficiency

- Reliability
- Stability
- Accuracy

In practice, reasoning and requirement analysis are performed to incorporate or remove some criteria depending on their importance in the design problem.

Table 7-1: The types of relationships between different criteria.

Type of Interaction	Description	Relationship
Positive Correlation	Criteria i and j combined have weights higher than the weights of their interactions	$\boldsymbol{g(ij) < g(i) + g(j)}$ (7-9)
Negative Correlation	Criteria i and j combined have weights lower than the weights of their interactions	$\boldsymbol{g(ij) > g(i) + g(j)}$ (7-10)
Veto Effect	A good score in criterion i results in a bad global score	$g(A) \approx 0$ (7-11)
Pass Effect	A good score in criterion i results in a good global score	$g(A) \approx 1$ (7-12)
Substitutiveness	Criteria i and j are parallel and, therefore, they are independent and interchangeable	$(C) \approx \begin{Bmatrix} g(A \cup i) \\ g(A \cup j) \end{Bmatrix} < G(C \cup i \cup j), A \subseteq C \backslash i, j$ (7-13)
Complementarity	Criteria i and j are prerequisites for each other to achieve good global satisfaction	$g(C) < \begin{Bmatrix} g(A \cup i) \\ g(A \cup j) \end{Bmatrix} \approx G(C \cup i \cup j), A \subseteq C \backslash i, j$ (7-14)

Fig. 7-2 illustrates the process methodology developed in the present work for obtaining the MDQ for a given conceptual design solution. The process methodology consists of top-down and a bottom-up stages. The bottom-top stage is where the conceptual design solution model is established and, consequently, the behavior variables are developed. The top-down stage starts with conducting the need analysis and acquiring the customer requirements, where the design criteria are developed accordingly. Each of the design criteria is assigned a weighting factor, which

represents its degree of importance. Sugeno λ-measure [161], [162] is then calculated to address the correlation between different design criteria.

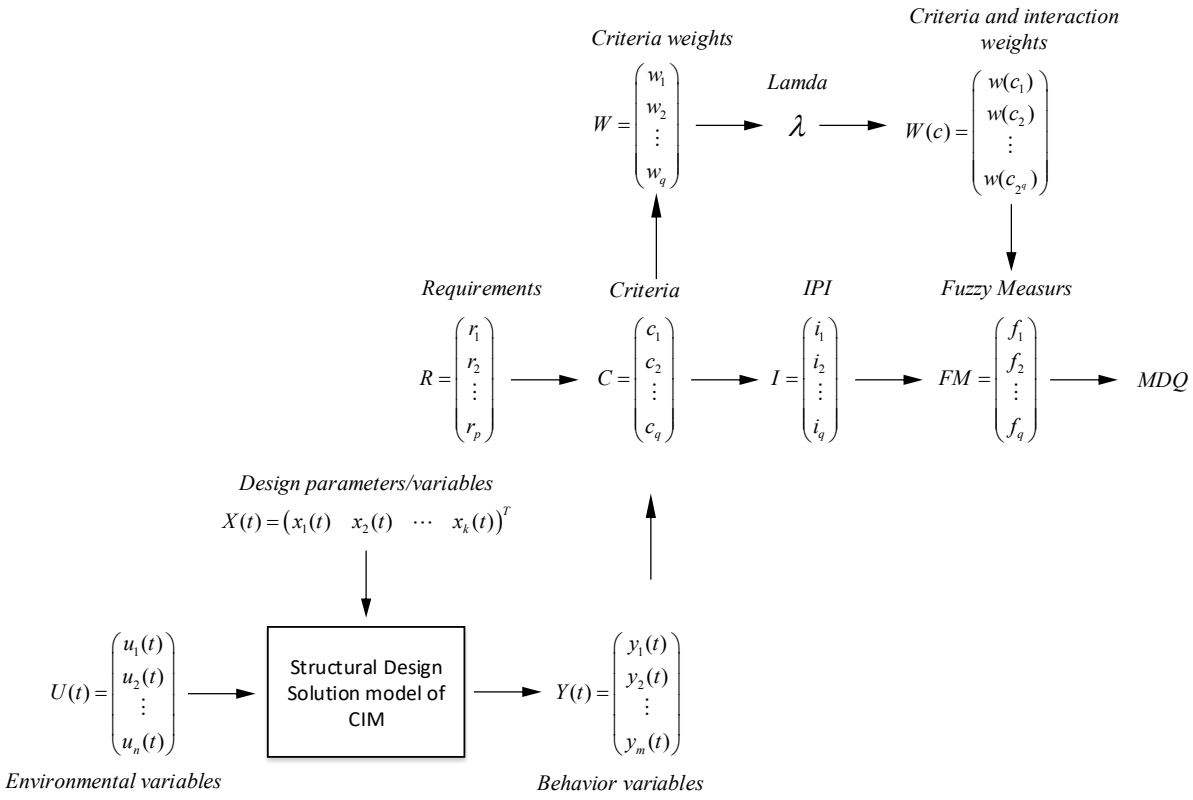


Figure 7-2: Process methodology for obtaining the Mechatronic Design Quotient (MDQ).

The formulation of the IPI is carried out next by linking the design criteria to the behavior variables using a mathematical concept. The process methodology continues to determine the aggregated performance indicator through the MDQ, where the weights of different criteria along with the weights of their interaction are integrated into the MDQ. The details of the proposed process methodology are given next.

For a given mechatronic system design, a vector of q customer requirements $C = \{c_1, c_2, \dots, c_q\}$ is given which is defined by the stakeholders. Also, a set of p design criteria $R = \{r_1, r_2, \dots, r_p\}$ is identified to fulfill the design requirements.

The evaluation of the performance of a conceptual solution is carried out for establishing the IPI. After identifying the design criteria vector, it is linked to the behavior variable vector $Y = \{y_1, y_2, \dots, y_m\}$, which represents different dynamic behaviors of the system. This association determines the reference values of the corresponding behavior, for example, max value, min value, and average.

Consequently, the performance metrics are decided according to the distance of each variable and the reference for the specific time period, using the following equation:

$$i_j = \underset{t\in[t_0,t_f]}{I}(y(t_j), y_{ref}) = \sum_{t=0}^{t=j}(1 - \frac{\left|y(t_j) - y_{ref}\right|}{y_{ref}})\prod_{k=1}^{n} s_k(t_j) \tag{7-15}$$

where:

i_j: performance indicator for the j^{th} criterion represented between zero to one

I: metrics vector of the time period between t_0 and t_f

$s_k(t_j)$: a function that represents a constraint violation, which is equal to 0 if the k^{th} constraint has been violated and 1 otherwise.

To appropriately describe the time-varying behavior, a sufficient number of observations is required - $y(t_j), j = 1, \dots n, n \gg 1$. The resulting IPI vector is then obtained as $I = \{i_1, i_2, \dots, i_q\}$.

Each of these criteria is assigned a weighting factor that is represented by a fuzzy measure, which is typically assigned by expert designers. However, to address the correlation between different criteria, fuzzy measures are also assigned for each subset of the criteria. For example, the fuzzy measures to be assigned to five criteria would involve 2^5=32 subset of criteria. The number increases exponentially when incorporating more criteria, which presents a challenge in assigning

these fuzzy measures. Therefore, the Sugeno λ-measure is introduced, which determines the subsets of fuzzy measures using a function.

