INTEGRATION OF SYSTEM-LEVEL DESIGN AND MECHANICAL DESIGN MODELS IN THE DEVELOPMENT OF MECHANICAL SYSTEMS

ROYAL INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER AND SYSTEM SCIENCES (DSV)

A thesis presented to the academic faculty

By

HAMID SHAHID

hshahid@kth.se

(June - 2011)

In partial fulfillment of the requirements for the degree
Masters in Interactive Systems Engineering

Supervisor at KTH
Ahsan Qamar
Examiner at KTH
Fredrik Kilander
Supervisor at Mirconic AB
Carl During

This thesis corresponds to 20 weeks (30 HCS) of full time work

This page has been left blank intentionally

Abstract

Modern-day systems are becoming complex due to the growing needs of the market. These systems contain various subsystems developed by different groups of engineers. Particularly, all mechatronics systems involve different mechanical, electrical and software parts developed by multidisciplinary teams of engineers from different backgrounds. Designing of these complex systems requires effective management, of the engineering and the system integration information, across all the involved disciplines. Model Based System Engineering (MBSE) is one of the effective ways for managing the engineering design process. In MBSE, design information is formally stored in the form of models, which allows better control of requirements throughout the development life cycle and provides ability to perform better analysis.

Engineers usually are expert in their own discipline, where they utilize modeling languages and tools with a domain-specific focus. This creation of models with the domain-specific focus does not provide a view of the system as a whole. Hence, in order to have a complete system view, it is required to provide information transfer means across different domains, through models developed in different modeling languages and tools supporting them. Model integration is one of the ways to integrate and transfer model information across different domains.

An approach for model integration is proposed, with the focus on the integration between system level models created in SysML and mechanical CAD (MCAD) models. The approach utilizes the feature of SysML to create domain specific profiles and presents a *SysML profile for MCAD* domain. This profile aids in establishing a mapping between SysML and MCAD concepts, as it allows the extension of SysML constructs to represent MCAD concepts in SysML. Model transformations are used to transform a model created through *SysML profile for MCAD* in to the corresponding model in a MCAD tool, and vice versa. A robot model is presented to exemplify the working of the approach and to explain the integration of mechanical design model with a system-level design model and vice versa.

The approach presented in this thesis depicts a scalable concept, which can be extended towards the integration of other domains with MCAD, by building new relations and profiles in SysML. This approach aids in co-evolution of a system model during domain-specific development activities, hence providing better means to understand the system as a whole.

This page has been left blank intentionally

Acknowledgement

First of all, I must thank my family because of all the love and support they provided to me in the good times and the bad times. I would like to thank Mirconic AB, Machine Design Department at KTH and ICT Department at KTH for giving me this opportunity to work on a cutting edge research. It has been an amazing and fruitful experience.

I would like to thank my supervisor, Ahsan Qamar, at Machine Design Department for all the guidance, moral and technical support. It is due to his constant support and encouragement that I have been able to accomplish my tasks. His valuable comments regarding the structure and contents of this report also helped me to improve the quality of my thesis report.

My sincere gratitude also goes to Prof. Carl During, Prof. Jan Wikander and Prof. Fredrik Kilander for their support throughout my work. I would also like to thank Matthias Biehl at Machine Design Department for his valuable advices and help.

I am also grateful for the academic and test licenses provided by No Magic, Inc. Last but not the least I would like to thank No Magic Support Team for providing technical support during crucial phases of my thesis.

This page has been left blank intentionally

Table of Contents

List of Figures

This page has been left blank intentionally

Thesis Outline

The thesis is organized in the following chapters;

Chapter 1 – Background & Introduction

This chapter provides a detailed background and introduction to the thesis by describing the main concepts and terminologies involved in this thesis. This chapter also covers the aims and objectives of the thesis.

Chapter 2 – State of the Art

A lot of research work has been done with respect to model transformation techniques and model integration approaches. This chapter highlights some of the most relevant research work in the area of tool and model integration.

Chapter 3 – Proposed Approach for Model Integration

This chapter highlights the approach proposed for model integration in this thesis. It also, describes the SysML profile created for MCAD domain and its compliance with the MCAD meta-model. Steps to perform model integration are also presented in this chapter.

Chapter 4 – Integration Case Study

In this chapter a complete example of a mechanical model of a robot is used to illustrate the working of purposed model integration approach.

Chapter 5 – Closure, Analysis & Future Work

This chapter provides a concise review over the model integration approach presented in this thesis. Moreover, this chapter highlights the limitations of the presented approach. This chapter defines the areas in which future work can be performed.

This page has been left blank intentionally

Chapter 1 – Introduction

Science and technology has become the part and parcel of our lives. If we look around, from personal vehicles to local transport systems, from passenger cruise-ships to warships and from passenger planes to jet fighters and helicopters, everything contains highly advanced and complex systems underneath.

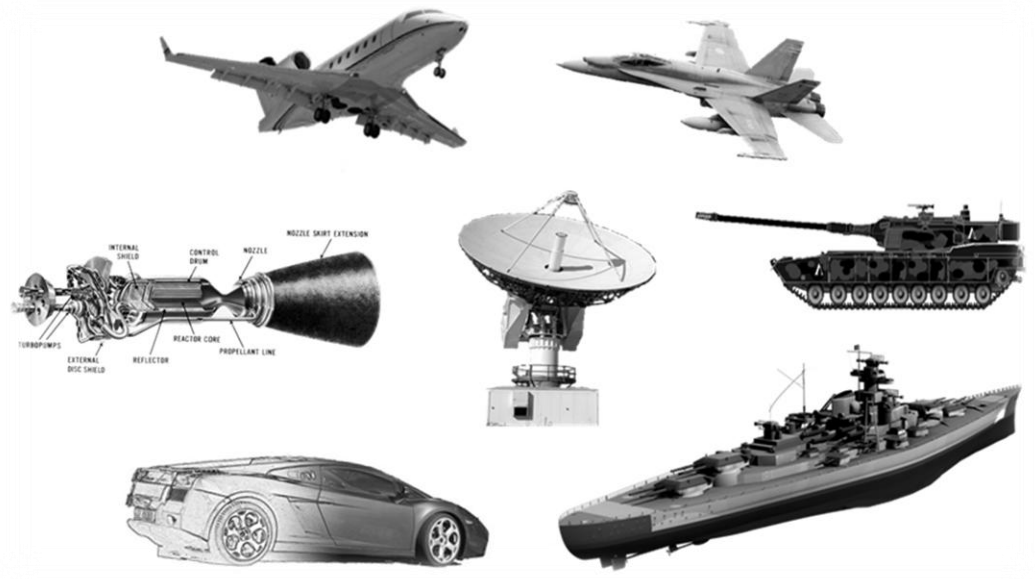

Figure 1 – Complex Systems **[***1***]**

Modern-day systems *(Figure 1)* are normally composed of various subsystems performing several distinct tasks within the main system. Usually, different subsystems are developed by different groups of engineers having different backgrounds. Engineers are mostly experts of their own domain and could not interpret the technical details of the subsystems developed by the engineers from any other domain. Irrespective of all the technical differences across the involved domains, it is vital for the stakeholders to always have a clear picture of the system as a whole.

As described by Sage et al. [*2*], most of the problems encountered in the development of multidisciplinary system are related to the organization and management of the growing quantities of design information rather than direct technological apprehensions. Therefore, designing of such multidisciplinary systems requires effective management, of the engineering and the system integration information, across all the involved disciplines.

In this thesis work, the main focus is on *mechatronic systems* in general and *mechanical design* in particular. Mechatronics is a multidisciplinary field of engineering and mechatronic systems usually involves concepts from electrical engineering, mechanical engineering and computers science. As elaborated by Tomiyama et al. [*3*], the complexity of a mechatronic system is due to the involvement of multi-disciplinary groups of engineers and the integration of various subsystems into a product. While developing mechatronic systems it is also vital, for the engineers and analysts, to understand the design and behavior of the system as a whole, along with the integration details of different subsystems involved in the products. Mechatronic design also requires an effective management of the engineering process.

1.1. Systems Engineering

Systems engineering is a multidisciplinary field of engineering with a focus on how complex projects should be designed and managed. It is an approach to develop balanced system in response to the diverse needs of stakeholders as discussed by Friedenthal et al. in [*4*]. Systems engineering focuses on the system as a whole, therefore guiding the engineering of complex system is the main functionality of systems engineering. It provides a bigger picture of the system and how the system will interact with the other systems. Moreover, systems engineering aids in the conceptual design phase of the system. Conceptual design is the initial and the most abstract stage of the design process starting with requirements and resulting in design concepts. Hence, many important design decisions are made during the conceptual design phase.

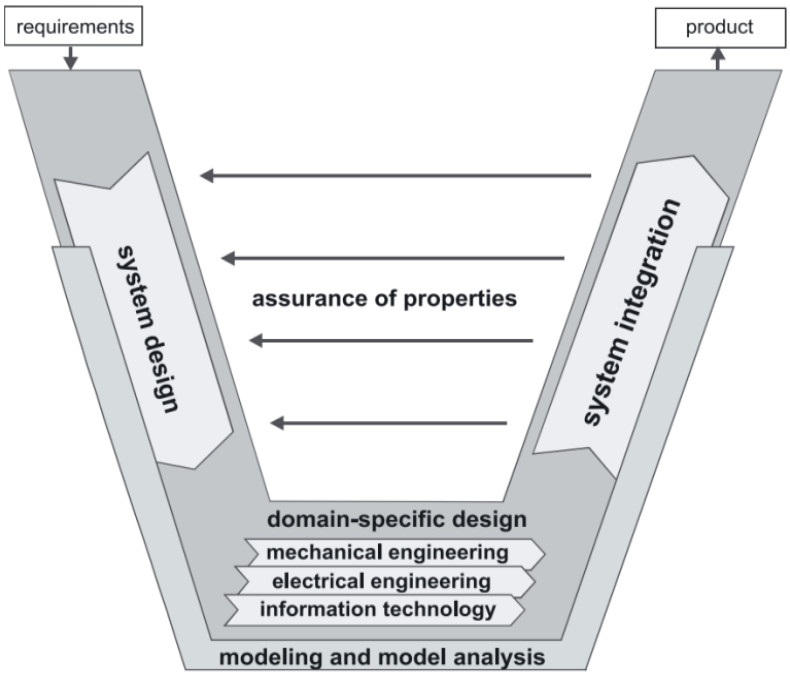

Figure 2 - The Systems Engineering "V" Model [*5*]

It is important to note that the systems engineering process is not a sequential process; it is rather iterative in nature and there are several graphical representations of the systems development life cycle, which are based on the needs of a particular field and type of the system. However, the most commonly used representation of system development life cycle for systems engineering is the Systems Engineering "V" [*5*] diagram which is also known as; the Systems Engineering "V" model (*Figure 2*).

