DESIGNING MECHATRONIC SYSTEMS: A MODEL-INTEGRATION APPROACH

Ahsan Qamar[1], Jan Wikander[1] and Carl During[2]
(1) KTH-Royal Institute of Technology, Sweden (2) Micronic Laser Systems, Sweden

ABSTRACT
Development of mechatronic products requires different types of design models in order to support both domain-independent specifications and domain-specific principles. This research aims to find out how *system-level modeling* can support mechatronic design, and how the integration of system-level modeling and domain-specific modeling can be supported during different design phases. A design example of a hospital bed's propulsion system is presented to show firstly the relationship between conceptual design and system-level modeling, and secondly the need for integration of system level and domain specific design models. An integrated modeling and design infrastructure is proposed to support abstraction between mechatronic design models, hence supporting co-evolution of design models. The paper concludes that a mechatronic design problem can be better supported through such an integrated design approach. However, usability of this approach needs to be further supported by more case studies in the future

Keywords: Mechatronics, system design, design infrastructure, model integration

1 INTRODUCTION
Technical systems today have to support a large number of functions at reduced cost. The efficiency and cost effectiveness gained by implementing the required functionality through electronics and computer software has become a major driver towards development of mechatronic products. These products are characterized by increased integration of mechanics, electronics, and computer software, requiring companies to establish cross-disciplinary teams consisting of several domain-experts. Tomiyama et al. [1] state that multi-disciplinary product development (as in mechatronics) introduces difficulties such as the need for an inter-disciplinary design language, how to deal with different stakeholders, and how to deal with inter-disciplinary design problems. Mechatronic design requires a collaborative effort between different domain experts within the cross-disciplinary team. However, each domain contains different collections of design methods and tools. An overall mechatronic design method is yet to be conceived. As engineers do not necessarily possess cross-domain knowledge, it is difficult for an individual to understand the inter-disciplinary problems, especially in the context of complex products. Therefore, communication between involved domains is essential throughout different design phases in order to avoid integration problems in mechatronic product development. In addition to this, technological advancements and increased functionality contribute to high complexity in designing mechatronic products.
Modeling is an important design activity, where modelers try to gain information about consequences of their decisions early in design. Buur and Andreasen [2] state that the success of a mechatronic design project depends especially on the ability of the designers to communicate and visualize their ideas to the rest of the group. Design models permit a designer to describe his or her thoughts for better understanding, both individually and by the group. Depending on the design stage, a design model can be abstract or detailed. A design model should be carefully developed to model only the product properties necessary at the current design stage [2]. This restriction in scope is necessary: firstly, since information about a design problem increases through different phases of design, secondly, because a design model with too many product properties will become unnecessarily complex to serve the purpose for the designer. Hence, product design is based on different design models reproducing different product properties. In a mechatronic design scenario, design models vary between different domains. Some of these models define and describe the product from the domain perspective such as mechanics or electronics; others are used to evaluate product properties within a domain such as dynamic analysis of a mechanical design.

The partitioning between different mechatronic domains is laid very early in product design. By defining function principles to the function structure, the designer allocates different technologies to the product function/sub-functions (*design concept*). However, this often leads to an isolated development within a domain, and optimization of the individual modules, rather than the complete system. Buur et al. [2] and Gausemeier et al. [3] state that specifying a mechatronic *design concept* requires a new design model. Buur et al. [2] define the new model types to support abstract function structure independent of a technology, function principles supported through different technologies, and specification of the interfaces between different technologies. Gausemeier et al. [3] utilize a new modeling language for supporting abstract function structure, domain-spanning function principles, and interfaces.

One of the main ideas behind the research treated in this paper is to utilize design models that are suitable to capture domain-independent specifications. At the same time, the designer can apply multi-technological function principles to the *design concept.* The domain-specific models evolving from these multi-technological function principles can be integrated with the domain-independent model, the other main proposal of this paper.

Since design evolves not only from one design phase to the next, but also in between domains, it is important to support abstraction between models of different engineering domains. Moreover, it is also important to keep the design models consistent with each other if a certain design model is modified at a certain design stage. One approach towards achieving this abstraction is through model transformations. Some examples of integrating design models through model transformations are presented in [4] and [5]. Consistency between mechatronic design models has been discussed in [6]. While these approaches extend the software engineering principles towards model-based development, this paper aims to explain the relationship between design models within a mechatronic design problem, and proposes a solution to multi-domain model integration. The paper utilizes a design study of a servo-propelled hospital bed performed at the Department of Machine Design at the Royal Institute of Technology, Sweden, to answer questions such as:

1. How to establish an initial domain-independent design specification through a design model, and obtain a complete system view required in a mechatronic design problem?
2. How to establish relationships between domain-independent and domain-specific design models?
3. How to integrate different design models developed at different design stages?

The remaining part of the paper is structured to answer the above questions as follows. Section 2 discusses model-based design in relation to engineering design methods. The section proposes a solution for solving the communication problem between domain experts in a mechatronic context (question 1). Section 3 presents the design study of the hospital bed system highlighting the conceptual design phase where domain-specific design models were also utilized. Section 4 is about answering the question: how to integrate models (question 2), and how better design can be achieved through model integration, supported by a small example (question 3). Section 5 concludes the paper, including a discussion on proposed future work.

2 MECHATRONIC DESIGN AND MODEL-BASED DEVELOPMENT

Model-based engineering (MBE) is about elevating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system [7]. The systematic design approach from Pahl & Beitz [8] shows three main product design phases: conceptual design, embodiment design, and detailed design. During these design phases, models increase in detail with the passage of time, and abstraction between models needs to be supported in order to manage the modeling process as a whole.