For two subsets A and B of the universe X, such that: $A \subseteq X$ and $A \subseteq X$, Sugeno λ-measure is measured as:

$$\boldsymbol{g_\lambda(A \cup B) = g_\lambda(A) + g_\lambda(B) + \lambda g_\lambda(A) g_\lambda(B)} \tag{7-16}$$

And λ is obtained by:

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda g^i) \tag{7-17}$$

where:

λ is the Sugeno measure, in which $\lambda > -1$. That means: $\sum_{i=1}^{n} g^i > 1$ and g^i is a notation for the fuzzy measure $g(c_i)$

After obtaining the weights of different criteria and their interactions, they are integrated with the IPI using the MDQ, which is an aggregated indicator of the system performance with respect to their weighted criteria. The integration method used to address the overall performance is the *fuzzy integral* – see section 7.2.1.1. To compute the global score of each design solution, the following equation is used:

$$MDQ = \sum_{k=1}^{m} i_{\pi_k} (\mu(A_K) - \mu(A_{K-1})) \tag{7-18}$$

where:

i_k is the individual performance indicator of the k^{th} performance behavior

μ_k is the density value (fuzzy measure) of the k^{th} criterion

π is a permutation index to order the set, such that $i(c_{\pi_1}) \geq i(c_{\pi_2}) \geq \quad \geq i(c_{\pi_m})$

c_k is the k^{th} criterion

A_k gives the order of the information sources: $A_k = \{c_{\pi_1}, c_{\pi_2}, \quad , c_{\pi_m}\}$

A higher value of MDQ indicates a better level of performance of the design solution with reference to the design criteria.

7.4 Case Study

In this section, the approach developed in the present work will be applied to an electro-mechanical conveyor system that falls within the category of a mechatronic system. The system is used to transport fish from the feeding station to the cutting station (see Fig. 7-3). It is composed of three subsystems: an electric motor, a PID controller and a simple crank mechanism. The proposed process methodology for evaluating the conceptual design solutions is used to show the possible optimal design configuration, in which the time-to-market is decreased.

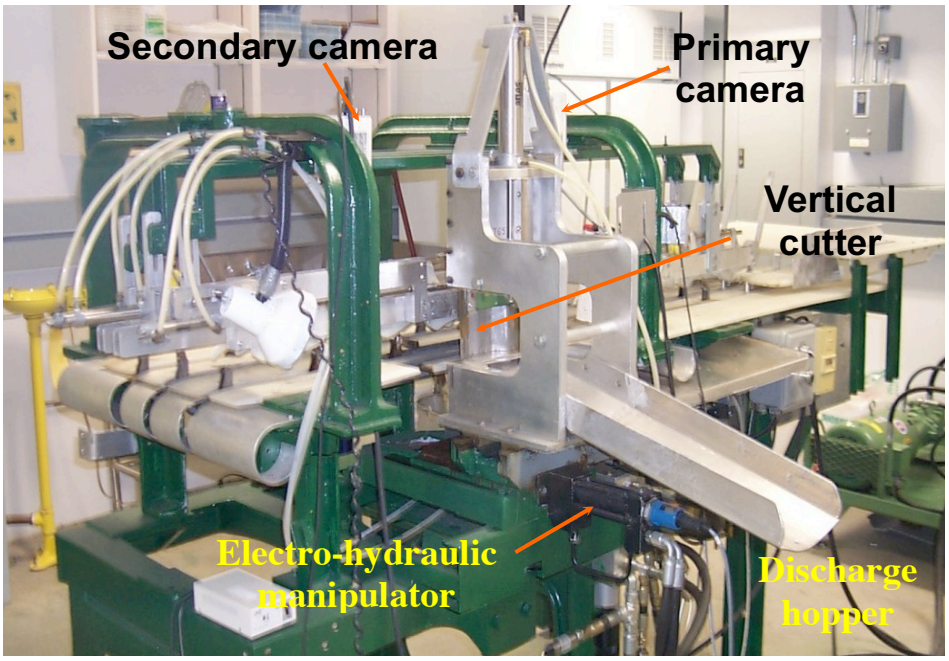


Figure 7-3: An industrial fish processing machine—Intelligent Iron Butcher.

The following are the design requirements.

- Efficiency: The average power losses are to be minimized
- Speed of response: The rotational velocity of the crank shaft is to reach the specified value of 10 $\frac{\text{rad}}{\text{sec}}$ in less than 1 second
- Reliability: The overshoot of the system should be reduced
- Stability: The settling time should be maintained at a minimum

One generated conceptual design solution is shown in Fig. 7-4. This system is modeled using Amesim – see Chapter 6. Therefore, a system of differential algebraic equations, which associate the behavior variables, the environment variables, and the design parameters are handled and solved by Amesim. The list of design variables is described in table 7-2.

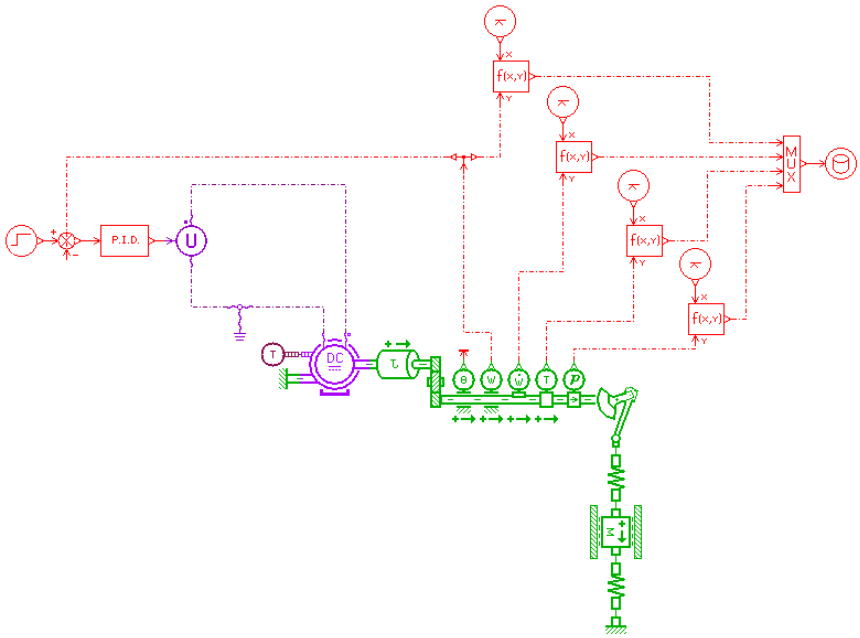


Figure 7-4: Generated Amesim simulation models of Simple Crank.

Table 7-2: The list of design variables.