1.2. Model Based Systems Engineering (MBSE)

Document Based System Engineering (DBSE) has been widely used to manage the system information before the emergence of Model Based System Engineering (MBSE). In DBSE approach, textual specification and design documents are exchanged between customers, users, developers, testers and with other stakeholders. These documents could be in hard-copy or electronic form restraining the design requirements and system specifications. As described by Friedenthal et al. in [*4*]; in the document based approach all the specification of the system and its subsystems, including hardware and software components, are usually combined in a hierarchical tree which is known called a specification tree.

DBSE has some evident limitations. It is very hard to keep a relation and track between design requirements, engineering analysis and test information. Moreover, it is really very difficult to assess the completeness and consistency of the system because information is spread across various documents.
MBSE, addresses the limitations of document-based approach and provides a much meticulous grasp of the system in all the phases of system development life cycle (SDLC). Therefore, in recent times, Model Based System Engineering has become a standard practice in the different engineering disciplines for instance in software engineering, mechanical engineering & etc.

In MBSE, design information is formally stored in the form of models, which allows better control of requirements throughout the development life cycle and provides the ability to perform better analysis. Now days, MBSE is typically accomplished with the help of various computer tools because computer tools aid systems engineers to efficiently create system models. In MBSE, a *model repository* contains the model elements which are depicted using graphical symbols in different types of diagrams. Computer tools enable the modelers to create, modify and delete individual model elements and their relationships in the model repository. The modeler uses graphical symbols on the diagrams to input the relevant information into the model repository. The primary purpose of a *system model* contained in a model repository is to define a system that complies with system requirements and allocates the requirements to the components of the system.

As described by Johnson in [*6*], design models are categorized into two types; *system models* and *analysis models*. System models are used to specify the desired structure, functions and behaviors of the system, for example MCAD models to illustrate mechanical structure of the system, etc. Whereas, the other type of models known as *analysis models* are used to anticipate the behavior of the system for instance, cost models, finite element models, etc.

1.3. Modeling Formalisms

Various information modeling languages have been developed over time for the support of the MBSE design process. The Object Management Group (OMG) has presented two of the most successful information modeling languages; namely "Unified modeling language" (UML) [*7*] and "Systems Modeling Language" (SysML) [*8*] [*9*]. In this thesis, SysML would be the center of attention.

1.3.1. SysML

SysML is also a graphical modeling language but with an inclination towards systems engineering. As an alternative of developing SysML as a totally new language, it has been extended from UML 2.0 to support and address requirements which are specific to systems engineering. SysML successfully fulfills the modeling needs of the systems engineering process and therefore, SysML has been widely in use for Model Based Systems engineering.

SysML is developed with the focus on the needs and the requirements of systems engineers. SysML tries to generalize UML for complex systems, which could include non-software components such as, hardware, information, processes, personnel and facilities as elaborated by Friedenthal et al. in [*10*]. SysML provides the following diagrams which enable systems engineers to express system behavior;

- Activity Diagrams
- Sequence Diagrams
- State machine diagram
- Parametric diagrams

Figure 3 depicts the relationship between UML and SysML along with a list showing a few well known SysML extensions to UML, also known as *SysML Profile*. SysML is discussed in detail in section-1.7.

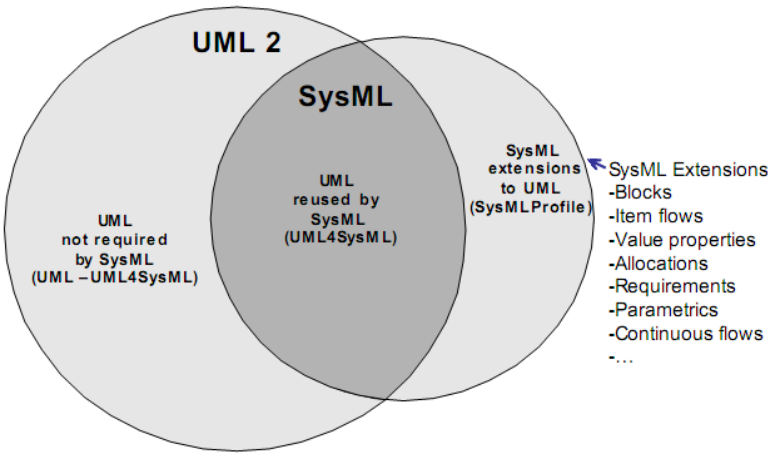

Figure 3 - Overview of SysML/UML Inter-relationship **[*9*]**

1.4. Modeling of Mechatronics Systems

Although, mechatronics is multi-disciplinary field and mechatronic systems are composed of electronics, mechanics, and software, yet mechatronic systems are not the result of separately designed subsystems merged together in the end. In contrast, a mechatronic system should be designed and modeled as an integrated system from the very beginning and the system design should co-evolve among all the involved domains.

Since mechatronic system combines different disciplines and technologies therefore, to find out the ideal blend of the technologies, different design solutions are considered and analyzed. Conventionally, the system is modeled on the abstract level in the conceptual phase of the system design. Whereas detailed modeling, in domain-specific modeling languages, is performed in the later stages of system design.

However, as discussed by Tomiyama et al. in [*3*], engineers and designers are educated and trained discipline-wise. Therefore, domain experts usually do not possess inter-disciplinary knowledge. It is also difficult and time consuming to educate domain experts to be a multi-disciplinary expert. Hence, an inter-disciplinary team of experts is utilized for multi-disciplinary development; this kind of team is particularly beneficial for modeling the system in its conceptual phase. For domain-specific and detailed system modeling, experts utilize domain-specific languages and professional modeling tools.

1.5. Establishing a Complete System View

The involvement of multidisciplinary experts and different modeling tools used for each domain splits the design of mechatronic systems among various views, where each view exhibits a different level of abstraction. A view of the system can reflect a certain part of the system or the system as a whole. A domain specific view can be misunderstood by the members of other domains because of the domain specific concepts and terminologies for instance; a computer software engineer would not be able to understand an electrical circuit diagram and the impacts of un-familiar concepts and terminologies. Therefore, when multiple domains are involved in system design, the role of system engineering becomes more vital because it facilitates engineers to create abstract views of the system which can be understood across different domains.

Abstraction can be categorized in two types, *vertical abstraction* and *horizontal abstraction. Vertical abstraction* is between models at different levels of detail whereas; *horizontal abstraction* is the abstraction of information among models that are at the same level of abstraction. A similar account on the types of model abstraction is discussed by Ahsan et al. in [*11*]. This multi-view modeling allows sharing critical design information more easily but also creates problems of information consistency and traceability due to different levels of abstraction in different tools. Therefore, a complete system view is always helpful and beneficiary for understanding the relations between the sub-systems and the inter-dependencies between design activities performed in different domains.

1.6. Model Integration

As discussed in previous section (1.5); a complete view of the system is important to analyze the system as a whole. However, models created in different domain specific tools contain various domain specific characteristics, while only certain aspects of these models are relevant across the other domains. Thus to create such a complete system view, a mechanism is required to extract and then integrate the critical design information captured in all these analysis and design models created in different domain specific modeling tools. It is also vital to maintain information consistency among different models, so that all views and models should remain consistent with the actual system requirements.

Model integration is a way to integrate and transfer model information across different domains. Model integration is usually categorized into two main distinctions known as “deep integration” and “functional integration”. Deep integration combines two or more given models and results into a single new model which is represented in the same definitional formalism as the given models. On the other hand, functional integration does not generate a new model, but superimposes some computational agenda on given models to perform calculations or to direct certain models output to other models input. Detailed descriptions, of both types of model integration with examples, are presented by Geoffrion [*12*]. The core concept of the model integration approach (chapter-3) presented in this thesis work is a blend of both *functional* and *deep* model integration because it aims to establish a complete system view by combining multiple models created across different domains as well as it superimposes the condition of extracting only the useful design information from the domain specific models.

One important thing to consider here is that, integration of models across multiple domains is different than maintaining interoperability between models and modeling tools. *Interoperability* involves managing similar information across different formats, for instance using standard file formats such as XML, etc. Conversely, model integration involves managing different information across different tools and domains. In such cases, tools usually do not conform to common file formats and if they do then there are limitations such as data loss and readability issues with XML and XMI [*13*,*14*].

One of the alternatives to model integration would be to exclusively use a huge modeling language or a platform that can support all types of modeling requirements. This language or platform should be able to represent all aspects of the systems engineering life cycle, all the way from requirement capturing to the deployment of the system. Practically, creating such a huge language is not a feasible solution and it would be equivalent of *re-inventing the wheel,* because all those existing domain specific features and capabilities provided by domain specific tools would be created again for such a huge modeling language. Moreover, this solution requires educating the engineers with a new modeling language which itself is time consuming task. Hence, for multidisciplinary systems, model integration is the most feasible solution in order to create a complete system view.