Design activities for a mechatronic system are typically performed by a multi-disciplinary team of domain experts. Different design methods (available in different domains) are followed by the domain experts, who are unlikely to possess inter-disciplinary knowledge to get a detailed enough understanding of the whole design problem. Therefore, it is difficult to establish a common mechatronic view; rather different domain views are established and the dependencies in-between are not clear. Hence, it is necessary to establish some means of communication between such views in order to avoid integration problems. Frey et al. [9] classify the communication between two design-domains in terms of communication possibilities between persons, between methods, between models, and between analysis tools. We argue that for a mechatronic design problem, two possibilities can be undertaken to attack the communication problem:

1. A mechatronic design methodology could be followed. VDI2206 [10] introduces such a design methodology. However, it does not cover the management of dependencies between mechatronic design domains. Also, the means to support abstraction between system-design and domain-specific design (per VDI2206) are lacking. A vertical abstraction adds detail to a model or reduces it, while a horizontal abstraction typically takes place between models of the same detail, often in different domains. As highlighted by Buur et al. [2], new mechatronic design models are needed that support both domain-independent and multi-domain modeling capability. Some modeling languages, such as Modelica [11] and MapleSim [12] support the creation of multi-domain design models (spanning different stages of VDI2206). However, domain-independence is also important for the different domain-experts to share a common product view. We conclude here that a single multi-domain modeling language cannot provide a solution for mechatronic design problems, a conclusion also supported by Shah et al. [5]. Such a language is difficult to develop, support, and evolve with time.
2. A framework that supports mechatronic design can be utilized. Here, we can let the domain experts utilize the existing methodologies inside each domain, and provide means of communication between domains through the framework. As suggested by Frey at al. [9], a person-to-person and method-to-method communication is either error-prone, or not directly possible. Even though the communication between analysis tools is possible, however, it is based only on execution of design models and not their development. Therefore, a model-to-model communication is employed during this paper, which allows for both vertical and horizontal abstraction during model-based development.

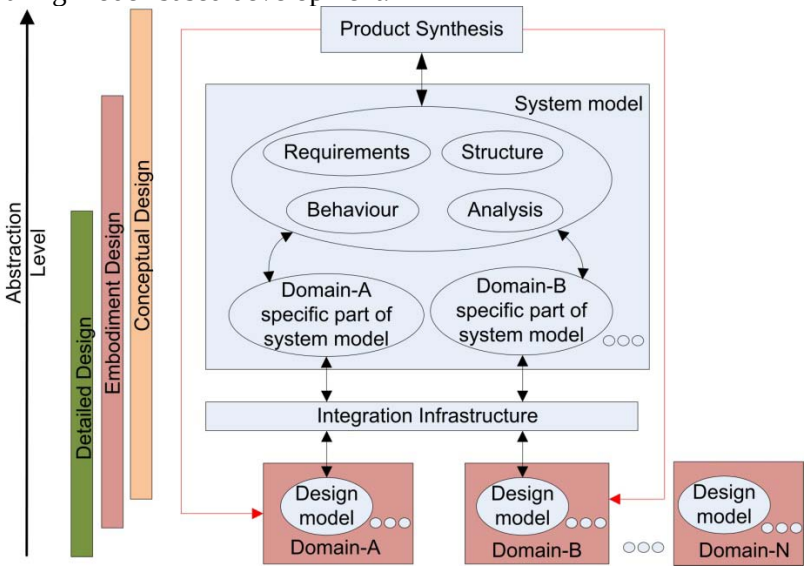


Figure 1. Model—based development in engineering design in the context of an integrated modelling and design infrastructure for mechatronic system design

For communication through multi-domain models in a perspective of engineering design, an infrastructure is required, supporting evolution of design models while design proceeds between different design phases. The infrastructure should support development of domain-independent design models and specification of interfaces between design domains, development of domain-specific design models, and integration between all these design models. Such an infrastructure is vital to integrate multi-domain models, which is a key proposal of this paper, as explained in Section 4.

Conceptual design is a dynamic phase in terms of change in design and interaction of designers. Initial product synthesis – being part of conceptual design – serves as a basis for developing an abstract function structure and corresponding function principles. In a model-based development with model as the primary artifact, it is important to capture the product synthesis information inside models. In agreement with Gausemeier et al. [3], we will use the term *common specification language* to denote a common platform for different domain experts to define and specify a system. Figure 1 shows that a system-model can be developed through a common language, based on the findings of product synthesis phase. Here, the interfaces between chosen *design concepts* can be specified starting in a black-box manner and continuing towards increasing detail. The Systems Modeling Language (SysML) [13] is a general purpose modeling language where a modeler can specify a system to a level

that enables its association to other design models [14]. We utilize SysML to represent the multi-domain function principles through generic SysML constructs. The resulting model can be utilized as a common design model, by building relationships with other design models. The relationships can be built by utilizing the extension capability in SysML to create domain-specific parts of the system model (Figure 1), where the parts are built with concepts of a particular domain. The proposed integration infrastructure allows integration of the domain-specific design models with the domain-specific parts of the system model. Hence, as the design progresses through domain-specific design models, the system model also increases in detail. Design iterations continue to take place between different design stages, and design models consistently evolve through the integration infrastructure (Figure 1). Another approach is to first create domain—specific design models based on product synthesis (red arrows in Figure 1). The resulting design models can be related with each other by transforming them into the system model through the integration infrastructure. We do not strictly propose to follow this order or the order starting with the system model, to leave flexibility for future work.

3 DESIGN EXERCISE: A SERVO-PROPELLED HOSPITAL BED

This section is based on a mechatronic design example in order to better reflect on the following questions (rephrased from questions: 1, 2, and 3 in section 1):

- How to establish relationships between domain-independent and domain-specific design models?
- How to integrate different design models developed at different design stages?

By performing a design exercise on the hospital bed example, we try to identify the needs for the integrated modeling and design infrastructure, and determine how dependencies can be managed through the SysML model. The aim with the exercise is to design an active (driven) wheel module which can be utilized on common hospital beds. The design activities performed within the conceptual design phase are explained in the subsequent sections.

3.1. Conceptual design

During conceptual design, a group of designers were provided with a set of requirements for the hospital bed. The design team consisted of six team members with backgrounds in mechanical engineering, electrical engineering, control engineering, system engineering and computer science. The requirements for the design of the wheel module were classified in different categories on a white board, and used as a reference throughout the initial conceptual design phase. Some of the main requirements for hospital bed are presented below; the rest is omitted for space concerns:

- Bed speed of 2 m/sec on a maximum 5^{o} slope (power, driving requirements)
- Being able to turn on the spot and move in any direction (driving configuration, control)
- Wheel solution packaged as one module (sensor, motors, control, wheel module configuration)

Based on these requirements, the design group discussed the following main points:

- Which configuration of driving and steering (number and placement of active wheels: *propulsion system configuration*) will provide the best solution in terms of drivability, steering, and cost effectiveness. Six different configurations were discussed.
- How to provide a modular solution, yet one that works on current hospital beds. This includes decision on sensors (wired/wireless), power required to move maximum load, battery (central or distributed), battery charging scenarios, and mechanical interfacing.
- Configuration of driving and steering within one wheel module assembly (*wheel module configuration*). Two configurations were discussed, knowing available wheels and information about drive and steering actuators. The braking requirement for each wheel was also discussed to be accounted in the solution.
- A central controller is required to control the driving and steering of the bed as a whole, by controlling the available drive and steer actuators in each wheel module assembly.
- Safety considerations in relation to the use of wireless devices, patient safety and comfort requirements.