Design parameter/variable	Name	Default value	Design domain
PID Gain constant	K	10	[10, 50]
PID integration constant (s)	T_i	20	[10, 50]
PID drivation constant (s)	T_d	5	[2, 10]
Motor inertia (kg.m^2)	I	50	[10, 60]
Armature winding resistance (ohm)	R_a	1	[1, 8]

The design variables X are:

$$\boldsymbol{X} = \{\boldsymbol{K}, \boldsymbol{T_i}, \boldsymbol{T_d}, \boldsymbol{I}, \boldsymbol{R_a}\} \tag{7-19}$$

The vector of behavior variables has been limited to the needed electric power P_w(W) and the rotational velocity of the motor shaft ω_m($^{\text{rad}}/_{\text{s}}$). Accordingly, the vector Y is defined by:

$$\boldsymbol{Y(t)} = \{\boldsymbol{\omega_m(t)}, \boldsymbol{P_w(t)}\} \tag{7-20}$$

A fuzzy measure is assigned to each subset of criteria. The remaining four criteria: "Efficiency", "Speed of Response", "Reliability", and "Stability" form $2^4 = 16$ subsets of criteria. Therefore, they make the criteria vector - $C = \{c_1$ "Efficiency", c_2 Speed of Response", c_3 "Reliability", c_4"$Stability$"}. These criteria need 16 fuzzy measures using the ordinary Choquet integral (two of them are self-evident: $w(\emptyset) = 0$, and $w(C) = 1$). The fuzzy measures are typically assigned by expert designers. The fuzzy measures used in this case study are obtained from [111] since the engineering system is the same and the criteria are similar. The fuzzy measures are as follows: $w(c_1)$ =0.35, $w(c_2) = 0.18$, $w(c_3) = 0.52$, $w(c_4) = 0.60$.

The Sugeno λ-measure is used to calculate the interaction between different criteria, as in equations (7-16) and (7-17). First, λ is calculated using equation (7-17):

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda g^i)$$

$$\lambda + 1 = (1 + \lambda(0.35)) \times (1 + \lambda(0.18)) \times (1 + \lambda(.52)) \times (1 + \lambda(0.60))$$

Four roots are found:

$\lambda_1 = 0$
$\lambda_2 = -0.8257$
$\lambda_3 = -5.588 + 2.97\mathrm{i}$
$\lambda_4 = -5.588 - 2.97\mathrm{i}$

Therefore, the correct choice is $\lambda = -0.8257$

Accordingly, using equation (7-16), we find the weights of the interactions, as:

$$w(c_1, c_2) = -0.0033,\ w(c_1, c_3) = -0.0274,\ w(c_1, c_4) = -0.0364,\ w(c_2, c_3) = -0.0072,$$

$$w(c_2, c_4) = -0.0096,\ w(c_3, c_4) = -0.0804.$$

Next, the IPI evaluation is performed, in which it shows the effect of the variations of the design parameters on the dynamic performance. Fig. 7-5 shows the output behavior of the system simulation, where the angular velocity (left), and the output power (right) are presented.

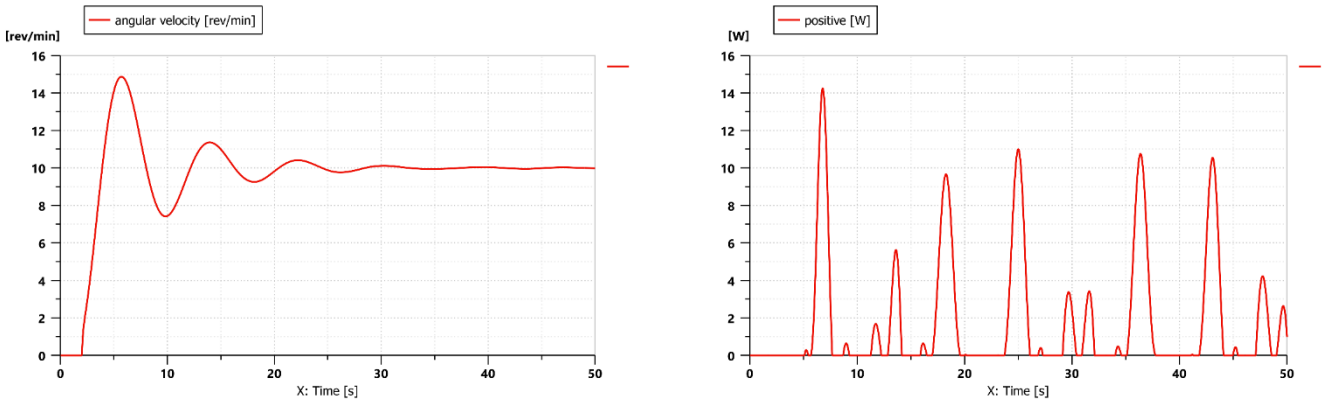


Figure 7-5: Angular velocity output (left), and power output (right).

By conducting the design exploration process, one can study the effect of the design paramters. IPI represents the individual performance indicator of each of the design criteria. It can be used for the analyzing the effect of the design variables on the criteria. This is able to give an insight into the design configuration and how they affect the system.

Fig. 7-6 shows how IPI can be used to show to variations of the efficiency criteria with the variations of the Motor inertia ($kg.m^2$) and with all the other design parameters constant. It can be seen that the efficiency of the system decreases as the value of the inertial increases. The efficiency was evaluated at different values of the motor inertia (10 **$kg.m^2$**, 20 **$kg.m^2$**, 30 **$kg.m^2$**, 40 **$kg.m^2$**, 50 **$kg.m^2$**, 60 **$kg.m^2$**).

Likewise, the speed of response criteria was evaluated at different variations of the PID gain constant (K) and with all the other design parameters constant. Fig. 7-7 shows that as the value of K increases, the speed of response evaluation increase as well. The different values of the gain constant that were computed are 10, 20, 30, 40, and 50. A penalty value was assigned to the output of the rotational velocity of the shaft if the speed shaft did not reach the specified value of 10 $^{rad}/_{sec}$. Therefore, the negative values indicate inability to comply with the condition.

Fig. 7-8, shows that the evaluation of the reliability criteria decreases as the value of the Armature winding resistance (ohm) increases, and with all the other deign variables kept constant. The values of the Armature winding resistance that were taken are 1 ohm, 4 ohms, 6 ohms, and 8 ohms. Finally, the stability criteria were evaluated with different variations of PID integration constant (T_i). As shown in Fig. 7-9, the increase of the value of the PID integration constant leads to a decrease of the evolution of the stability. The PID integration constant values were: 10, 20, 30, 40, and 50. With all the other design variables constant.

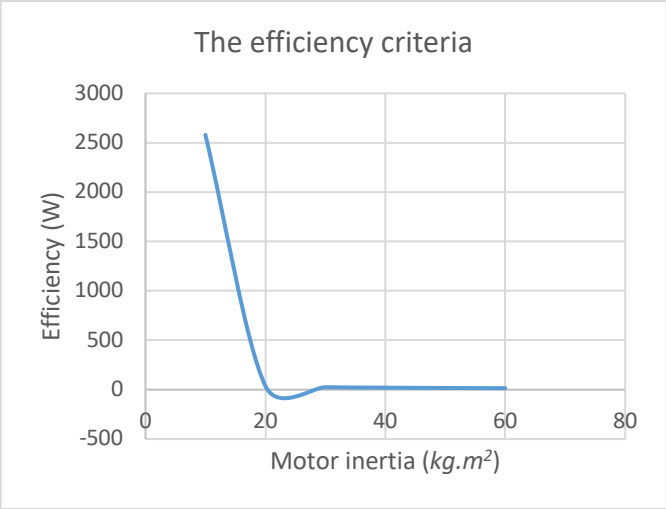


Figure 7-6: Efficiency variations as a function of motor inertia ($kg.m^2$)