1.6.1. Model Transformation

Model transformation is a key concept in model integration. *Figure 4*, gives an overview of the main concepts involved in the process of model transformation presented by K. Czarnecki [*15*]. A simple scenario of a model transformation is illustrated in the *Figure 4*. In order to perform a model transformation, some transformation rules are defined with respect to both source and target metamodels. These transformation rules define the mapping between the model elements of both metamodels. The model transformation engine then reads the source model which is compliant to the source metamodel and executes the transformation rules to create the target metamodel which conforms to the target metamodel.

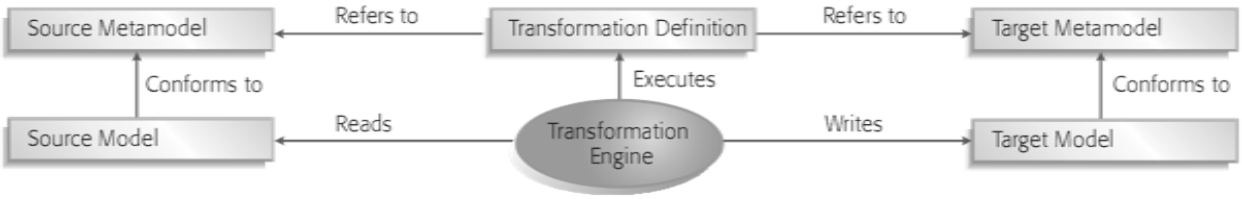

Figure 4 – Basic Concept of Model Transformation **[***15***]**

1.7. Model Integration support in SysML

SysML specifications illustrates that SysML not only provides support for model and data exchange but also one of the intents of the SysML language is to provide a platform for model unification and integration [*8*]. Johnson elaborates in his work [*6*], that SysML constructs are capable of supporting model integration but they do not provide a straight forward mechanism to perform model integration to users rather it relies on users to enable model integration.

SysML is a modeling language and a modeling language is defined by metamodel. Metamodel contains a number of distinct language concepts, represented by metaclasses. Metaclasses have a set of properties and constraints on them. Metaclasses are related to each other using generalizations and associations. As stated by Friedenthal et al. in [*4*], the models created by modelers are basically the instances of these metaclasses.

Since, SysML is an extension of UML consequently most of the SysML constructs have been defined as UML stereotypes. A stereotype is use to create customized classifications of modeling elements. SysML provides support for system level design as well as a mechanism for customizing SysML by creating customized profiles. A profile adds new concepts to a language by means of stereotypes and the use of profiles enables the domain-specific description of different domains in SysML. Hence, in order to

perform domain specific modeling, SysML users are allowed to create additional stereotypes according to the concepts of their particular domains.

In this thesis, stereotyping has been used to create a SysML profile for MCAD domain, to map the mechanical concepts used in MCAD on SysML constructs. A SysML profile compliant to MCAD meta-model is purposed, as it would enable modelers to perform domain specific modeling using SysML in different modeling tools. As stated by Shah et al. in [*14*], the use of profile also allows mapping to be specified between only those parts of domains that are related to each other instead of translating complete model from one domain to another. SysML profile for MCAD would be discussed in detail in proceeding sections.

1.8. Thesis Motivation & Purpose

As discussed in 1.4 & 1.5, mechatronics is a multidisciplinary field of engineering and it involves different groups of engineers from diverse backgrounds to design the mechatronic systems. These engineers use their domain specific languages and modeling tools to perform modeling activities. This involvement of different domains and domain specific modeling tools not only creates complexity of design in terms of managing critical design information across different domains but also produce various challenges while integrating the subsystems within a mechatronic system.

Despite of all these challenges, the nature of mechatronic design requires the design to co-evolve across different domains. The co-evolution of mechatronic design requires some means of information transfer across different domain and model integration plays a vital role in the transfer of critical design information across different models and domains. Due to this importance and the impact of model integration in system design paradigm, many researchers are actively participating in this area, to come up with effective solutions for model integration, some of them are discussed in chapter-2.

1.9. Aims and Objectives

The central theme of the thesis is to integrate system level design and domain specific design models, and to investigate what are the needs for doing integration among them. Therefore, the work presented here limits its scope towards integrating SysML and MCAD models only. As illustrated in *Figure 5*, one of the aims of this thesis involve; the study and analysis of mechanical CAD (MCAD) domain and SysML modeling. This thesis work, also aims to understand the problems occurring during the integration of design models from different design tools.

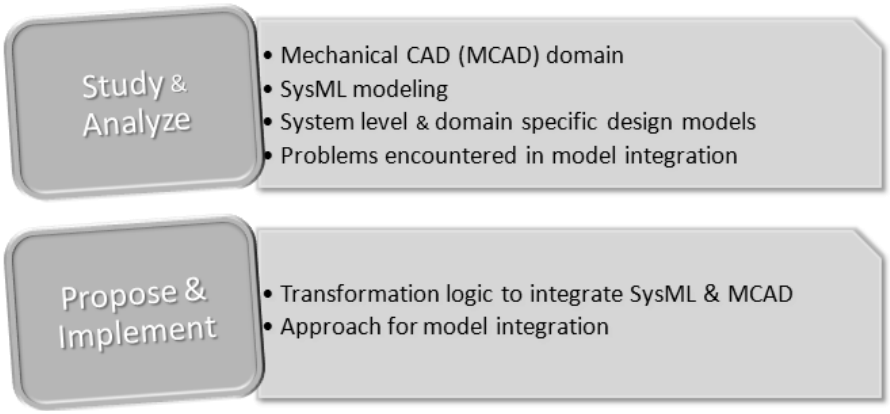

Figure 5 - Aims & Objectives of the thesis

Moreover, one of the core objectives of the thesis it to propose and implement a transformation and integration approach to integrate system level and domain specific design models. The research questions that are the base of this research are as follows;

1.9.1. Research Questions and methodologies

a. "Can a SysML model be used to represent an abstract view of a mechanical design model?"

It required a thorough understanding of SysML language and its capacities along with the analysis of MCAD meta-model and MCAD models. The profile mechanism of SysML is used to extend SysML constructs to map them according to MCAD concepts, so that modelers can directly create system level models using the concepts of target domain e.g. MCAD. Therefore, by using SysML profile, SysML models can be used to represent abstract views of mechanical design models.

b. "What is the process to integrate a system model, created in SysML by using SysML profile for MCAD, with a corresponding MCAD design model created in a MCAD tool?"

In order to propose and implement an approach to integrate a system model with corresponding MCAD design model, it not only required the analysis of system models and mechanical design models but it also required the analysis of modeling tools and the support they provide to extract selected design information. Since, MCAD design models are much detailed in terms of structural information, consequently, it is important to be able to extract the selected model information from modeling tools so that it can be processed and transformation languages can be used to transform the extracted model information according to the target tools.

c. "What are the challenges and limitations of the integration of system level and mechanical design models?"

To answer this question, the results of the model integration approach would be analyzed to verify if the model integration between SysML to MCAD would aid the co-evolution of the design for mechatronic systems. This analysis would also help to determine, if there are any limitations in this approach and what are the things that should be considered while performing such type of model integration.

Chapter 2 – State of the Art

This thesis aims to integrate different design models to support the design process of mechatronic systems. As discussed in chapter-1, model integration and model transformation plays an important role in the presented approach. A lot of research work has been done with respect to model transformation techniques and model integration approaches. This chapter highlights some of the most relevant research work in the area of tool and model integration.

2.1. Integration of UML and Modelica

A lot of work has been done in order to integrate UML with Modelica as it supports model exchange capabilities. Pop et al. [*16*] have worked on the integration of UML and Modelica. To address the integration problems they created a UML profile for Modelica called ModelicaML. ModelicaML profile reuses several UML and SysML constructs to create completely new language constructs and it includes new diagrams like Modelica class diagram, the equation diagram, and the simulation diagram. ModelicaML enables users to graphically represent a Modelica simulation model [*17*].

A specialized version of UML called UML^{H} is developed by Nytsch-Geusen [*18*]. Nytsch-Geusen uses UML^{H} in ModelicaML for graphical description and model-based development of hybrid systems. Using Modelica state-chart extension, Nytsch presented hybrid system models as Modelica models that are based on DAE's combined with discrete state transitions models. When users use UML^{H} editor and a Modelica tool that supports code generation, users must only insert the equation-based behavior of the system [*17*].

2.2. Modeling Continuous System Dynamics in SysML

Johnson et al. [*17*] presented a formal approach to model continuous system dynamics in SysML using differential algebraic equations (DAE's). Their approach is based on a language mapping between SysML and Modelica and supports the modeling of system behaviors based on exchange of energy and signals. They have shown that existing elements of SysML can be adapted for this purpose. Their main objective is to enable the integration of Modelica models in SysML and improve SysML's ability to aid designers during the development of contemporary, complex systems [*17*].

2.3. Integration by Constraints Propagation

In order to cope with the design changes, mechatronics system design requires an immediate feedback from all the different domains. In order to provide immediate feedback, cross-disciplinary constraints can be classified, represented, modeled and bi-directionally propagated. An approach, based on cross-disciplinary constraint modeling and propagation, for the integration of mechanical and electrical CAD systems is proposed by Kenway Chen et al. in [*19*].

In constraint modeling, mechatronic systems are modeled in the form of block-diagrams and relationships between domain-specific constraints are identified and categorized [*19*]. A selected part of mechatronic system is analyzed to identify and classify domain specific constraints. This identification and classification is based on the associated functions, physical forms, system behavior and other design requirements.

2.4. Multi-view modeling using SysML profiles

Shah et al., have presented a method to use SysML profiles and graph transformations to integrate multiple views of a system in [*20*]. They developed a method to integrate EPLAN an electrical CAE tool with SysML models, using SysML profile and graph transformations.

Electrical CAE tools such as EPLAN are well suited for detailed design of power and control systems with the support of schematic or wiring diagrams. This methods aims to enable modelers to trace decisions made to the corresponding requirements defined in the SysML as well as by maintaining bidirectional consistency between other domains that are link with SysML [*20*].