During the initial conceptual design phase (lasting for 5 hours), an initial decision was made on some of the concepts. However, some concepts needed further analyses before any decision about them could be made. Some of these concepts were:

- Wheel-module configurations

- Analysis of driving, steering and braking actions in relation to bed movement for six different propulsion system configurations, and two different wheel module configurations
- Controller complexity for different wheel module- and propulsion system- configurations
- Which battery is suitable for the bed, i.e. an analysis of power to weight ratio and cost considerations of batteries
- Relationship of distributed battery and central battery in connection to wired and wireless sensors, and the cost difference for each concept

Different propulsion system configurations led to six possible design alternatives, all providing driving and steering capability. These concepts were compared in a weighted decision matrix (omitted here due to space concerns) against a set of basic design criteria. The decision matrix showed that the propulsion system configuration based on either a diagonal configuration or a fully configurable configuration as the most feasible options. The same result is modeled inside a system-model (Figure 3(b)) which is discussed in the following section.

3.2. System design/modeling

After completing the initial conceptual design, the requirements for the wheel module along with the possible solution concepts were known (working structure). A system model was built based on these findings. The suitability of SysML as a modeling language during the conceptual design phase has been documented in [15] and [16]. This section will present a few snapshots of the SysML model, especially its relation to the performed synthesis, and to the creation of domain-specific models.

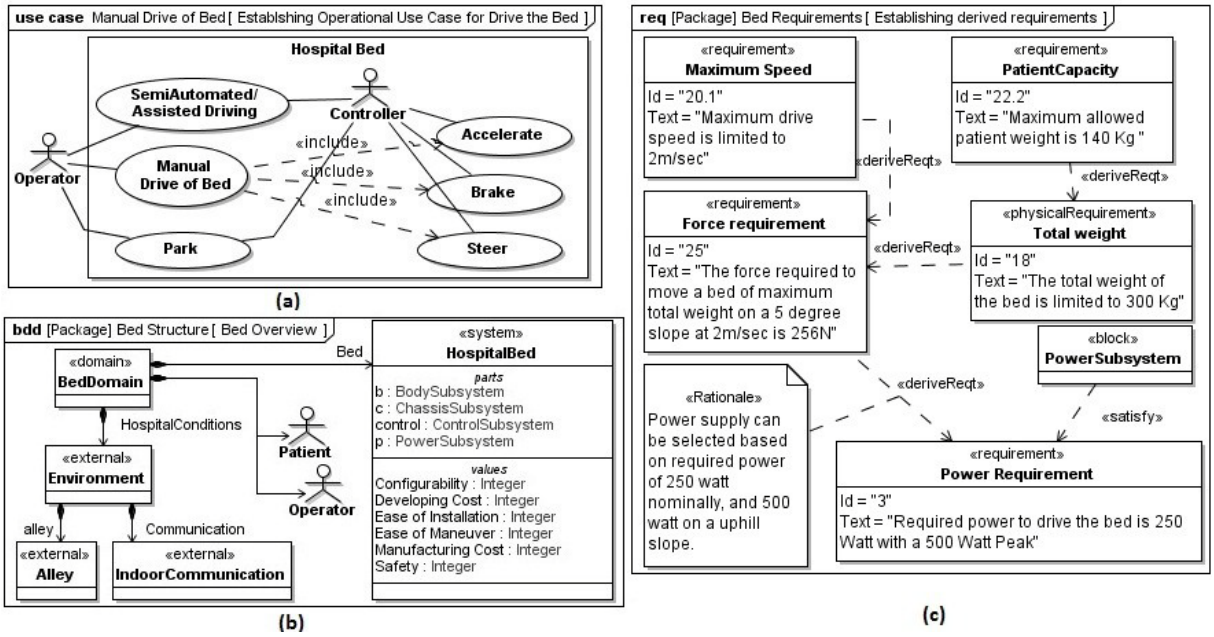


Figure 2. (a) Use cases showing manual and propelled operation of bed. (b) Top-level overview of bed showing main actors. (c) A set of requirements for the power subsystem

Figure 2(a) shows the manual and automated driving use cases performed by the bed operator. The accelerate, brake, and steer use cases are performed by the operator manually. The top-level structure of a hospital bed is shown in Figure 2(b), containing the main actors and the environment in which the bed operates. The *system HospitalBed* contains parts (regardless of technology) mentioned in the *parts compartment*. Figure 2(c) shows some of the derived requirements, in this case for the power subsystem. All models in Figure 2 are developed based on the information obtained through the product synthesis phase of the design exercise. The main structure of the hospital bed propulsion system is shown in Figure 3(a), showing different subsystems. Each subsystem contains a combination of multi-domain components, represented through general purpose semantics. The interfaces between different system components (represented by generic constructs) can also be specified.

The C*hassisSubsystem* in Figure 3(a) contains two to four W*heelModule* blocks. This is based on the conceptual design phase discussion deciding that there will be a minimum of two active wheels on each bed, and each wheel will be enclosed as one complete module with driving and steering capability (*Modular Components Constraint*). The six alternative propulsion system configurations are

compared against stated criteria by a weighted objective function (*WheelModuleObjectiveFunction*) used inside the *WheelModulePerformance TradeOff* block (Figure 3(b)).