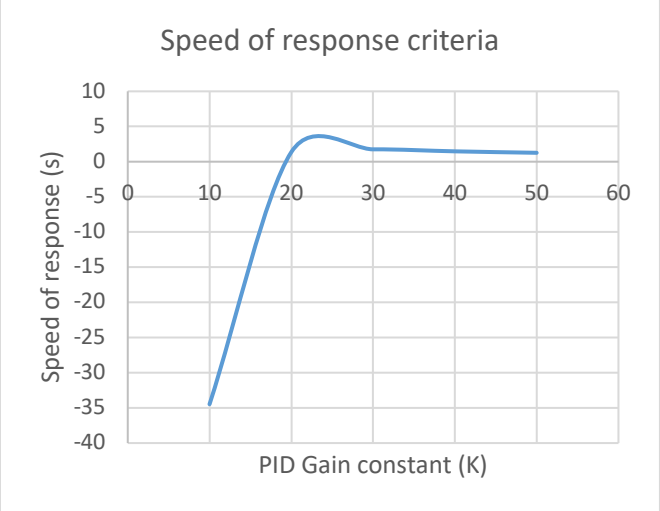


Figure 7-7: Speed of response variations as a function of PID gain constant (K)

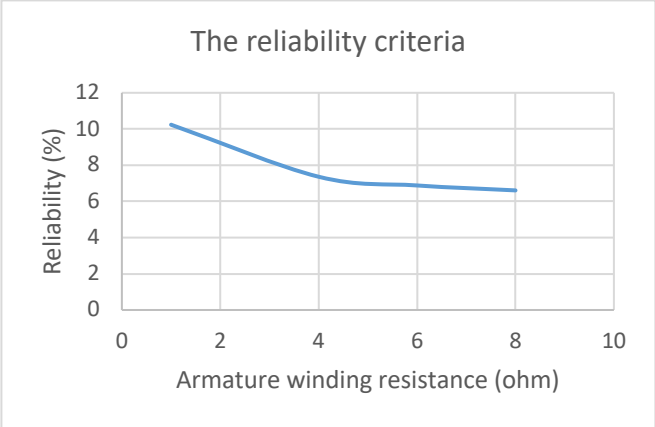


Figure 7-8: Reliability variations as a function of Armature winding resistance (*ohm*)

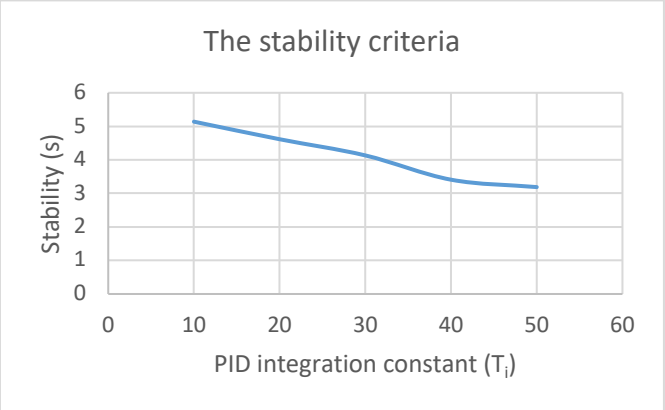


Figure 7-9: Stability variations as a function of PID integration constant (T_i)

Next, the MDQ evaluation is conducted, which represents a global evaluation index to all the presented criteria. It has an advantage of the ability to calculate the interactions between these criteria and take them into account when performing the evaluation. Each design alternative is formulated according various values of the design parameters. The following steps are to be followed:

1. The IPI is evaluated according to the corresponding values of the design variables.

2. Each design criterion is assigned a partial scores, which represent the importance of the design criteria.
3. Sugeno λ-measure can be used to calculate the partial scores between all the design criteria.
4. MDQ is used to aggregate all the partial scores of the all the criteria, as well as the partial score of between the criteria.

The result of the MDQ evaluation index is shown in Table 7-3.

Table 7-3: MDQ evaluation of design alternatives

	# 1	# 2	# 3	# 4	# 5
PID Gain constant	10	20	30	40	50
PID integration constant (s)	20	20	30	40	40
PID derivation constant (s)	50	40	30	20	10
Motor inertia (kg.m^2)	50	50	50	50	50
Armature winding resistance (ohm)	1	2	4	6	8
Efficiency (c1)	0.730	0.612	0.781	0.657	0.824
Speed of Response (c2)	0.554	0.467	0.512	0.471	0.587
Reliability (c3)	0.788	0.646	0.741	0.609	0.911
Stability (c4)	0.519	0.624	0.697	0.773	0.646
Global score	0.654	0.579	0.621	0.601	0.703

The best design configuration of the system is #5, which corresponds to design variables of PID gain constant of 50, PID integral constant of 40, PID derivative constant of 10, motor inertia of 50 kg.m^2, and armature winding resistance of 8 ohms.

This case study demonstrated a design exploration based on MDQ. The better configurations were identified; in which they help in the multi-disciplinary optimization of the mechatronic system.

Chapter 8: Conclusions and Future Work

8.1 Conclusions

This dissertation developed a design framework for a mechatronic system through the improvement of the Conceptual Design Development Process (CDDP) and the Conceptual Integrated Model (CIM). Two main types of problems were addressed: process-based problems and design data-related problems. System engineering approach was considered for the design and development of a mechatronic system in the conceptual design phase through Unified Modeling Language (UML) and System Modeling Language (SysML).

First, an integrated design process methodology was developed. In this process, a multi-layer V-model was proposed for a micro-level design process. The details of the characteristics of the design tasks and activities were presented. A case study of an industrial fish cutting machine was presented to demonstrate the application of the developed design methodology.

The details of the micro-level process model were explored, where the underlying organization of different design activities in the concept and modeling sub-process phase is defined. This work proposed *Interconnection Classifications* for the modeling of the communication data between different design activities within the macro-level process. A case study was presented, in which these interconnection classifications were implemented.

The functional modeling in the conceptual design phase was investigated. In particular, the implementation of the functional modeling and its library, specifically the Functional Basis, in SysML was proposed. It introduced the modeling of the functions and their ports, and the development of the library through the *Block Definition Diagram*. The implementation of the proposed functional modeling approach in SysML was demonstrated using a case study.

An algorithm was developed that described different modeling views and their activities in the concept and modeling phase. Also, the algorithm was developed to transform the functional model into a simulation model, computationally. This required a precise algorithm description of the requirement model, the functional model, the simulation model including its simulation library, and the simulation components, which was explored. The synthesizer principles and the algorithm were presented, which were illustrated using a case study.

Finally, methodology was proposed for the evaluation of conceptual design solutions of mechatronic systems. The underlying principles of the evaluation indicator were based on the Mechatronic Design Quotient (MDQ). Also, Sugeno λ-measure was proposed to compute the subsets of the fuzzy measures, to solve the exponential growth of the fuzzy measures that are assigned.

8.2 Possible Future Work

This dissertation developed a framework for the conceptual design of a mechatronic system. The main focus was the development of the CDDP and its activities. Further research may be done on the optimization of detailed design utilizing the same multi-layer V-model structure.

The modeling of the data exchange in the micro-level that takes place between different design activities of the micro-level may be further investigated. In particular, the decomposition of these data could be established and the granularity should be determined.

The utilization of artificial intelligence (AI) tools and machine learning approaches to create a framework that simultaneously analyzes and learns from successful product designs, should be investigated. In the same context, Machine Health Monitoring System (MHM) can be

employed to provide continuous improvements for the design process by identifying the design weaknesses. Also, methodology for automatic selection of components for the design library, may be developed.

Bibliography

[1] J. E. Carryer, R. M. Ohline, and T. W. Kenny, *Introduction to Mechatronic Design*, Prentice Hall, 2011.