2.5. Thesis Contribution

As discussed in chapter-1, the approach presented in this thesis is mainly focused on the integration of system level and mechanical design models in order to support the design process of mechatronic systems. The presented approach is relatively similar to the work done by Shah et al. [*20*] in terms of using SysML profile and model transformation techniques for model integration. However, the difference is that, Shah does not cater for mechanical design.

Moreover, in this thesis work, Atlas Transformation Language (ATL) [*21*] and Model Query Language (MQL) [*22*] have been used as transformation languages for model transformation instead of graph transformation approach used by Shah. Tool specific APIs and plug-ins have also been used to extract and inject the model information from one tool into another, as described in following sections. *Figure 6* illustrates a conceptual view of the model transformation approach presented in this thesis;

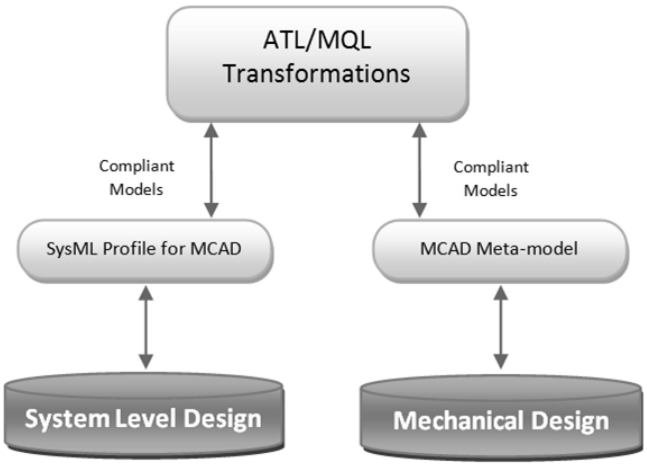

Figure 6 – Conceptual-view of the Proposed Model Integration Approach

Thesis method involves performing the following steps;

1. Defining metamodels for both SysML and MCAD domains.
2. Creating SysML profile to enable domain specific modeling, for MCAD, in modeling tools e.g. Magic-Draw [*23*].
3. Using transformations to transform the SysML profile to domain specific metamodel e.g. MCAD metamodel.
4. Use of tool specific API calls to update information in tools after the transformation process is complete.

This page has been left blank intentionally

Chapter 3 – Proposed Approach for Model Integration

This chapter elaborates the proposed approach to integrate the system level design models with mechanical design models by highlighting all the steps involved in model integration between Magic-Draw [*23*] and a MCAD specific tool, Solid-Edge [*24*]. It describes about the SysML profile created for MCAD and its compliance with the MCAD metamodel. Moreover, the Ecore models created in Eclipse modeling framework (EMF) [*25*] and their significance in model transformations is also presented in this chapter.

3.1. SysML profile for MCAD

As described before (section-1.7), SysML provides a mechanism of creating profiles for customizing SysML for specific domains and by using a profile mechanism SysML constructs can be mapped on the concepts of the target domain. One advantage of using profiles is that it reduces the complexity of integrating different views. Hence, instead of creating customized mapping among all the domains involved, individual mapping can be defined between each domain and its corresponding SysML profile as discussed in [*14*]. As this thesis is focused on MCAD domain therefore, after a detailed analysis of Solid-Edge's MCAD-metamodel (*Figure 7*), a SysML profile (*Figure 8*) is presented which is actually compliant to the Solid-Edge's MCAD metamodel (*Figure 7*).

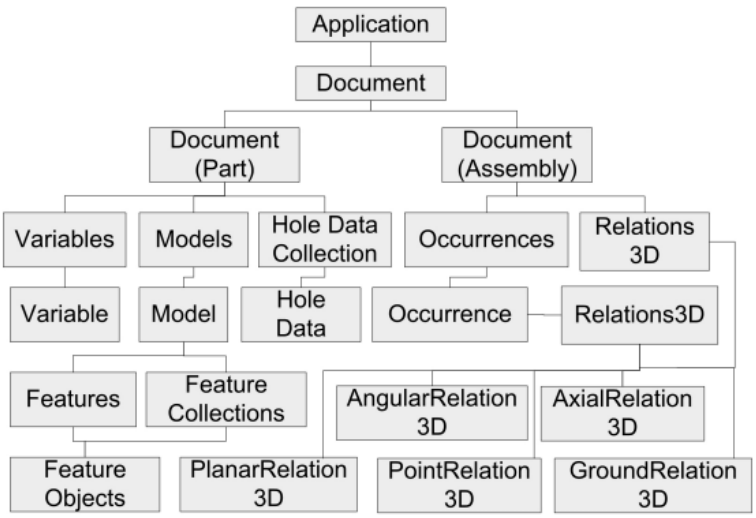

Figure 7 – Solid-Edge's MCAD metamodel

Solid-Edge's MCAD metamodel (*Figure 7*) depicts that every Solid-Edge application consists of a *Document*. This *Document* could be an *Assembly* or a *Part.* An *Assembly* can contain multiple *Occurrences* of *Part* or sub-assemblies, and these *occurrences* (parts) can have different types of relationships among them, these relationships can be of 2D and 3D type. In this thesis, we are accounting only the 3D relations which are known as *Relationship3D* in Solid-Edge. Furthermore, each *Part* can have different types of *Variables,* which can represent different properties of the object being modeled e.g. material properties, geometric properties, etc.

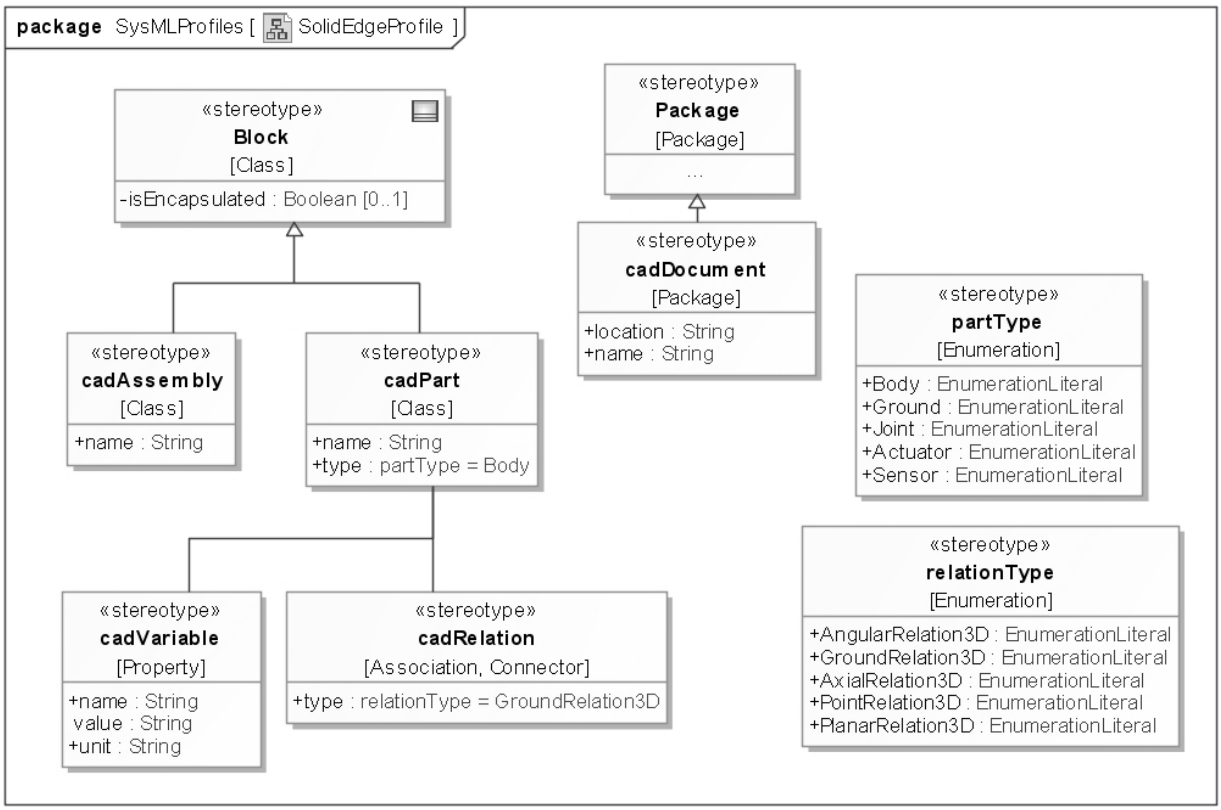

Figure 8 - SysML profile for Solid-Edge

The SysML profile for Solid-Edge (Figure *8*) contains the concepts from Solid-Edge MCAD metamodel. These MCAD concepts are mapped on basic SysML constructs with the help of SysML stereotypes. For instance, the stereotype *cadDocument* is mapped on the concept of *Document* from MCAD metamodel. Similarly stereotypes like *cadAssembly, cadPart, cadVariable* and *cadRelation* are mapped on the concepts of *Assembly, Part, Variable* and *Relationship3D* from MCAD metamodel respectively.

However, the introduction of *Enumerations* such as *relationType* and *partType*, are to enable the modeler to choose from the provided *part* and *relation* types. Although, in the purposed profile, these *Enumerations* contain a few literal values, nevertheless they are extendable and more literals can be added to the *Enumeration,* if and whenever needed.

The SysML profile presented here (*Figure 8*), is created from SysML and UML constructs by considering the concepts of Solid-Edge's MCAD metamodel. Concepts such as cadAssembly, cadPart are directly extended from SysML *Block,* as shown in MCAD metamodel (*Figure 7*); *Assembly* and *Part* can have different *Variables* and *Relations*. A *Variable* can be represented by a *Property* of a *Block* which is shown as cadVariable (*Figure 9*). Similarly, *cadRelation* (*Figure 8*) is extended from the base type *Association* and *Connector* because these types can represent the *Relation3D* between *Parts*. *Property, Association and Connector* both are SysML constructs and are explained in SysML specifications [*9*].