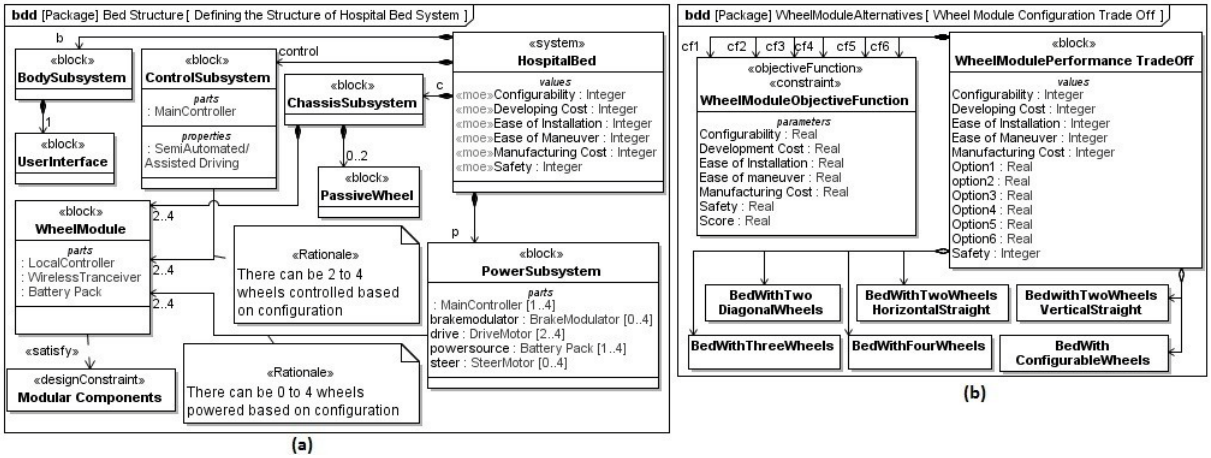


Figure 3. (a) Main Structure of the propulsion system (b) A model of a trade-off study for propulsion system configuration alternatives. The criteria such as configurability etc. are represented as measures of effectiveness (moe) of each alternative.

During the conceptual design phase, a discussion was made about off the shelf wheel-motors, only requiring a steering mechanism to provide a steering and driving capability in one module. Based on the wheel-motor, two configurations for the wheel module assembly were discussed, as represented in Figure 4(a). Each configuration consists of a drive motor-wheel unit and a steering actuator/transmission as constituent parts. Interfaces *Drive* and *Steer* are *provided interfaces* to each configuration, letting the drive motor and steer configuration to interact with the wheel-module configuration. Interface *Sensor* (to measure angle and velocity of wheel) is a *required interface*, letting the wheel configuration send sensor measurements to other blocks. Configuration 1 consists of a gearing with a steer motor to steer the drive motor assembly (Figure 4(b)). Configuration 2 contains freely revolving driving assembly with the drive wheel mounted off the vertical rotational axis of the steering assembly. A brake modulator inside the steer assembly locks the drive assembly at the required angle. Position encoders were selected to measure the steering and driving rotation angles.

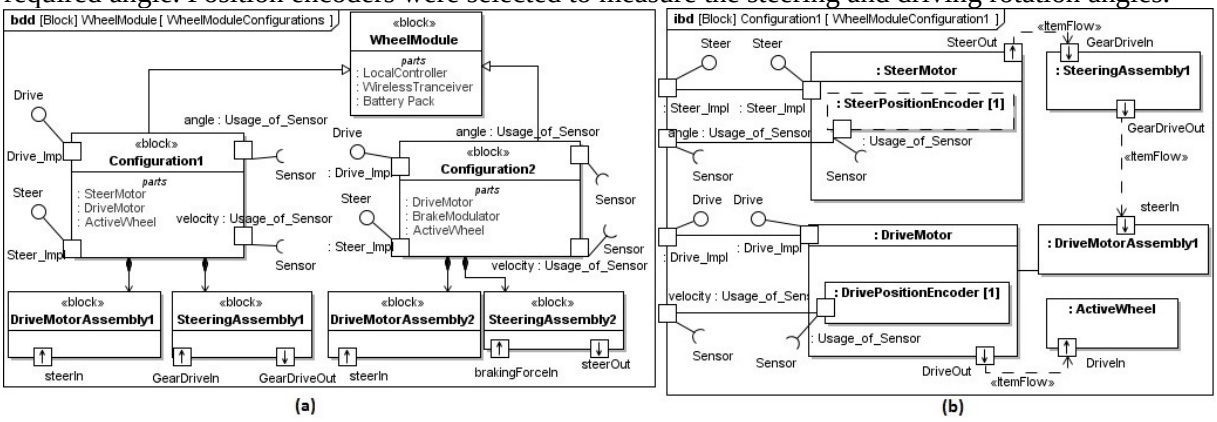


Figure 4. (a) Two alternatives for wheel module configuration. (b) Internal block diagram showing internal connections for configuration 1

In order to make a decision about the wheel module configuration, it was important to realize each configuration in terms of form, and analyze them in terms of behavior. For this purpose, a CAD model and a dynamic analysis model for each configuration were built, as discussed in section 3.3.

For the controller, a centralized architecture and a distributed architecture was considered. Although both the architectures were also dependent upon selection of wireless or wired transmission, it was decided to control the angle and velocity of each wheel through a local controller (Figure 5(b)). If a wireless transmission is considered, a local battery can power the wheel module and the local controller. In this case, a wireless transceiver would send data to the main controller and receive reference commands from it as shown in Figure 5(a). Figure 6 shows the complete distributed control architecture with two wheel modules and the user interface inputs. It can be noted that in case of wired

transmission, the interface *WirelessData* and *WirelessCmd* will be replaced by wire connections. In this case, a central battery can also be considered relevant instead of a localized battery.

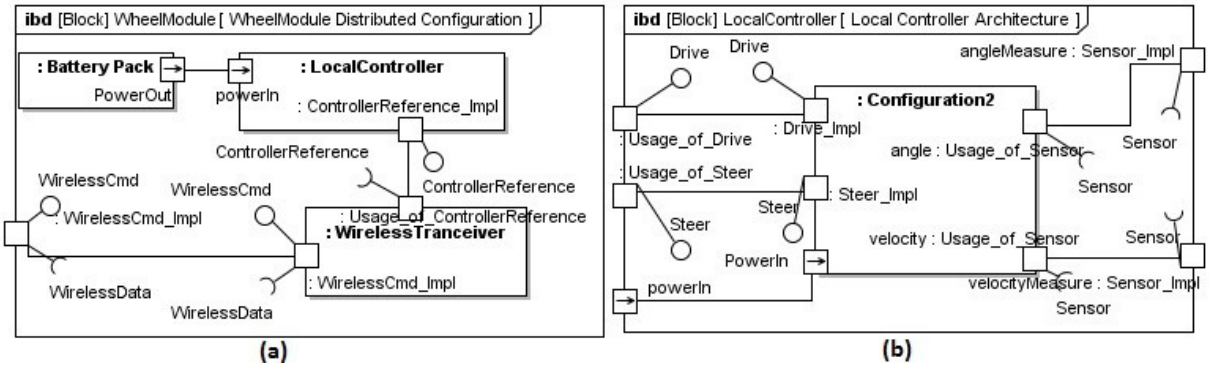


Figure 5. (a) Wheel module in distributed architecture. (b) Internal structure of local controller, and its interface with the wheel module configuration 2

The system-level modeling through SysML proved to capture the information gathered during the synthesis phase effectively, as displayed through different SysML diagrams. Moreover, all SysML diagrams presented here are consistent with each other. This means that introducing a change in one diagram during design leads to relevant changes in other diagrams too. The SysML tool (MagicDraw [20]) supports keeping different SysML diagrams consistent with each other.