[2] G. C. Onwubolu, *Mechatronics: Principles and Applications*, Elsevier Butterworth-Heinemann, 2005.

[3] K. Janschek, *Mechatronic Systems Design: Methods, Models, Concepts*, Springer Science & Business Media, 2011.

[4] C. W. de Silva, *Mechatronics: An Integrated Approach*, CRC Press, 2005.

[5] A. A. Cabrera M. J. Foeken, O. A. Tekin, K. Woestenenk, M. S. Erden, B. de Schutter, M. J. L. van Tooren, R. Babuska, F. J. A. M. van Houten, and T. Tomiyama, "Towards automation of control software: A review of challenges in mechatronic design," *Journal of Mechatronics*, vol. 20, no. 8, pp. 876–886, 2010.

[6] D. Shetty and R. A. Kolk, *Mechatronics System Design*, PWS Publishing company, 1997.

[7] K. Ehrlenspiel, A. Kiewert, U. Lindemann, and M. S. Hundal, *Cost-efficient Design*, Springer, 2007.

[8] E. Kroll, S. S. Condoor, and D. G. Jansson, *Innovative Conceptual Design: Theory and Application of Parameter Analysis*, Cambridge University Press, 2001.

[9] M. Abramovici and F. Bellalouna, "Integration and complexity management within the mechatronics product development," in the *Proceedings of the 14th CIRP conference on Life Cycle Engineering*, pp. 113–118, 2007.

[10] C. Zheng, M. Bricogne, J. Le Duigou, and B. Eynard, "Survey on mechatronic engineering: A focus on design methods and product models," *Journal of Advanced Engineering Informatics*, vol. 28, no. 3, pp. 241–257, 2014.

[11] I. Crnkovic, U. Asklund, and A. P. Dahlqvist, *Implementing and Integrating Product Data Management and Software Configuration Management*, Artech House, 2003.

[12] E. Von Hippel, "'Sticky information' and the locus of problem solving: implications for innovation," *Journal of Management Science*, vol. 40, no. 4, pp. 429–439, 1994.

[13] S. Thomke, *Learning by experimentation: Prototyping and testing*, C. Loch, S. Kavadias, *Handbook of New Product Development Management*, Elsevier Butterworth-Heinemann, p. 401, 2007.

[14] S. Minderhoud and P. Fraser, "Shifting paradigms of product development in fast and dynamic markets," *Journal of Reliability Engineering & System Safety*, vol. 88, no. 2, pp. 127–135, 2005.

[15] T. S. Schmidt and K. Paetzold, "Agilität als Alternative zu traditionellen Standards in der Entwicklung physischer Produkte: Chancen und Herausforderungen," in the *Proceedings of the 27th Symposium Design for X (DFX),* pp. 5-6*,* Jesteburg, Germany, October 2016.

[16] D. G. Ullman, *The Mechanical Design Process*, *McGraw-Hill Science/Engineering/Math*, 2002.

[17] R. Sell, *Model Based Mechatronic Systems Modeling Methodology in Conceptual Design Stage*, PhD Dissertation, Tallinn University of Technology, 2007.

[18] M. Eigner, T. Gilz, and R. Zafirov, "Interdisciplinary product development-model based systems engineering," *PLMportal*, 2017, [Online]. available: http://www.plmportal.org/en/research-detail/interdisciplinary-product-development-model-basedsystems-engineering.html [Accessed: December 13, 2018].

[19] L. E. Hart, "Introduction to model-based system engineering (MBSE) and SysML," in *Delaware Valley International council on Systems Engineering Chapter Meeting*, 2015.
[20] S. Wölkl and K. Shea, "A computational product model for conceptual design using SysML," in the *Proceedings of ASME 29th Computer and Information in Engineering Conference,* 2009.
[21] Y. Nemoto, F. Akasaka, and Y. Shimomura, "A knowledge management method for supporting conceptual design of product-service systems," in the *Proceedings of ASME 18th Design for Manufacturing and the Life Cycle Conference,* 2013.
[22] B. Beihoff, C. Oster, S. Friedenthal, C. J. J. Paredis, D. Kemp, H. Stoewer, D. Nichols, and J. Wade, "A world in motion: Systems engineering vision 2025," *International Council on Systems Engineering,* 2014.
[23] B. Kruse, *A Library-Based Concept Design Approach for Multi-Disciplinary Systems in SysML*, PhD Dissertation, ETH Zurich, 2017.
[24] A. Qamar, *An Integrated Approach Towards Model-based Mechatronic Design*, KTH Royal Institute of Technology, 2011.
[25] M. L. Griss, "Software reuse architecture, process, and organization for business success," in the *Proceedings of the 8th Israeli Conference on Computer Systems and Software Engineering*, pp. 86–89 1997.
[26] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, and K. L. Wood, "A functional basis for engineering design: Reconciling and evolving previous efforts," *Journal of Research in Engineering Design*, vol. 13, no. 2, pp. 65–82, 2002.
[27] Siemens PLM Software, "Simcenter - System simulations solutions," *Simcenter Amesim Libraries*, 2018.
[28] T. Kurtoglu, *A Computational Approach to Innovative Conceptual Design,* PhD Dissertation, The University of Texas at Austin, 2007.
[29] C. W. de Silva, "Sensory information acquisition for monitoring and control of intelligent mechatronic systems," *International Journal of Information Acquisition*, vol. 1, no. 01, pp. 89–99, 2004.
[30] N. P. Suh, *Axiomatic Design: Advances and Applications*, Oxford University Press, 2001.
[31] ISO 9000, "Quality management systems - Fundamentals and vocabulary," *European Standard*, 2005.
[32] G. Pahl and W. Beitz, *Engineering Design: A Systematic Approach,* Springer Science & Business Media, 2007.
[33] H. R. Buhl, *Creative Engineering Design,* Wiley-Blackwell, 1960.
[34] E. V. Krick, *An Introduction to Engineering and Engineering Design*, Wiley & Sons, Inc., 1969.
[35] K. N. Otto, and K. L. Wood, *Product Design: Techniques in Reverse Engineering and New Product Development,* Prentice Hall, 2001.
[36] K. T. Ulrich, *Product Design and Development*. McGraw-Hill Education, 2003.
[37] W. Beitz and G. Pahl, *Engineering Design: A Systematic Approach,* Springer Science & Business Media, 1996.
[38] N. P. Suh, *Axiomatic Design as A Basis for Universal Design Theory*, *Shaker Verlag*, pp. 3–24, 1998.
[39] B. S. Blanchard, *System Engineering Management*, John Wiley & Sons, 2004.