3.2. Model Integration Approach

Modelers can use the SysML profile for Solid-Edge (*Figure 8*), to create system models in modeling tools such as Magic-Draw, Enterprise Architect, etc. Once, system models are created in modeling tools using SysML profile compliant with MCAD meta-model and the detailed design models are created in any MCAD domain specific tool e.g. Solid-Edge.

Then the next step would be to perform model transformation to integrate these models, so that information remains consistent among different models and tools. This process requires a platform which can provide support for model transformation, so that models from one domain can be mapped to another domain in order to perform the model integration between different tools. One such platform is Eclipse Modeling Framework (EMF) [*25*].

EMF is a modeling framework and code generation facility for building tools and applications based on structured data model. Most importantly, EMF provides the foundation for interoperability with other EMF-based tools and applications [*25*]. EMF provides a facility to define meta-models, both by code and graphically (plug-ins can be installed). As described by Ahsan et al. [*11*], the metamodels in EMF are referred as integration models because they support information exchange between models. However, the models created in EMF are referred as Ecore models.

EMF provides the capability to generate Java code for the created Ecore models and also allows to generate plug-ins for the created meta-models, these plug-ins can be imported in order to create models compliant to particular meta-model of the plug-in. Ecore models can also be populated from external inputs e.g. from XMI and output of EMF-Based tools, if the input is compliant to that particular Ecore model. In order to utilize the features and capabilities of Eclipse Modeling Framework (EMF), Ecore model for *SysML profile for Solid-Edge* (*Figure 9*) and Ecore model for *Solid-Edge* (*Figure 10*) are created. Both these Ecore also contains similar concepts as they were discussed before (section-3.2) but these figures are important to understand the structure of Ecore and generated XML files.

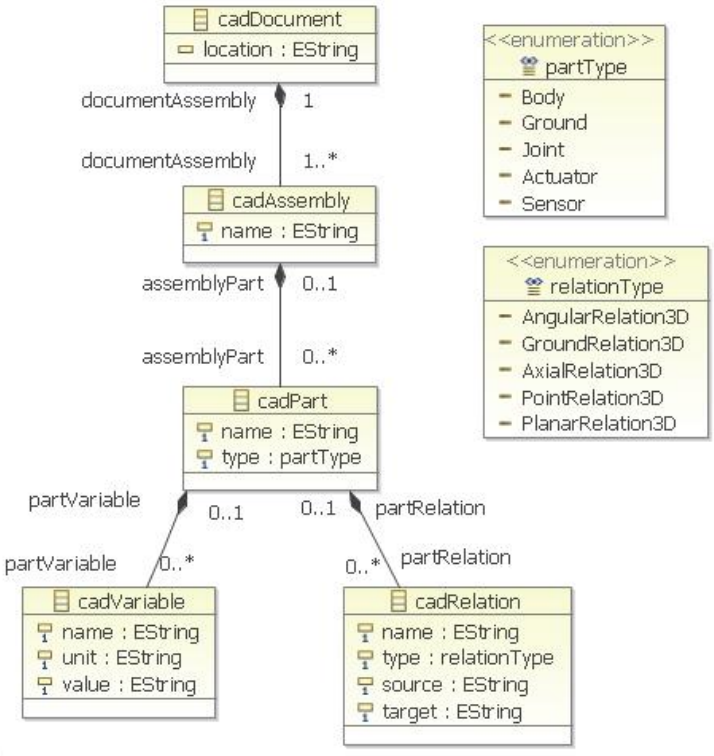

Figure 9 – SysML profile for Solid-Edge Represented in Ecore

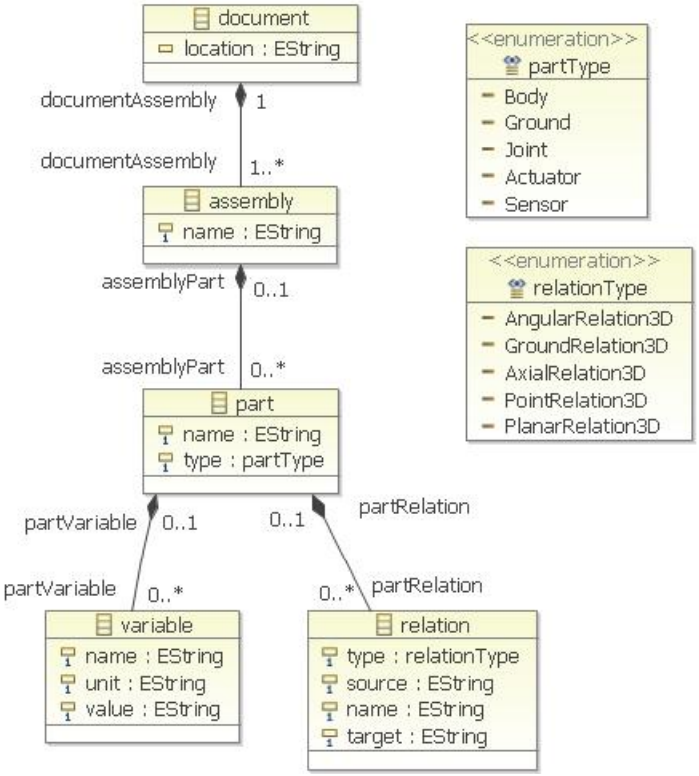

Figure 10 – Solid-Edge Meta-model Represented in Ecore

3.3. Extraction of model information

Once models are created in SysML modeling tools and domain specific modeling tools, and once Ecore models are also defined in EMF, then it comes the need to extract actual model information from these tools so that it can be used to populate the Ecore models and model transformations can be performed. However, the extraction mechanism of model information from a tool is totally dependent on the support provided by the tool and tool vendors.

Most of the modeling tools support model export to XMI format. However, the exported file formats do not conform with each other and if they do then there are limitation such as data loss and readability issues with XML and XMI [*13*,*14*]. Hence, there is a need of having some sort of a connecter or plug-in to extract model information from the modeling tools and to export that information for populating Ecore. Tool vendors usually provide such plug-ins for their modeling tools whereas different third parties also provide some powerful connectors and plug-ins. Moreover, customized plug-ins can also be developed by using tool specific APIs provided by tool vendors. However, developing a customized plug-in requires a lot of time and programming effort.

In this thesis work, a plug-in for eclipse framework called Cameo Workbench [*26*] is used to connect with Magic Draw, in order to inject and extract the model information from Magic-Draw. Cameo Workbench is based on EMF and provides the capability for model transformation. Cameo Workbench supports both Atlas Transformation Language (ATL) [*21*] and Model Query Language (MQL) [*22*] as well. Cameo Workbench also provides the capability to extract Magic-Draw models and populates an Ecore model complying with UML.2.1 metamodel. This populated Ecore model contains all information regarding the system model created in Magic-Draw along with tool specific information that is required to maintain the models inside the tool.

Every modeling tool associates some extra (tool specific) information with the models created inside those tools, this extra information also includes environment settings and the information about the drawing area inside the tool, where a modeler can drag-drop the language constructs to create a model. Cameo Workbench manages all the communication with Magic-Draw and it also allows to create Magic-Draw models from an Ecore source complying with UML.2.1 metamodel.

On the other hand, a simple tool specific connecter, written in Visual Basics .Net (VB parser), is used to extract out and inject in the model information into Solid-Edge. This parser is based on the API provided by Solid-Edge to communicate with the software. This parser can read a model from Solid-Edge, extracts only model specific information, and creates an XML file as an output which is compliant to Solid-Edge Ecore. This parser is also able to parse the XML, compliant to Solid-Edge Ecore, to update the content of a Solid-Edge model.

The Visual Basic parser consists of the following simple interface (*Figure 11*). The *UpdateXML* button is used to read the model from Solid-Edge and to update the XML file according to SolidEdge metamodel. Whereas the *UpdateCAD* button is to update the model in Solid-Edge according to the XML file. The *Exit* button is to close the the parser application. The main area of the parser is used to display the model

information read from the Solid-Edge. The VB parser used in this thesis work is an updated version of the VB parser developed by Ahsan Qamar during his research work as explained in [*27*]. The mechanisum in which VB parser generates the output XMI after reading the Solid-Edge model has been updated to adopt the format of Solid-Edge meta-model. Accordingly the mechanisum of reading the XMI to inject the model information back to Solid-Edge has been modified as well.

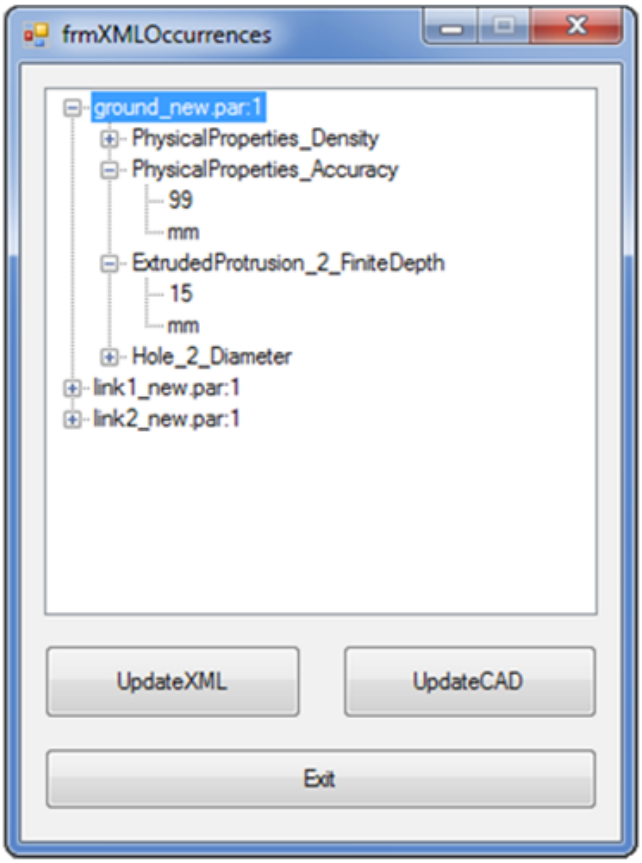

Figure 11 - VB Parser/Connecter UI

3.4. Model transformation

Once Ecore models are created and populated using the tool specific parsers, the next step is to perform the actual transformation between models. Transformation is performed by defining a mapping between source and target metamodels, this mapping explains the links between both metamodels and the concepts contained in those metamodels. As described by Shah et al. [*14*], transformations are defined in declarative fashion at the metamodel level and then are compiled into an executable that performs the transformations at the user-model level. One important thing to note here is that transformations are based on the mapping defined on metamodels; hence they are not model

dependent and once a transformation is defined it would be an automated process to transform models, compliant to those particular metamodels.