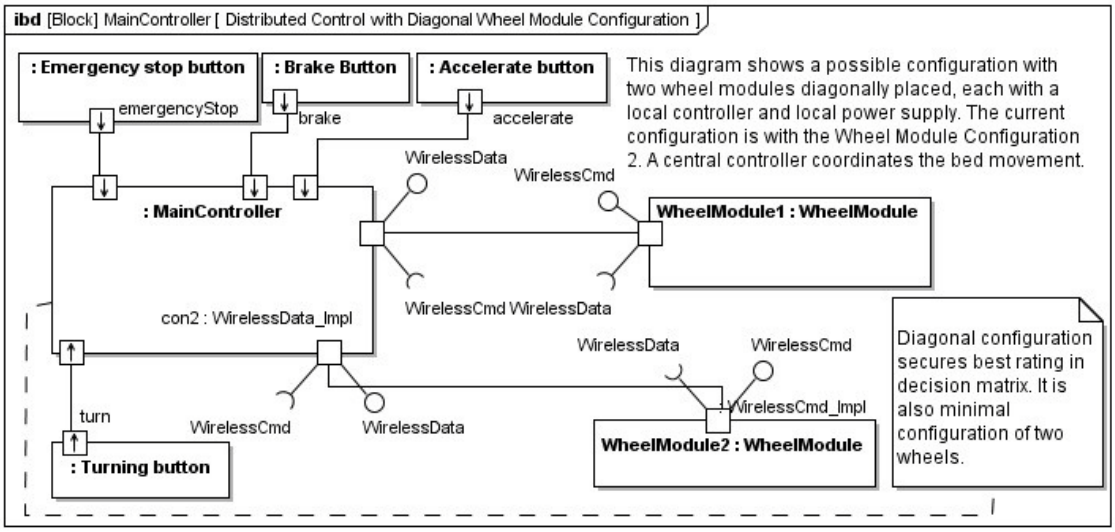


Figure 6. Architecture of a distributed controller with the user interface buttons, and two wheel modules in the diagonal propulsion system configuration

3.3. Domain-specific design

As discussed earlier, the decision about some of the components, configurations, and alternatives needed further analyses through domain specific models. This section provides a limited overview about mechanical design and analysis of the wheel module, in order to make a decision about the wheel module configuration assembly and the relevant components. Other issues such as controller bandwidth, energy supply and communication between wheel modules etc. were also discussed. However they are not presented here due to space concerns.

Wheel module assembly design

The CAD modeling of the two wheel module configurations was performed to get an estimation of the size of the whole module and to know the necessary components. The *wheel-motor* model was utilized to construct the assembly. The two concepts can then be compared based on manufacturing cost, safety, and performance criteria. Figure 7 (a) and (b) show the two configurations.

Analysis of wheel module configurations

In order to analyze how the bed moves with each wheel module configuration, a dynamic analysis model was created. The multi-domain physical modeling and simulation tool MapleSim [12] was used

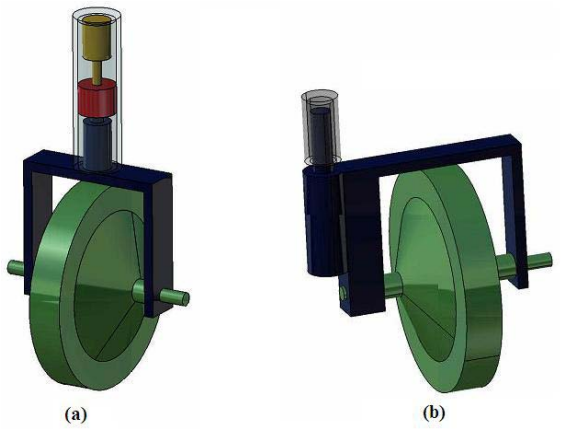


Figure 7. CAD models of wheel module configurations. Both configurations contain the same drive motor and wheel (a) Configuration 1 with steering motor for steering (b) Configuration 2 with brake modulator for steering

to construct an initial simplified model of the two concepts. This analysis also highlights the control complexity of each configuration. Figure 8(a) shows the rigid body model using configuration 1. Figure 8(b) shows the rigid body model using configuration 2 with the brake modulator represented as a clutch that locks or releases the drive wheel. Both configurations gave acceptable performance. However, configuration 2 (i.e. utilizing a brake modulator to provide steering) requires an intelligent control strategy to control steering angle, whereas configuration 1 is a much simpler control problem.

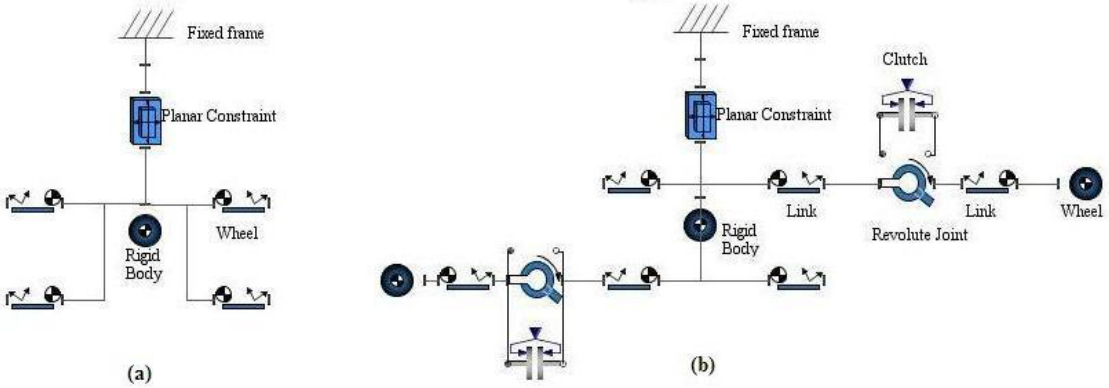


Figure 8. Design models created to analyze the bed movement. (a) Configuration 1 (b) Configuration 2, the clutch mimics the brake modulator. A planar force in each driven wheel mimics the steering and driving actuator in both configurations

The above design and analysis through domain-specific design models was necessary in order to take a decision about wheel-module configuration – a situation which is typical to most mechatronic system developments. This further emphasizes the need to integrate the system-model with these domain specific design models in order to ensure consistency while designing. Section-4 throws further light on this topic.