[40] J. M. Torry-Smith, N. H. Mortensen, and S. Achiche, "A proposal for a classification of product-related dependencies in development of mechatronic products," *Journal of Research in Engineering Design*, vol. 25, no. 1, pp. 53–74, 2014.
[41] C. Haskins, K. Forsberg, M. Krueger, D. Walden, and D. Hamelin, "Systems engineering handbook," *International Council on Systems Engineering,* 2006.
[42] J. Dewey, *How we think*, Dover Publications, 1997.
[43] T. Weilkiens, *Systems Engineering with SysML/UML: Modeling, Analysis, Design*, Elsevier, 2011.
[44] B. W. Boehm, *Software Engineering Economics*, Prentice-hall, vol. 197, 1981.
[45] B. W. Boehm, "A spiral model of software development and enhancement," *IEEE Transactions on Computers*, vol. 21, no. 5, pp. 61–72, 1988.
[46] K. Forsberg, and H. Mooz, "System engineering for faster, cheaper, better," in *International Council on Systems Engineering Symposium*, vol. 9, pp. 924–932, 1998.
[47] P. B. Crosby, *Quality is Free: The Art of Making Quality Certain*, McGraw-Hill, 1979.
[48] VDI Guideline, "2221: Systematic approach to the design of technical systems and products," *Verlag des Vereins Deutscher Ingenieure*, vol. 2221, 1993.
[49] J. Jänsch and H. Birkhofer, "The development of the guideline VDI 2221-the change of direction," in the *Proceedings Design, the 9th International Design Conference*, Dubrovnik, Croatia, 2006.
[50] VDI-Gesellschaft, "Systematical development of devices controlled by microelectronics," The Association of German Engineers VDI, 1994.
[51] H. M. van Brussel, "Mechatronics-A powerful concurrent engineering framework," *IEEE/ASME Transactions on Mechatronics*, vol. 1, no. 2, pp. 127–136, 1996.
[52] R. Isermann, "Modeling and design methodology for mechatronic systems," *IEEE/ASME Transactions on Mechatronics*, vol. 1, no. 1, pp. 16–28, 1996.
[53] J. Bathelt, A. Jonsson, C. Bacs, S. Dierssen, and M. Meier, "Applying the new VDI design guideline 2206 on mechatronic systems controlled by a PLC," in the *Proceedings ICED, the 15th International Conference on Engineering Design*, p. 2601, 2005.
[54] J. Gausemeier and S. Moehringer, "New guideline VDI 2206 - A flexible procedure model for the design of mechatronic systems," in the *Proceedings ICED, the 14th International Conference on Engineering Design, 2003*.
[55] V. S. Vasić and M. P. Lazarević, "Standard industrial guideline for mechatronic product design," *FME Transactions*, vol. 36, no. 3, pp. 103–108, 2008.
[56] D. Hofmann, M. Kopp, and B. Bertsche, "Development in mechatronics-Enhancing reliability by means of a sustainable use of information," in the *Proceedings of IEEE/ASME International Conference, Advanced Intelligent Mechatronics (AIM)*, pp. 1263–1268, 2010.
[57] J. Gausemeier, R. Dumitrescu, S. Kahl, and D. Nordsiek, "Integrative development of product and production system for mechatronic products," *Journal of Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 4, pp. 772–778, 2011.
[58] G. Barbieri, C. Fantuzzi, and R. Borsari, "A model-based design methodology for the development of mechatronic systems," *Journal of Mechatronics*, vol. 24, no. 7, pp. 833–843, 2014.
[59] R. Nattermann and R. Anderl, "The W-model-Using systems engineering for adaptronics," *Journal of Procedia Computer. Science*, vol. 16, pp. 937–946, 2013.

[60] J. Lefèvre, S. Charles, M. Bosch-Mauchand, B. Eynard, and E. Padiolleau, "multidisciplinary modeling and simulation for mechatronic design," *Journal of Design Research*, vol. 9, pp. 127-144, 2012.
[61] J. Fisher, "Model-based systems engineering: A new paradigm," *Insight*, vol. 1, no. 3, pp. 3–16, 1998.
[62] J. A. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.
[63] N. A. Tepper, *Exploring the Use of Model-based Systems Engineering (MBSE) to Develop Systems Architectures in Naval Ship Design*, Defense Technical Information Center, 2010.
[64] K. Thramboulidis, "Challenges in the development of mechatronic systems: The mechatronic component," in the *Proceedings of IEEE International Conference of Emerging Technologies and Factory Automation*, pp. 624–631, 2008.
[65] N. Adamsson, *Interdisciplinary Integration in Complex Product Development: Managerial Implications of Embedding Software in Manufactured Goods*, KTH Royal Institute of Technology, 2007.
[66] H. Anacker, R. Dorociak, R. Dumitrescu, and J. Gausemeier, "Integrated tool-based approach for the conceptual design of advanced mechatronic systems," in the *Proceedings of IEEE International Systems Conference (SysCon)*, pp. 506–511, 2011.
[67] J. El-Khoury, *A Model Management and Integration Platform for Mechatronics Product Development*, KTH Royal Institute of Technology, 2006.
[68] Object Management Group, "OMG systems modeling language (OMG SysMLTM)," 2008.
[69] No Magic, Inc., "Unified modeling language (UML), SysML, UPDM, SOA, business process modeling tools," [Online]. Available: https://www.nomagic.com/. [Accessed: February 27, 2018].
[70] N. Magic, Inc., "Cameo simulation toolkit," 2011.
[71] J. E. Kasser, "Seven systems engineering myths and the corresponding realities," in the *Proceedings of the Systems Engineering Test and Evaluation Conference,* 2010.
[72] S. Friedenthal, A. Moore, and R. Steiner, "A practical guide to SysML: The systems modeling language," Morgan Kaufmann, 2014.
[73] Y. Umeda, H. Takeda, T. Tomiyama, and H. Yoshikawa, "Function, behaviour, and structure," *Applications of Artificial Intelligence in Engineering*, vol. 1, pp. 177–194, 1990.
[74] J. S. Gero, "Design prototypes: A knowledge representation schema for design," *AI Magazine*, vol. 11, no. 4, pp. 26, 1990.
[75] B. Chandrasekaran, "Representing function: Relating functional representation and functional modeling research streams," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 19, no. 2, pp. 65–74, 2005.
[76] B. Bennett and C. Fellbaum, "Behavior of a technical artifact: An ontological perspective in engineering," in the *Proceedings of 4th International Conference in Frontiers in Artificial Intelligence and Applications*, vol. 150, pp. 214-225, 2006.
[77] S. Szykman, J. W. Racz, and R. D. Sriram, "The representation of function in computer-based design," in the *Proceedings of the 11th ASME Design Engineering Technical Conferences*, 1999.
[78] M. N. Saunders, C. C. Seepersad, and K. Hölttä-Otto, "The characteristics of innovative, mechanical products," *Journal of Mechanical Design*, vol. 133, no. 2, pp. 021009, 2011.

[79] A. Albers and C. Zingel, "Challenges of model-based systems engineering: A study towards unified term understanding and the state of usage of SysML," in the *Proceedings of the 23rd CIRP Design Conference, Smart Product Engineering*, pp. 83–92, 2013.

[80] B. Kruse, C. Münzer, S. Wölkl, A. Canedo, and K. Shea, "A model-based functional modeling and library approach for mechatronic systems in SysML," in the *Proceedings of the 32nd ASME Computers and Information in Engineering Conference*, pp. 1217–1227, 2012.

[81] A. Sofer, et. al., "Guide to the systems engineering body of knowledge (SEBoK) v. 1.4," *IEEE Computer Society Press,* 2015.

[82] P. E. Vermaas, "The coexistence of engineering meanings of function: four responses and their methodological implications," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 27, no. 3, pp. 191–202, 2013.

[83] I. Reymen, *Improving Design Processes Through Structured Reflection: A Domain-Independent Approach,* Technische Universiteit Eindhoven, 2001.

[84] V. Hubka and W. E. Eder, *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. Springer Science & Business Media, 2012.

[85] R. Reil, "What makes model-based systems engineering transformational?" *Insight*, vol. 18, no. 3, pp. 16–17, 2015.