In the proposed framework two different transformation languages are used i.e. ATL and MQL. The MQL transformation rules are used to for transforming UML.2.1 metamodel into SysML profile for Solid-Edge metamodel. As highlighted in Cameo Workbench's documentation [*22*] MQL's imperative rule-sets provide conditional and loop statements that simplifies the complex transformations which could be difficult to implement in a declarative manner. The reason for choosing MQL for transformation is to simplify the overall transformation process and to extract only the required information, because *Cameo Workbench* populates UML.2.1 metamodel, which not only extracts the system model information but also contains all the tool specific information that is required by Magic-draw to manage the system models created by the modelers. After applying MQL transformation rules in Cameo Workbench an XML is generated that is complaint with SysML profile for Solid-Edge metamodel.

Once Magic-Draw model is transformed into a model complying with SysML profile for Solid-Edge and the relevant Ecore model is populated. Then according to the *Figure 12*, on the next step, another transformation ATL is used to transform the SysML profile compliant model into a model compliant to Solid-Edge Ecore. The ATL transformation results in to a model complaint to Solid-Edge Ecore and hence, this model can be parsed by the VB. parser to update the model information in the Solid-Edge. One important thing to note here is that, these parsers and transformations are bi-directional. Therefore, the changes made in the design model using Solid-Edge can be reflected back to Magic-Draw, using the same process in reverse order.

3.5. Approach – Implementational View

The whole integration approach presented in the thesis is summed up in the Figure *12*. Figure *12* depicts that on one end there is a SysML modeling tool i.e. Magic-Draw and on the other end, there is a domain specific tool i.e. Solid-Edge for MCAD modeling. Cameo Workbench is used for communicating with Magic-draw and to extract model information in UML.2.1. This UML.2.1, model is transformed using MQL into a model compliant to SysML for Solid-Edge Ecore model.

As discussed in 3.4, an ATL transformation is then used to transform XML compliant to SysML for Solid-Edge metamodel into a model compliant to Solid-Edge metamodel. The XML complying with Solid-Edge metamodel is finally parsed by the VB parser to update the information in Solid-Edge.

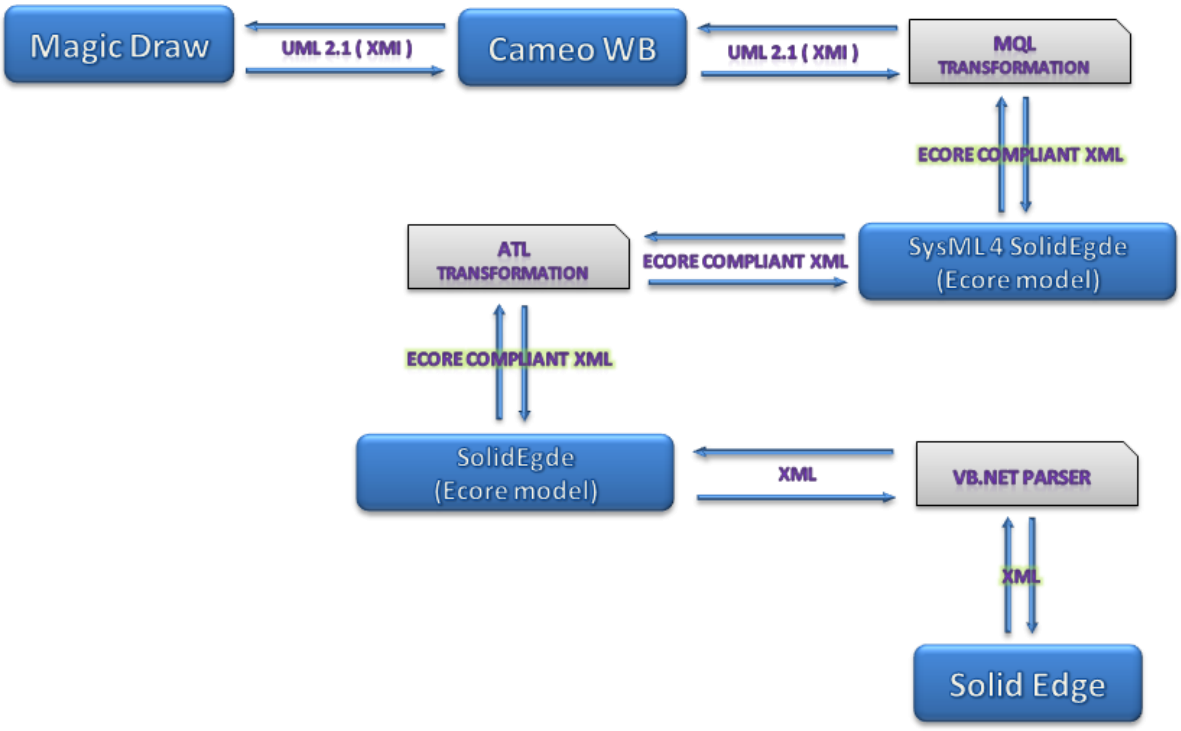

Figure 12 – Implementational view of the purposed approach for Model Integration

3.5.1. Solid-Edge to Magic-Draw

In order to perform model integration from Solid-Edge models to Magic-Draw models, following steps would be required.

1. Create a Solid-Edge (SE) model
2. Use Visual Basic (VB.Net parser) to extract the model information from SE in the form of Solid-Edge metamodel/Ecore compliant XMI.
3. Use ATL transformation, to convert "Solid-Edge" metamodel/Ecore compliant XMI into "*SysML profile for Solid-Edge*" Ecore compliant XMI.
4. Use MQL, to transform "*SysML profile for Solid-Edge*" metamodel/Ecore complaint XMI into UML 2.1 XMI compliant to Cameo.
5. Use Cameo, to load UML. 2.1 XMI in Magic-Draw (MD) to generate model from the provided information

3.5.2. Magic-Draw to Solid-Edge

In order to perform model integration from Magic-Draw models to Solid-Edge models, following steps would be required.

1. Create a Magic-Draw (MD) model using "*SysML profile for Solid-Edge*"
2. Use Cameo, to extract UML 2.1 XMI of the MD model
3. Use MQL, to transform UML 2.1 XMI into "*SysML profile for Solid-Edge*" metamodel/Ecore compliant XMI
4. Use ATL transformation, to convert "*SysML profile for Solid-Edge*" Ecore compliant XMI into "Solid-Edge" metamodel/Ecore compliant XMI
5. Use Visual Basic (VB.Net parser), to parse the converted XML and load the model into Solid-Edge.

This page has been left blank intentionally

Chapter 4 – Integration Case Study

In this chapter a robot design model is used to show how the purposed model integration approach would work on actual models. The robot model is created in Solid-Edge as used by Ahsan in his research work [*27*]. This chapter elaborates how the robot model has been successfully integrated with the corresponding system models created in Magic-Draw.

4.1. Model Structure in Solid-Edge

To exemplify the purposed approach, a simple robot model is created in Solid-Edge (*Figure 13*). This robot model has two parts and a ground. MCAD modeling in Solid-Edge is done in a 3d-Space, therefore, a ground is considered as a reference point or the base point for any model created in Solid-Edge. The robot parts and ground are linked together by *assembly relations* as discussed in 3.1. The robot parts also have different properties associated to them. These properties and relationships are not visible in the *Figure 13,* but they are shown in *Figure 14.*

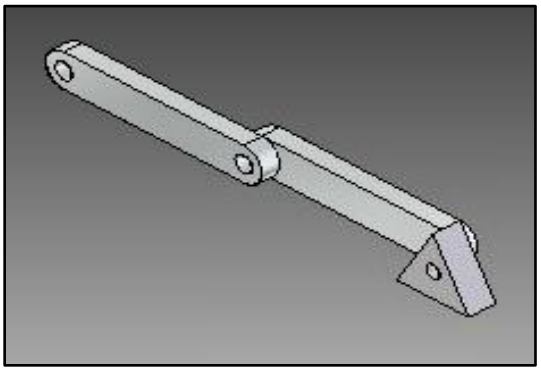

Figure 13 - Robot Structure Model

4.2. Parsing Solid-Edge Model

Figure 14 shows the XML generated by the VB parser, after reading and parsing the Robot Model (*Figure 13*) from Solid-Edge. This XML is fully compliant with the *Solid-Edge Ecore Model* (*Figure 10*) and contains only the model specific information. The XML nodes *assemblyPart* are showing of the parts that are in Robot Model. XML nodes *partVariable* are showing the properties associated to each part. Name, value and unit for each *partVariable* are also retrieved while parsing the robot model from Solid-Edge. Similarly, *partRelations* are also shown here. One important thing to note here is that *relationType and name* of the relation is associated with one attribute *name* in the XML, in order to avoid redundancy of similar information in the XML.