3.4. Dependencies between domain-models

Through the SysML model, the *design concept* for the propulsion system was specified. Further domain-specific design provided more information in order to make decision about sub-systems, components, and alternatives. The domain-specific models were created based on information obtained from the SysML model. For example, the CAD model in Figure 7(a) was made based on the initial sketches made during conceptual design phase, which led to creation of figure 4(b). The information about wheel, wheel-motor, steering motor and steering configuration was needed before the CAD model could be created; a dependency between the SysML model and the CAD model. Moreover, the dynamic analysis model in Figure 8(a) was created based on the propulsion system configuration, a dependency between CAD model, SysML model and the dynamic analysis model. Since the SysML model contains the wheel module requirements and the complete system structure, it is important to keep this model consistent with other design models. A change in e.g. the CAD model

during design iteration should be traced back to the corresponding model element inside the SysML model, a problem of multi-view consistency as explained in [5]. The following section presents our proposed infrastructure for integrating models.

4 INTEGRATING MODELS FOR ABSTRACTIONS IN DESIGN

Several iterations may take place both between the design phases and between the corresponding design models before the design process is complete. Each model contains a certain level of detail. For example, SysML provides a rather abstract system view, whereas, a CAD model provides a detailed view. Figure 9(a) shows a domain-independent system-model, and the corresponding domain-specific models that are typically created during design.

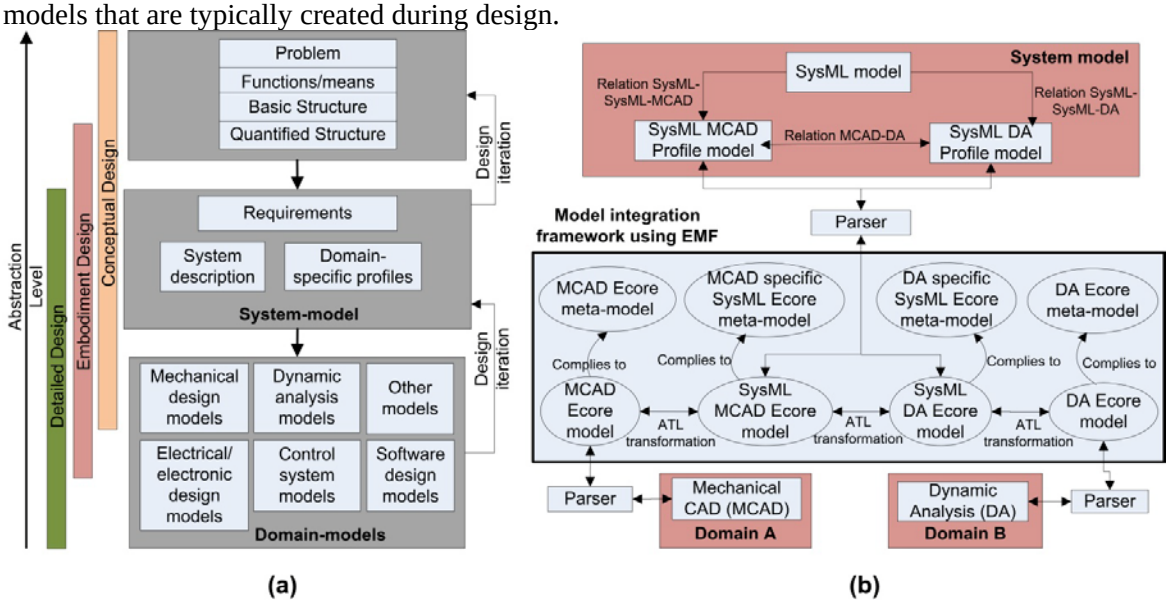


Figure 9. (a) Iterations between multi-domain design models, and abstraction during different design phases (b) Integrated modeling infrastructure based on EMF

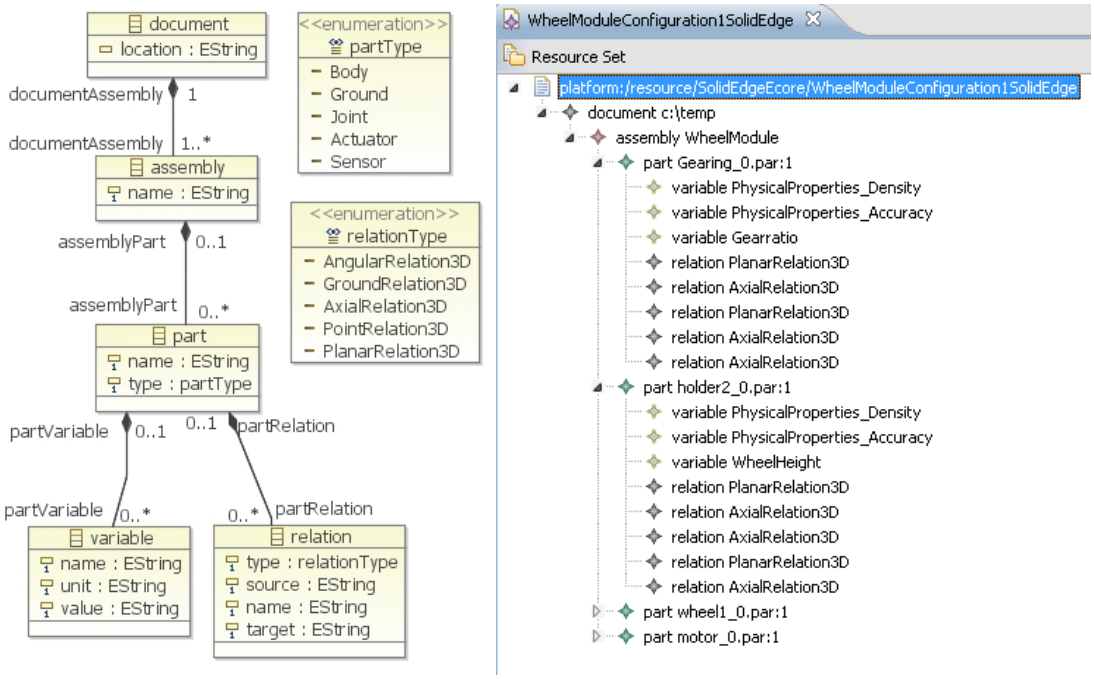
Keeping the domain models consistent with each other requires means to manage inter-domain dependencies. A domain-model contains information that has relation to other domain-models (dependencies). It also contains information that is relevant only within the domain-model itself. We utilize a meta-modeling approach to characterize the information relations between domain-models. A meta-model specifies the abstract syntax of a modeling notation [17]. A meta-model for a domain specifies the concepts that exist within that domain [5]. Therefore, different domain-models comply with different meta-models. By defining relationships between these meta-models, it is possible to write transformations between models, which comply with those meta-models. We utilize the Eclipse Modeling Framework (EMF) [18] to define the meta-model of each domain, and we define relations between meta-models through a rule-based language (ATL) [19]. Figure 9(b) shows the integrated modeling infrastructure, in this case involving two domains: mechanical CAD, and dynamic analysis. Inside EMF, a model is represented as an Ecore model. Therefore, each domain is specified through a domain-meta model in Ecore, e.g. MCAD Ecore meta-model (Figure 9(b)).