[86] H. Mili, A. Mili, S. Yacoub, and E. Addy, *Reuse-based Software Engineering: Techniques, Organization, and Controls*. Wiley-Interscience, 2001.

[87] C. Jackson and M. Buxton, *The Design Reuse Benchmark Report: Seizing the Opportunity to Shorten Product Development*, Aberd. Group, 2007.

[88] E. Girczyc and S. Carlson, "Increasing design quality and engineering productivity through design reuse," in *Proceedings of the 30th international Design Automation Conference*, pp. 48–53, 1993.

[89] A. H. B. Duffy and A. F. Ferns, "An analysis of design reuse benefits," In the *Proceedings of the 12th International Conference on Engineering Design*, pp. 799-804, 1998.

[90] R. M. Arlitt, R. B. Stone, and I. Y. Tumer, "Impacts of function-related research on education and industry," *Impact of Design Research on Industrial Practice*, Springer, pp. 77–99, 2016.

[91] R. B. Stone and K. L. Wood, "Development of a functional basis for design," *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359–370, 2000.

[92] H. Balkhair and C. W. de Silva, "A systematic approach for functional decomposition of mechatronic system design using mechatronic design quotient (MDQ)," in the *Proceedings of the 9th International Conference on Computer Science & Education (ICCSE)*, pp. 135–139, 2014.

[93] B. W. Caldwell, *Evaluating the Use of Functional Representations for Ideation in Conceptual Design*, PhD Dissertation, Clemson University, 2011.

[94] B. W. Caldwell, C. Sen, G. M. Mocko, J. D. Summers, and G. M. Fadel, "Empirical examination of the functional basis and design repository," *Design Computing and Cognition*, Springer, pp. 261–280, 2008.

[95] B. W. Caldwell, C. Sen, G. M. Mocko, and J. D. Summers, "An empirical study of the expressiveness of the functional basis," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 25, no. 3, pp. 273–287, 2011.

[96] C. Sen, B. W. Caldwell, J. D. Summers, and G. M. Mocko, "Evaluation of the functional basis using an information theoretic approach," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 24, no. 1, pp. 87–105, 2010.
[97] C. Sen, J. D. Summers, and G. M. Mocko, "Topological information content and expressiveness of function models in mechanical design," *Journal of Computing and Information Science in Engineering*, vol. 10, no. 3, pp. 031003, 2010.
[98] W. Borutzky, "Bond graph modeling and simulation of multidisciplinary systems–an introduction," *Journal of Simulation Modeling Practice and Theory*, vol. 17, no. 1, pp. 3–21, 2009.
[99] U. Sellgren, *Simulation-driven Design: Motives, Means, and Opportunities*, KTH Royal Institute of Technology, 1999.
[100] Siemens PLM Software, "Siemens NX," [Online]. Available: https://www.plm.automation.siemens.com/en/products/nx/. [Accessed: March 10, 2018].
[101] CATIA™ - Dassault Systèmes®, "3D Modeling Solutions," [Online]. Available: https://www.3ds.com/products-services/catia/. [Accessed: March 07, 2018].
[102] Agnisys, "Best Products & Services for System," *Verilog / UVM*, 2017.
[103] C. Münzer, *Constraint-Based Methods for Automated Computational Design Synthesis of Solution Spaces*, PhD Dissertation, ETH Zurich, 2015.
[104] C. J. Paredis, A. Diaz-Calderon, R. Sinha, and P. K. Khosla, "Composable models for simulation-based design," *Engineering with Computers*, vol. 17, no. 2, pp. 112–128, 2001.
[105] A. Canedo and J. H. Richter, "Architectural design space exploration of cyber-physical systems using the functional modeling compiler," Journal of *Procedia CIRP*, vol. 21, pp. 46–51, 2014.
[106] M. Grabisch, "The application of fuzzy integrals in multicriteria decision making," *European Journal of Operational Research*, vol. 89, no. 3, pp. 445–456, 1996.
[107] J. L. Marichal, "An axiomatic approach of the discrete Choquet integral as a tool to aggregate interacting criteria," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 800–807, 2000.
[108] C. W. de Silva, "Sensing and information acquisition for intelligent mechatronic systems," in *Proceedings of the Symposium on Information Transition, Chinese Academy of Science, Hefei,* pp. 9–18, 2003.
[109] C. Labreuche and M. Grabisch, "The Choquet integral for the aggregation of interval scales in multicriteria decision making," *Fuzzy Sets Systems*, vol. 137, no. 1, pp. 11–26, 2003.
[110] C. Labreuche, "Construction of a Choquet integral and the value functions without any commensurateness assumption in multi-criteria decision making.," in the *Proceedings of the 7th Conference of the European Society for Fuzzy Logic and Technology* (*EUSFLAT),* pp. 90–97, 2011.
[111] S. Behbahani and C. W. de Silva, "Mechatronic Design Quotient as the Basis of a New Multicriteria Mechatronic Design Methodology," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 2, pp. 227–232, 2007.
[112] M. Xia, *Application of Machine Health Monitoring in Design Optimization of Mechatronic Systems*," University of British Columbia, 2017.
[113] S. Gao and C. W. de Silva, "Estimation distribution algorithms on constrained optimization problems," *Applied Mathematics and Computation*, vol. 339, pp. 323–345, 2018.
[114] B. Prasad, *Concurrent Engineering Fundamentals*, Prentice Hall, 1996.

[115] R. R. B. Abd, P. Udo, and S. Ralf, "Systematic mechatronic design of a piezo-electric brake," *Guidelines for a Decision Support Method Adapted to NPD Processess*, 2007.
[116] G. A. David and B. H. Michael, *Introduction to Mechatronics and Measurement Systems*. McGraw Hill, 2007.
[117] P. iViP Association, "Mechatronic process integration (MPI)," *Abschlussbericht ProSTEP IViP*, 2009.
[118] J. Gausemeier, M. Flath, and S. Moehringer, "Modeling and evaluation of principle solutions of mechatronic systems, exemplified by tyre pressure control in automotive systems," in the *Proceedings of Symposium of Design for X*, pp. 541–548, 2000.
[119] X. Liu-Henke, J. Lückel, and K.-P. Jäker, "Development of an active suspension/tilt system for a mechatronic railway carriage," *IFAC Proceedings Voume.*, vol. 33, no. 26, pp. 283–288, 2000.
[120] R. Sinha, C. J. Paredis, V.-C. Liang, and P. K. Khosla, "Modeling and simulation methods for design of engineering systems," *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, pp. 84–91, 2001.
[121] C. W. De Silva, *Mechatronics: A Foundation Course*. CRC press, 2010.
[122] P. Hehenberger, "Perspectives on hierarchical modeling in mechatronic design," *Advanced Engineering Informatics*, vol. 28, no. 3, pp. 188–197, 2014.
[123] S. D. Eppinger and K. T. Ulrich, *Product Design and Development*, *McGraw-Hill Education*, 1995.
[124] C. L. Dym, P. Little, E. J. Orwin, and E. Spjut, *Engineering Design: A Project-based Introduction*. John Wiley and sons, 2009.
[125] S. Szykman, J. W. Racz, and R. D. Sriram, "The representation of function in computer-based design," in the *Proceedings of the 11th ASME Design Engineering Technical Conferences, International Conference on Design Theory and Methodology*, 1999.
[126] H. Balkhair and C. W. de Silva, "Data Management for Multidisciplinary Mechatronic Systems," *International Journal of Emerging Technologies and Innovative Research*, vol. 5, no. 8, pp. 46–50, Aug. 2018.
[127] C. Zheng, P. Hehenberger, J. Le Duigou, M. Bricogne, and B. Eynard, "Multidisciplinary design methodology for mechatronic systems based on interface model," *Research in Engineering Design*, vol. 28, no. 3, pp. 333–356, 2017.
[128] U. Lindemann, M. Maurer, and T. Braun, *Structural complexity management: an approach for the field of product design*. Springer Science & Business Media, 2008.
[129] C. Zheng, *Design and Integration of Multi-disciplinary Interfaces: Method and Modelling Language for Mechatronic Systems Engineering*, Université de Technologie de Compiègne, 2015.
[130] J. Blyler, "Interface management," *IEEE Instrumentation and Measurement Magazine*, vol. 7, no. 1, pp. 32–37, 2004.
[131] M. Eigner, T. Gilz, and R. Zafirov, "Proposal for functional product description as part of a plm solution in interdisciplinary product development," in the *Proceedings of the 12th Design, International Design Conference,* 2012.
[132] J. D. Summers, C. Eckert, and A. K. Goel, "Function in engineering: benchmarking representations and models," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 31, no. 4, pp. 401–412, 2017.