Figure 14 – Solid-Edge Ecore Compliant XML generated by VB Parser

4.3. Using ATL & MQL transformations

Once the model information is extracted from Solid-Edge into an XML compliant to Solid-Edge Ecore as shown in *Figure 14*, then the ATL transformation is performed to transform the extracted model information into an XML compliant to *SysML profile for Solid-Edge Ecore*. After the ATL transformation is applied, the resultant XML can then be used in MQL transformation. MQL transformation is applied in order to transform the *SysML profile for Solid-Edge* compliant XML into *UML.2.1* XMI.

Cameo Workbench, which is a connector for Magic-Draw, manages all the model information in the form of UML.2.1. XMI. Hence, it is required to transform the XML compliant to *SysML profile for Solid-Edge* into UML. 2.1. XMI, in order to create corresponding model in Magic-Draw while using Cameo Workbench a connector. Since, the purpose of using Cameo Workbench is to communicate with Magic-Draw in order to create a model by processing the extracted model information, this model also needs to be profiled.

Hence, while performing the MQL transformation an empty Magic-Draw project containing the *SysML profile for Solid-Edge* is used to provide the profile information for the Robot Model. All the tool specific extra information would also be gathered from the empty Magic-Draw project used while performing the transformation.

Figure 15 shows the structure of Cameo Workbench project with MQL transformation rules. In *Figure 15*, *SysML4SolidEdgetoUML21.mqr* is a file containing transformation logic to transform the model complying with SysML profile for SolidEdge metamodel into UML2.1, so that it can be used to while injecting model information in Magic-Draw. Similarly, *UML21toSysML4SolidEdge.mqr* is a file containing transformation logic that uses the model information in UML2.1, extracted from Magic-Draw, to transform it into an XMI complying with SysML profile for SolidEdge metamodel.

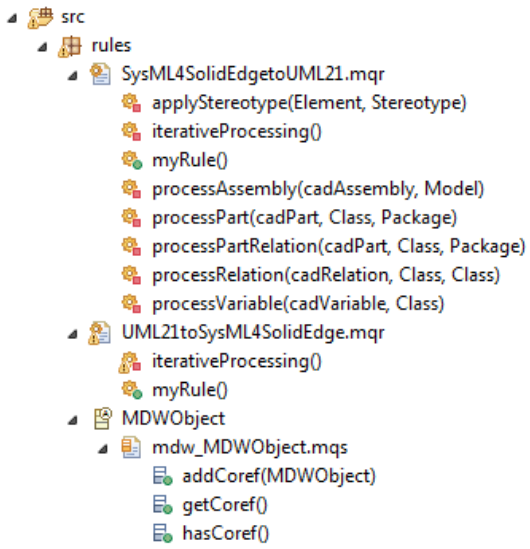

Figure 15 - MQL transformation rules in Cameo Workbench

Figure 16 - MQL ruleset declaration shows the MQL ruleset declaration syntax, the *inout* parameter *output: uml21* is basically referring to the empty Magic-Draw project, which would fill by the end of transformation.

```
public ruleset sysML2Uml(in input : sysml4solidedge, inout output: uml21 )
```

Figure 16 - MQL ruleset declaration

4.4. Loading Magic-Draw Model

Once the MQL transformations are executed successfully they results in the creation of the *Robot Model* in the empty Magic-Draw project. *Figure 17* depicts the successfully loaded view of Robot model in the Model Repository of Magic-Draw. *Figure 17* shows a package RobotModel, containing all the parts of Robot Model along with the *RobotAssembly* which shows that these three parts are the parts of one assembly.

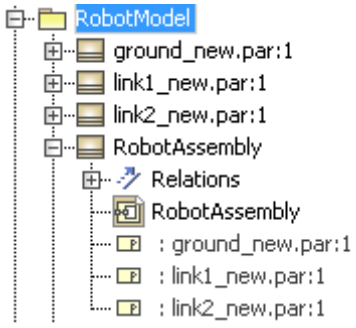

Figure 17 - Robot Model in Magic-Draw Repository

Figure 18 - Robot Model with applied stereotypes (CAD Concepts)

Figure 18, shows how the robot model would be displayed in a block definition diagram, with the stereotypes applied during transformation. Hence the Robot model created in Solid-Edge is successfully imported in Magic-Draw, now any changes made in Magic-Draw view of the robot model can be reflected back in the Solid-Edge by following the Steps mentioned in section-3.5.2. For instance, as highlighted in *Figure 19*, the value of *Link2_length* has been updated from "*0,12:mm"* to "*0,05:mm".* After following the steps mentioned in section-3.5.2, the robot model in Solid-Edge will be updated and the part *link2_new.par:1* will be updated with the new length value as shown in *Figure 20*.

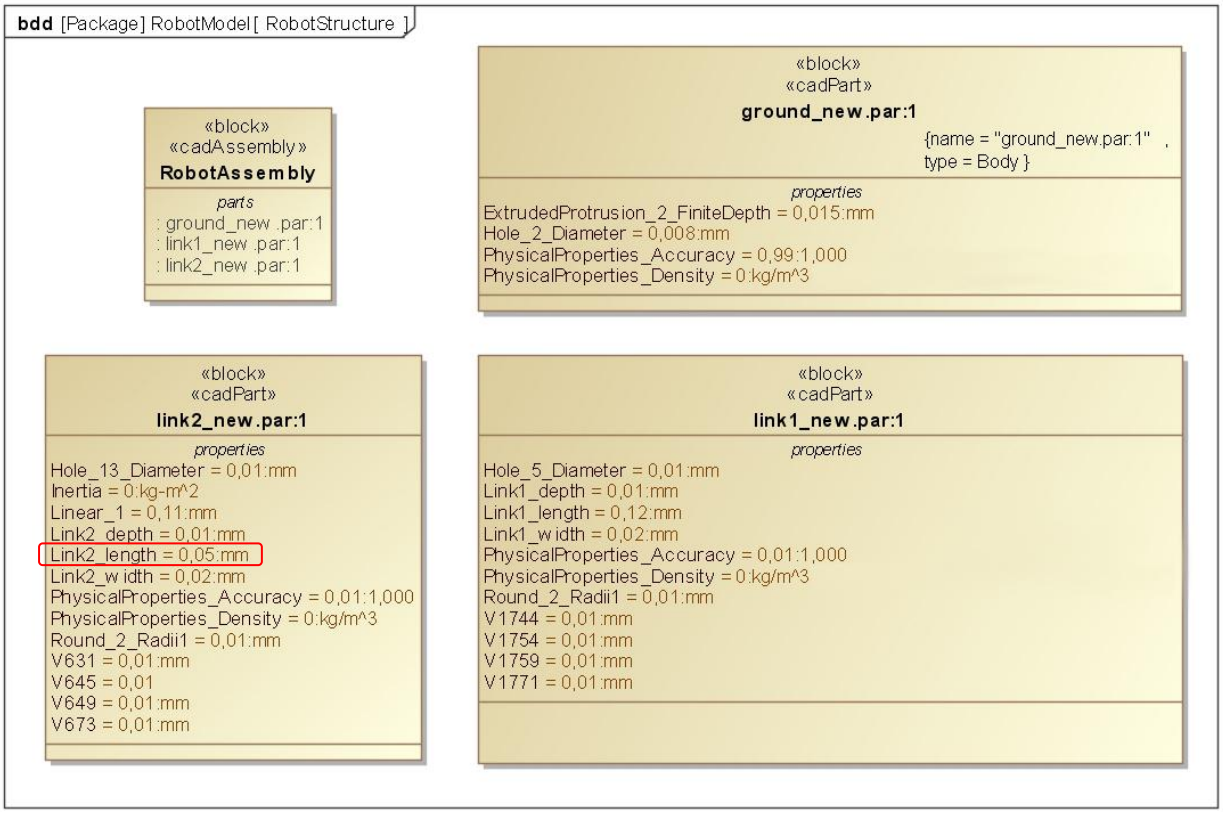

Figure 19 - Updated Robot Model in Magic-Draw

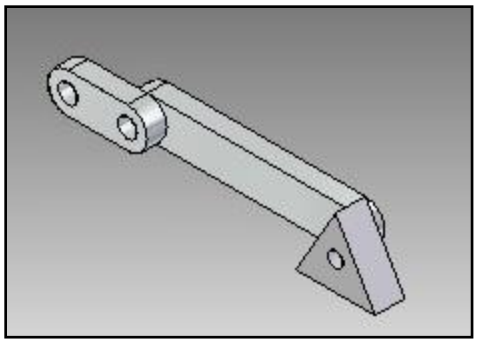

Figure 20 - Updated Robot Model in Solid-Edge

This back and forth transformation of model information makes aids in the process of modeling by providing the model integration of SysML profiled Models into domain specific Solid-Edge models. By following the presented approach the changes made in a model from one tool can be reflected into the other tool. This will make the designers aware of the changes and will result into consistent design information among different domains.

Chapter 5 – Closure Analysis & Future Work

This chapter provides a concise review over the model integration approach presented in this thesis. Moreover, the limitations of the presented approach are also highlighted here. This chapter also points out the areas in which future work can be done to enhance and improve the presented approach.

5.1. Analysis and Remarks

The main motivation behind this thesis work is to find out effective ways to support the analysis and the design process in mechatronic design. The main focus here is on the integration of *design* and *analysis* models created in SysML and MCAD domain. The purposed approach integrates models created in different tools; Solid-Edge & Magic-Draw, with different level of abstractions while attempting to support the co-evolution of mechanical design.

As discussed in *Chapter-2*, many people have presented different approaches of model integration using SysML but this thesis specifically focuses on the integration of *system-level design* and *mechanical design* models in the development of mechanical systems. A well known model integration framework and declarative model transformation languages such as ATL and MQL have been used as the base of model integration approach presented here. Tool specific parsers, such as VB parser for Solid-Edge and Cameo for Magic-draw, are used to extract and slot-in the model information from/into modeling tools.