Since SysML is a general purpose modeling language, the aim of using SysML is to create a system model independent of any domain or technology. However, our proposition to establish relationships between domain-models through SysML requires SysML to support domain-specific concepts. This has been supported in SysML through creation of profiles. A profile extends the SysML meta-model with the needed domain-specific constructs. Figure 9(b) shows two profile blocks, each relating to a domain-specific model. For example, the SysML MCAD profile model contains a model built with MCAD concepts such as: assembly, part, relations etc. An MCAD-specific SysML meta-model can then be created as an Ecore model. For dynamic analysis (DA) models, a DA-specific SysML meta-model can be created. Relationships between the SysML model and the domain-specific part of the SysML model are necessary to establish a link between the system design and the domain-specific design. This can be achieved by allocating (manually) elements/components in SysML model to their counter parts in domain-specific part of the SysML model. In this way, the relationships between different design models can be established (inter-domain dependencies) through the SysML model, by

establishing allocation relationships between SysML model and the corresponding domain-specific parts of it. For example, in Figure 9(b), it is possible to establish relationships between the MCAD model and the dynamic analysis model (DA) by establishing relationships between the *SysML model* and the *SysML MCAD Profile Model*, and between *SysML model* and the *SysML DA Profile Model*. These relations can be manually drawn, or can be automated. An automation procedure requires relating a concept in one meta-model to a corresponding concept in the other meta-model and writing transformation rules based on those relations. Figure 9(b) shows an ATL transformation between each domain model and the domain-specific part of SysML model in EMF.

4.1. Integration Example

This section will illustrate an integration example between the MCAD model of the hospital bed wheel unit and the SysML system model. Figure 10(a) shows a generalized meta-model for MCAD, based on constructs such as *assembly*, *part*, *variable*, *relation* etc. There are differences between MCAD tools, which can lead to a different meta-model for each tool. However, we propose a generalized meta-model to be adapted for a domain such as MCAD, and aligning different modeling languages within the domain to that meta-model. An important consideration here is that the proposed MCAD meta-model does not contain all MCAD concepts; rather it only contains constructs relating to the type of information that we are interested to obtain from MCAD tools.



(a) (b)

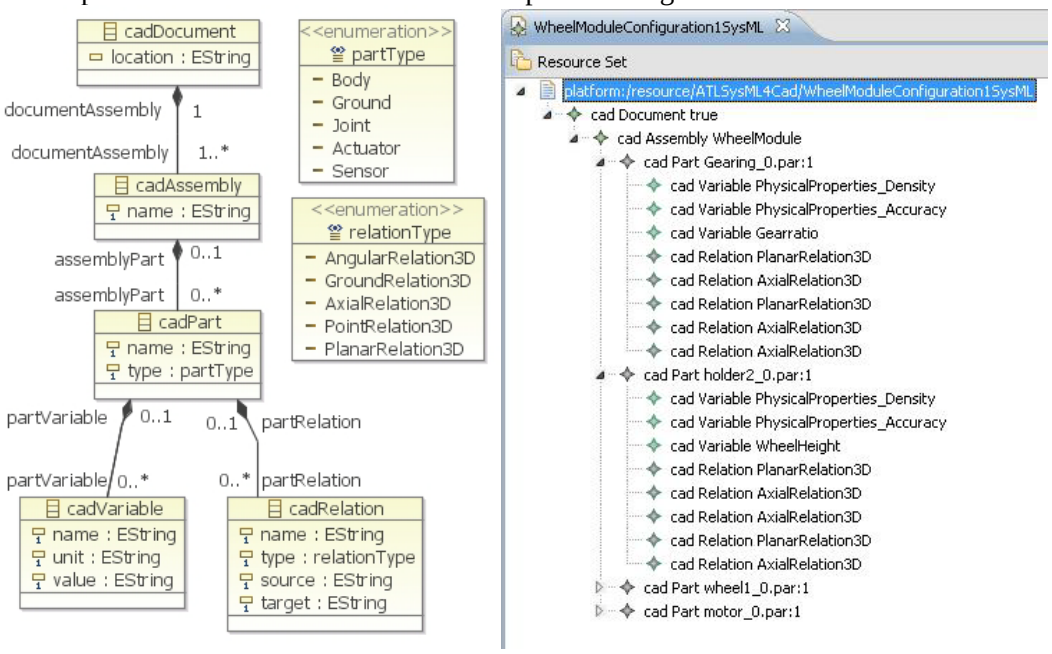
Figure 10. (a) MCAD meta-model in EMF Ecore (b) Configuration 1 represented in Ecore

A model complying with the MCAD meta-model can be created by making API calls to the MCAD tool of choice. We utilize Solid Edge [21] in this example, and populate a Solid Edge Ecore model (representing the MCAD Ecore model) through a developed parser (see Figure 9(b)). The same parser allows us to create a model inside Solid Edge based on an Ecore model. Hence, it is now possible to represent the wheel-module assembly configuration 1 as an Ecore model shown in Figure 10(b).

The SysML profile for MCAD is based on MCAD concepts, which are extended from the SysML meta-model elements. For example, a *Part* extends a SysML *block* etc. Figure 11(a) shows SysML4CAD profile meta-model represented in Ecore. Another parser is used that populates an Ecore model complying with SysML4CAD domain-specific meta-model. The same parser allows creating a *SysML MCAD Profile* model from an Ecore model (see Figure 9(b)). Figure 11(b) shows the wheel module configuration 1 as SysML4MCAD Ecore model.