[133] Y.-M. Deng, "Function and behavior representation in conceptual mechanical design," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 16, no. 5, pp. 343–362, 2002.
[134] B. Kruse, T. Gilz, K. Shea, and M. Eigner, "Systematic comparison of functional models in sysml for design library evaluation," *Journal of Procedia CIRP*, vol. 21, pp. 34–39, 2014.
[135] S. J. Wölkl, *Model Libraries for Conceptual Design*, Ph. D. thesis, Munich, Germany: Technische Universität München, 2013.
[136] H. Komoto and T. Tomiyama, "A framework for computer-aided conceptual design and its application to system architecting of mechatronics products," *Computer-Aided Design*, vol. 44, no. 10, pp. 931–946, 2012.
[137] S. Uckun, *Meta II: Formal Co-verification of Correctness of Large-scale Cyber-physical Systems During Design*, Palo Alto Research Center, Technical Report, pp. 1–43, 2011.
[138] A. Canedo, J. Wan, and M. A. Al Faruque, "Functional modeling compiler for system-level design of automotive cyber-physical systems," in the Proceedings of *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 39–46, 2014.
[139] J. Wan, A. Canedo, and M. A. Al Faruque, "Functional model-based design methodology for automotive cyber-physical systems," *IEEE Systems Journals*, vol. 11, no. 4, pp. 2028-2039, 2017.
[140] B. Kruse and K. Shea, "Design library solution patterns in SysML for concept design and simulation," in the *Proceedings of the 26th Design Conference, Procedia CIRP*, vol. 50, pp. 695–700, 2016.
[141] A. S. Vincentelli, "Defining platform-based design," *EE Design, EE Times*, 2002.
[142] A. Bhave, B. H. Krogh, D. Garlan, and B. Schmerl, "View consistency in architectures for cyber-physical systems," in the *Proceedings of IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS)*, pp. 151–160, 2011.
[143] H. Elmqvist *et al.*, "ModelicaTM-A unified object-oriented language for physical systems modeling," *Tutorial and Rational*, vol. 1, 1999.
[144] M. Han, Y. Song, W. Zhao, Y. Cheng, and J. Xiang, "Simulation and optimization of synchronization control system for cfetr water hydraulic manipulator based on AMEsim," *Journal of Fusion Energy*, vol. 34, no. 3, pp. 566–570, 2015.
[145] M. J. Zhang, "Die forming simulation of pm parts based on AMESim," *Key Engineering Materials*, vol. 667, pp. 47–53, 2016.
[146] M. Häggström, *Thermal Modelling of A Truck Gearbox*, Independent thesis Advanced level, Luleå University of Technology, 2017.
[147] M. A. A. Siddique, W.-S. Kim, S.-Y. Beak, Y.-J. Kim, and C.-H. Choi, "Simulation of hydraulic system of the rice transplanter with AMESim software," *The American Society of Agricultural and Biological Engineers Annual International Meeting*, 2018.
[148] Y. Cao, Y. Liu, and C. J. Paredis, "System-level model integration of design and simulation for mechatronic systems based on SysML," *Journal of Mechatronics*, vol. 21, no. 6, pp. 1063–1075, 2011.
[149] Y. Cao, Y. Liu, H. Fan, and B. Fan, "SysML-based uniform behavior modeling and automated mapping of design and simulation model for complex mechatronics," *Journal of Computer-Aided Design*, vol. 45, no. 3, pp. 764–776, 2013.
[150] B. Chabibi, A. Douche, A. Anwar, and M. Nassar, "Integrating SysML with simulation environments (Simulink) by model transformation approach," in the *Proceedings of IEEE*

25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 148–150, 2016.
[151] J. F. Broenink, "Introduction to physical systems modelling with bond graphs," *SiE Whitebook on Simulation Methodologies*, vol. 31, 1999.
[152] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, and K. L. Wood, "A functional basis for engineering design: reconciling and evolving previous efforts," *Research in Engineering Design*, vol. 13, no. 2, pp. 65–82, 2002.
[153] B. Helms, H. Schultheiss, and K. Shea, "Automated mapping of physical effects to functions using abstraction ports based on bond graphs," *Journal of Mechanical Design*, vol. 135, no. 5, p. 051006, 2013.
[154] H. Paynter, "Analysis and design of engineering systems," *Massachusetts Institute of Technology*, 1961.
[155] D. B. Dooner, A. Palermo, and D. Mundo, "An intermittent motion mechanism incorporating a geneva wheel and a gear train," *Transaction Canadian Society of Mechanical Engineering*, vol. 38, no. 3, pp. 359–372, 2014.
[156] S. I. Valdez, S. Botello-Aceves, H. M. Becerra, and E. E. Hernández, "Comparison between a concurrent and a sequential optimization methodology for serial manipulators using metaheuristics," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3155-3165, 2018.
[157] R. X. Lu, C. W. de Silva, M. H. Ang, J. A. Poo, and H. Corporaal, "A new approach for mechatronic system design: Mechatronic design quotient (MDQ)," in the *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 911–915 2005.
[158] S. Behbahani, *Practical and Analytical Studies on the Development of Formal Evaluation and Design Methodologies for Mechatronic System*s, PhD Dissertation, University of British Columbia, 2007.
[159] M. Hammadi, J. Y. Choley, O. Penas, A. Riviere, J. Louati, and M. Haddar, "A new multi-criteria indicator for mechatronic system performance evaluation in preliminary design level," in the *Proceedings of the 9th France-Japan & 7th Europe-Asia Congress on Mechatronics (MECATRONICS)-13th International Workshop on Research and Education in Mechatronics (REM)*, pp. 409–416, 2012.
[160] C. W. de Silva, *Mechatronic systems: Devices, Design, Control, Operation and Monitoring*, CRC press, 2007.
[161] H. T. Nguyen, V. Kreinovich, J. Lorkowski, and S. Abu, "Why sugeno λ-measures," in Proceedings of *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–7, 2015.
[162] A. K. Singh, "Signed lambda-measures on effect algebras," *Proceedings of the National Academy of Science, Indian Section A: Physical Science*, pp. 1–7, 2018.