One important thing to be considered here is that, although SysML and MCAD models are considered in this thesis, but in principle this approach can be used and extended towards other modeling formalisms. The approach is explained in *Chapter-3* and the robot example presented in *Chapter-4* shows how the proposed approach can be used to effectively integrate models created in different tools. The robot example also depicts how the proposed approach helps in maintaining the model information among co-evolving models throughout different design phases.

In principle the presented approach can be useful for the integration of different kinds of models. However, the actual effectiveness of the approach in dynamic and complex industrial scenarios is yet to be evaluated. This industrial evaluation would require testifying this approach on actual large scale industrial problems consisting on many huge and complex models. Feedback from professional modelers and designer from industry would also be beneficial to improve the presented approach.

5.2. Limitations of the Work

The work presented in thesis has not been evaluated on large scale design projects and professional designer and modelers have not evaluated its effectiveness yet. However, there are some obvious limitations, mainly from the implementation perspective.

5.2.1. Limitations of VB parser

One of the limitations is maturity level of the Visual Basic parser which is used to connect with Solid-Edge and extract the model information from the Solid-Edge. Although, the Visual Basic parser has been developed by using API provided by Solid-Edge but due to the time constraints, it lacks certain features and has room for improvements. The implementation of the parser lacks to extract the information regarding the "unit" of a MCAD *property* extracted from Solid-Edge. Similarly the VB parser also lacks the implementation of extracting the *PartType* of the parts from Solid-Edge.

One of the major limitations, apart from the above mentioned minor limitations, of the VB parser is that for now, it is not capable of creating new Solid-Edge models from scratch and also any new Solid-Edge construct cannot be added in the existing Solid-Edge model using the VB parser. For instance, with current implementation of VB parser, it is not possible to create a new model in Solid-Edge by just using the model information extracted from Magic-Draw because the VB parser cannot manage all the environment settings require in creating a new model. However, Cameo which is used to connect with Magic-Draw is a professional and powerful parser which has the capability to manage the environment settings required to create models from scratch in Magic-Draw.

To enable the VB parser for managing environment settings of Solid-Edge, it requires a lot of programming efforts. Moreover, through testing of the Visual Basic parser is required to make it enough reliable to parse any kind of XML complaint to Solid-Edge metamodel. Current, implementation can produce runtime errors while parsing XML with missing values.

5.2.2. Limitations of using SysML Profile for Solid-Edge

One fundamental limitation of this approach is due to the language mapping between SysML and MCAD domain. SysML constructs are different then MCAD constructs therefore a SysML profile is used to map SysML constructs on MCAD concepts. SysML contains a lot of constructs and features to support Model Based System Engineering as described in SysML specification [*9*] but the modelers are required to use the SysML profile for Solid-Edge in order to perform the model integration.

Similarly the metamodels used for transformations are kept simple according to the constructs of SysML profile for Solid-Edge. Any change in metamodels would require an update in the transformation logic and it could also require some changes in VB parser to accommodate the metamodel changes.

5.2.3. Limitations in Model Generation and Representation

There is one limitation while generating Magic-Draw models from scratch, the current implementation only considers the model information and processes it while applying ATL transformation from SysML to UML 2.1 metamodel. Hence, if it is required to create a SysML model from scratch in Magic-draw on the basis of model information extracted from Solid-Edge this can be achieved by using the current implementation. However, the current implementation would place all the constructs in the model repository (*Figure 17*) of the Magic-Draw project and modeler will need to create a corresponding diagram manually to view the model in a proper form as shown in *Figure 18*.

5.2.4. Limitations in Automation

The work presented in this thesis involves different tools and platforms (e.g. Solid-Edge, VB parser, Eclipse, Cameo, Magic-Draw, MQL and ATL) to perform the model integration process. The inclusion of all these frameworks and tools itself requires some integration to make the whole process fully automated. Due to the scope of this thesis the communication between all these frameworks & tools has been done manually. For example, the project input and output files paths have been be configured manually in the Cameo, MQL, ATL and VB parser. Similarly the paths for XML output files are adjusted manually. Moreover, the transformation rules in Cameo, and VB parser also requires to be executed step by step manually.

5.3. Future work

The main goals of the thesis have been successfully accomplished. However, there is always some room for improvements, particularly some enhancements in the process can be done by resolving the limitations discussed in section-5.2.

One major improvement in this process can be done by enhancing the capabilities of the VB parser to support the lacking features. Moreover, the whole process of integration presented here requires certain manual inputs such as, it is requires to specify the input & output file paths and the transformations are required to executed manually. All these manual inputs can be automated so that the whole transformation process can be used as plug-in and the model information could be exchange by a simple click.

Furthermore, the presented approach can be evaluated on complex industrial scenarios to assess its impact and usefulness in industry. Assistance from professional designer and modelers can also help to identify the areas of improvements in the presented approach.

5.4. Closing Remarks

This thesis is an attempt to provide a solution for integrating system-level design and mechanical design models in the development of mechanical systems. A SysML profile for MCAD domain is created to map the mechanical concepts used in MCAD on SysML constructs. The purposed profile is compliant to MCAD metamodel supported by Solid-Edge. ATL and MQL transformation are used to integrate SysML and MCAD models along with the parsers and connectors (VB parser and Cameo) used for modeling tools (Solid-Edge and Magic-Draw).

The work presented in this thesis not only provides an integration mechanism for SysML and MCAD models, but in principle it can be extended to integrate models form other domains as well. Hopefully, this approach would provide guidance for other researchers attempting to improve the support for model integration and MBSE in general.

References

[1] Creative Commons. (2011, March) Images to modify, adapt, or build upon, http://search.creativecommons.org/.

[2] A. Andrew, P. Sage, and J.E., *Introduction to Systems Engineering*.: John Wiley & Sons, Inc., 2000.

[3] T. Tomiyama, V. D'Amelio, J. Urbanic, and W. ElMaraghy, *Complexity of multi-disciplinary*, 56th ed.: CIRP Annals – Manufacturing Technology, 2007.

[4] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*.: The MK/OMG Press, 2008.

[5] *VDI 2206. Entwicklungsmethodik für mechatronische Systeme (Design methodology for mechatronic systems)*. Berlin: Beuth Verlag, 2004.

[6] T. A. Johnson, "Integrating Models and Simulations of Continuous Dynamic System Behavior into SysML," Georgia Institute of Technology, Atlanta, Georgia, USA, Master's Thesis 2008.

[7] Object Management Group. (2010, Nov) Unified Modeling Language™ (UML®). [Online]. http://www.omg.org/spec/UML/

[8] Object Management Group. (2010, Nov) OMG Systems Modeling Language Specification. [Online]. http://www.omg.org/spec/SysML/1.2/PDF/

[9] SysML. Open Source Specification Project. [Online]. http://www.sysml.org

[10] S. Friedenthal, A. Moore, and R. Steiner, "SysML tutorial base line to INCOSE," 2006.

[11] A. Qamar, J. Wikander, and C. During, "A Mechatronic Design Infrastructure Integrating Heterogeneous Models," Department of Machine Design, School of Industrial Engineering and Management, KTH - Royal Insitute of Technology, Stockholm, Sweden,.

[12] A. M. Geoffrion, "STRUCTURED MODELING: SURVEY AND FUTURE RESEARCH DIRECTIONS," in *Interactive Transactions of ORMS 1*, 1996.

[13] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development," Software, IEEE. 2003.

[14] A. A. Shah, A. A. Kerzhner, D. Schaefer, and C.J.J. Paredis, "Multi-View Modeling to Support Embedded Systems Engineering in SysML," Systems Realization Laboratory, G. W. Woodru School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA, 2010.

[15] K. Czarnecki and S. Helsen, "Feature-based survey of model transformation approaches," *IBM Systems Journal*, vol. v.45 n.3, pp. p.621-645, July 2006.

[16] A. Pop, D. Akhvkediani, and P. Fritzon, "Towards Unified System Modeling With ModelicaML UML profile," in *International Workshop on Equation-based Object-Oriented Languages and Tools, Linköping University Electronic Press*, Berlin, Germany, 2007.

[17] T. A. Johnson, C. J.J. Paredis, J. M. Jobe, and R. Burkhart, "Modling Continous System Dynamins in SysML," 2007.

[18] C. Nytsch-Geusen, "The Use of UML within the Modeling process of Modelica-Models," in *International Workshop on Equation-based Object-Oriented Languages and Tools, Linköping University Electronic Press*, Berlin, Germany, 2007.

[19] K. Chen, J. Bankston, J. H. Panchal, and D. Schaefer, "A Framework for the Integrated Design of Mechatronic Systems," Systems Realization Laboratory, The G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Savannah, GA 31407, USA, 2009.

[20] A. A. Shah, D. Schaefer, and C. J.J. Paredis, "Enabling Multi-View Modeling With SysML Profiles and Model Transformations," in *International Conference on Product Lifecycle Management*, 2009.

[21] Eclipse Foundation. ((2011)) Atlas transformation language (ATL). [Online]. http://www.eclipse.org/m2m/atl/

[22] Cameo Workbench. (2011) MQL - Cameo Workbench Documentation. [Online]. http://127.0.0.1:3749/help/index.jsp?topic=/com.sodius.mdw.platform.rdt.doc/reference/mql.html

[23] No Magic. (2011) Magic draw UML/SysML. [Online]. http://www.magicdraw.com

[24] Siemens PLM Software. (2011) Solid Edge - Design Bette. [Online]. http://www.plm.automation.siemens.com/en_us/products/velocity/solidedge/index.shtml

[25] Eclipse Foundation. (2011) Eclipse modeling framework (EMF). [Online]. http://www.eclipse.org/modeling/emf/

[26] No Magic and Sodius. (2011) Cameo Work Bench. [Online]. http://www.magicdraw.com/cameoworkbench

[27] A. Qamar, *An Integrated Approach Towards Model-Based Mechatronic Design*. KTH - The Royal Institute of Technology, Licentiate Thesis. ISBN:978-91-7501-025-0. Stockholm, Sweden, 2011.

This page has been left blank intentionally