Declarative ATL rules can be specified for each meta-model construct, to create a target model element based on the source model element. This completes a transformation between MCAD and SysML4CAD. In a similar fashion, transformations will be written between the DA model and SysML4DA model based on their corresponding meta-models as shown in Figure 9(b). At the end, the

relationships between SysML4CAD profile model and SysML4DA profile model can be specified, to describe the dependencies between the two domain specific design models.



(a) (b)

Figure 11. (a) SysML profile for CAD (SysML4CAD) in Ecore (b) Wheel module configuration 1 represented as an instance of SysML4CAD profile

4.2. Mechatronic procedure, domain models under SysML Umbrella

The extended system model allows us to first establish an overall overview of the mechatronic system, to specify the function principles, and to specify the principle solution through a common specification language. Through the model integration framework, domain-specific views are integrated, and dependencies and consistency between design models are maintained via the system model. We believe that establishing abstract system information through the system model and keeping other design models consistent with the system view provide good support for identifying inconsistencies among design models, and avoiding integration failures as a result. The integration infrastructure can in this way support development of better mechatronic design solutions.

5 CONCLUSION

In this paper, a proposal for mechatronic design infrastructure based on the integration of design models is presented. Mechatronic product development requires a common specification language for different domain experts to communicate with each other. It also requires design models that can support multi-domain constructs inside one modeling language to be able to model a mechatronic concept. Though some modeling languages support multi-domain modeling and analysis (such as Modelica), other design models are very domain-specific. Using SysML to establish a domain-independent system model, and establishing relationships and means for automated integration with other design models is the main theme of this paper. The paper presents a step by step construction of a SysML model and some domain-specific models for the design problem of a hospital bed's propulsion system. System level modeling can play a major role in mechatronic product development, thus it has to be supported through all development phases. The proposed integration infrastructure enables us to maintain both the system model and domain-specific design models throughout the product development process. A small integration example between an MCAD model and a SysML model is presented to exemplify our proposal. Future work targets extending the model-integration example towards a more comprehensive integration example between: SysML, MCAD, and dynamic analysis (DA), evaluating the support potential of this approach during mechatronic design phases.

ACKNOWLEDGEMENT

The authors are thankful to Carl-Johan Sjöstedt, Daniel Frede, Daniel Malmquist, Hamid Shahid, and Mohammad Khodabakhshian for their valuable inputs in performing the hospital-bed design-study.

REFERENCES

[1] Tomiyama T., D'Amelio V., Urbanic J. and ElMaraghy W., Complexity of Multi-Disciplinary Design. *Annals of the CIRP*, 2007, 56(1), pp 185-188

[2] Buur J. and Andreasen M. M., Design Models in Mechatronic Product Development. *Design Studies*, 1989: 10(3), pp 155-162

[3] Gausemeier J., Schäfer W., Greenyer J., Kahl S., Pook S. Management of Cross-Doamin Model Consistency During the Development of Advanced Mechatronic Systems. *Proc. Internationl Conference on Engineering Design, ICED'09*, Stanford, California, USA, 2009

[4] Johnson T. A., Paredis C. J. J. and Burkhart R. Integrating Models and Simulations of Continuous Dynamics into SysML. *Proc. 2008 Modelica Conference*, Germany, March 2008.

[5] Shah A. A., Kerzhner A. A., Shaefer D. and Paredis C. J. J. Multi-View Modeling to Support Embedded Systems Engineering in SysML. *Lecture Notes in Computer Science, Graph Transformations and Model-Driven Engineering*, 2010, Volume 5765/2010, pp 580-601

[6] Hehenbrger P., Egyed A. and Zeman K. Consistency Checking of Mechatronic Design Models. *Proc. ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2010*, Montreal, Canada, 2010

[7] Estefan A. J. *Survey of Model-Based Systems Engineering Methodologies*, Technical Report, Revision B, Incose Focus Group, 2008

[8] Pahl G., Beitz W., Feldhusen J. And Grote K., *Engineering Design- A Systematic Approach*, 2007, (Springer-Verlag), ISBN 3-540-19917-9

[9] Frey E., Ostrosi E., Roucoules L. and Gomes S. Multi-Domain Product Modelling: From Requirements to CAD and Simulation Tools. *Proc. Internationl Conference on Engineering Design, ICED'09*, Stanford, California, USA, 2009

[10] Association of German Engineers, VDI-guideline 2206, Design Methodology for Mechatronic Systems, Berlin, 2004

[11] Modelica Association, *Modelica Language Specification V3.2*, 2010, https://www.modelica.org/documents/ModelicaSpec32.pdf

[12] Maplesoft, *MapleSim V4.5*, http://www.maplesoft.com/products/maplesim/

[13] Object Management Group, *OMG System Modeling Language Specification V1.2*, 2010, http://www.omg.org/spec/SysML/1.2/PDF/

[14] Friedenthal S., Moore A. and Steiner R. *A Practical Guide to SysML- The Systems Modeling Language*, 2008, (MK/OMG Press), ISBN 978-0-12-374379-4

[15] Wölkl S. and Shea K. A Computational Product Model for Conceptual Design Using SysML. *Proc. ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2009*, California, USA, 2009

[16] Follmer M., Hehenberger P., Punz S. and Zeman K. Using SysML in the Product Development Process of Mechatronic Systems. *Proc. International Design Conference, Design2010*, Dubrovnik, Croatia, 2010

[17] Czarnecki K. and Helsen S. Feature-Based Survey of Model Transformation Approaches. *IBM Systems Journal*, 2006, Vol. 45, No. 3, pp. 621-645

[18] Eclipse Foundation, *Eclipse Modeling Framework (EMF)*, 2009. http://www.eclipse.org/modeling/emf/

[19] Eclipse Foundation, *Atlas Transformation language (ATL)*, 2009, http://www.eclipse.org/m2m/atl/

[20] NoMagic, *MagicDraw UML/SysML V16.9*, http://www.magicdraw.com/

[21] Siemens PLM Software, SolidEdge, 2010, http:// www.plm.automation.siemens.com/en_us/products/velocity/solidedge/index.shtml

Contact: Ahsan Qamar
PhD. Candidate
KTH- Royal Institute of Technology,
Division of Mechatronics, Department of Machine Design,
Brinellvägen 83, 10044, Stockholm, Sweden
Email: ahsanq@kth.se, Web: www.md.kth.se/~ahsanq