Understanding the Relationship of Information in Mechatronic Design Modeling

Peter Hehenberger[1], Alexander Egyed[2] and Klaus Zeman[1]

[1] Institute for Computer-Aided Methods in Mechanical Engineering, Johannes Kepler University Linz, Altenbergerstr. 69, A-4040 Linz, Austria/Europe
{peter.hehenberger, klaus.zeman}@jku.at
[2] Institute for Systems Engineering and Automation, Johannes Kepler University Linz, Altenbergerstr. 69, A-4040 Linz, Austria/Europe
alexander.egyed@jku.at

Abstract. Understand the information flow during engineering processes of mechatronic systems is an important point for competitive mechatronic engineering. The paper gives an overview about product models used in mechatronic design and analyzes also the flow of information through tools. Furthermore, there is also the need for considering model consistency because if objects and models are independently created and maintained by the various disciplines then correctness is no longer guaranteed. The same is true for objects or models that are transferred from one discipline to another, from one abstraction level to another, or from one design phase to the next one – if such objects or models are subsequently modified on both ends just as proposed in simultaneous engineering.

Keywords: Mechatronic Design, Product Model, Change Propagation, Information Flow

1 Introduction

Mechatronics can be considered to be an integrative methodology utilizing the technologies of mechanical engineering, electrical engineering/electronics and information technology in order to provide enhanced products, processes and systems [1, 2]. Mechatronic design of machines, devices and plants can help create improved and enhanced products, processes and systems. On the other hand, the complexity of these systems usually higher due to the intended beneficial interaction between components and technologies from different mechatronic disciplines. In all phases of the design process there is a need to build models which are simplified representations of corresponding originals, in many cases of the real world. In different phases, these design models have different aims. In the conceptual design phase, physical principles, functions, structures, etc have to be evaluated by building models. In most cases, analytical and virtual models are less expensive and less time-consuming than physical prototypes.

Engineering processes and tools accentuate the capture of engineering solutions and possibly intermediate solutions generated along the way. What is typically not captured is the flow of information that led to these (intermediate) solutions. Indeed, the engineering process is based on a complex flow of information where results computed in one step (perhaps by one discipline) are consumed in other steps (perhaps by other disciplines). The aim of this work is in better understanding this flow of information.

Trivially, we can think of the flow of information as data exchanges and the computations as processing that produces this data and consumes it. The main goals of this consideration are:

- Understand the impact of changes during engineering processes (direct, indirect and transitive impact)
- Understand the flow of information through tools: For the most part, engineering tools are well established and internally reasonably well integrated. To understand the impact of changes, however, we require its knowledge together with external flow.
- Understand the direct and indirect flow across tools
- Understanding trade offs (completeness and correctness of flow vs. completeness and correctness of change impact)

2 Background

Models are very important for the design of complex engineering activities. For example, numerical modeling and simulation are fundamental to the engineering of high performance characteristics. Such modeling approaches require experimenting with computer-based models. From the viewpoint of engineering design, models are means of storing knowledge which is used by simulations for producing information that may improve product knowledge and potentially also the quality of many analyses and decisions made during the design process [3].

When designing a mechatronic system, it is possible to design the mechanical equipment, before any of the control system design has been initiated. An obvious drawback of this sequential approach is the (probable) lack of compatibility between the subsystems which results in additional efforts and costs to (optimally) meet the specifications of the total (integrated) system. Another drawback of this approach is that during the design process decisions have to be made about whether to use a mechatronical or just a mechanical solution. Design engineers have to balance mechanical, electr(on)ic and software solutions.

In general, a model is devoted to the task of mapping reality onto a significant representation of reality in order to make valid predictions and conclusions for reality. It should include the relevant phenomena/effects of interest (“views of the object”, such as geometry, dynamics, stability, materials, electrodynamics, controllability, cycle time, maintenance, etc. During all phases of the design process there is a need to establish models. If these models are simplified representations of the product to be built then we speak of product models. The purpose of these models changes depending on the phases of the product development. During the conceptual design

phase, physical principles, functions, structures, etc have to be modeled and evaluated. In most cases, analytical (mathematical) and virtual models are less expensive and less time-consuming than physical prototypes. Virtual models can be implemented and used to simulate and evaluate (significant representations of) reality with the help of computers.

Some aspects of change management in the design of mechatronical products are discussed in [4,5]. The parts of a product (e.g. engine, gearing, clutch) are classified as absorbers, carriers and multipliers depending on whether a change in the input parameters of a part causes less, equal or more changes than in the output parameters. Multipliers can cause avalanches of changes of the design of a product. As solution to avoid this, "safety" margins are proposed, so the change of one part does not necessarily affect the adjoining parts.

In [6], a change propagation index (CPI) is introduced based on studies of change propagation in the design of automotive platforms. With the CPI, the identification of parts or elements as absorbers, carriers or multipliers is done more easily. The largest set of change requests during the development of a sensor system was analyzed in [7] to get a better understanding of change and change development of large scale products. Based on the documentation of change requests, a design structure matrix (DSM) and a ΔDSM was developed representing the actual state of a change.

3 Information in mechatronic design processes

3.1 Model views

In the design process of mechatronic systems product models and data from different angles have to be analyzed – representing the different disciplines involved. This diverse viewpoint dimensions include the following aspects (Fig. 1):

- objects and models from different disciplines, which are involved in the development process of a mechatronical system
- model granularity (which differs across disciplines), denoting the extent to which an object or model is broken down into smaller parts (e.g., sub-models with a higher level of detail and a lower level of abstraction, respectively)
- design phases: The development/design process can be structured into four phases, namely problem definition, conceptual design, preliminary design and detailed design

Furthermore, there is also the need for consistency because if objects and models are independently created and maintained by the various disciplines then correctness is no longer guaranteed. The same is true for objects or models that are transferred from one discipline to another, from one abstraction level to another and from one design phase to the next one – if such objects or models are subsequently modified on both ends just as proposed in simultaneous engineering. This problem, of course, becomes increasingly pronounced with system size and complexity because likely more people and disciplines are involved. A proven strategy to handle this increasing

complexity is to divide up the work as it is done in concurrent engineering. However, it is rarely possible to divide up a problem such that two people can work fully independently from one another and then, at the end, expect that the results of their work fit together without conflicts. Through good modularization, it is possible to achieve some degree of independence; however, it is never possible to avoid dependencies. The role of consistency is to characterize such dependencies – to identify the conditions that have to be met for objects and models to fit together.

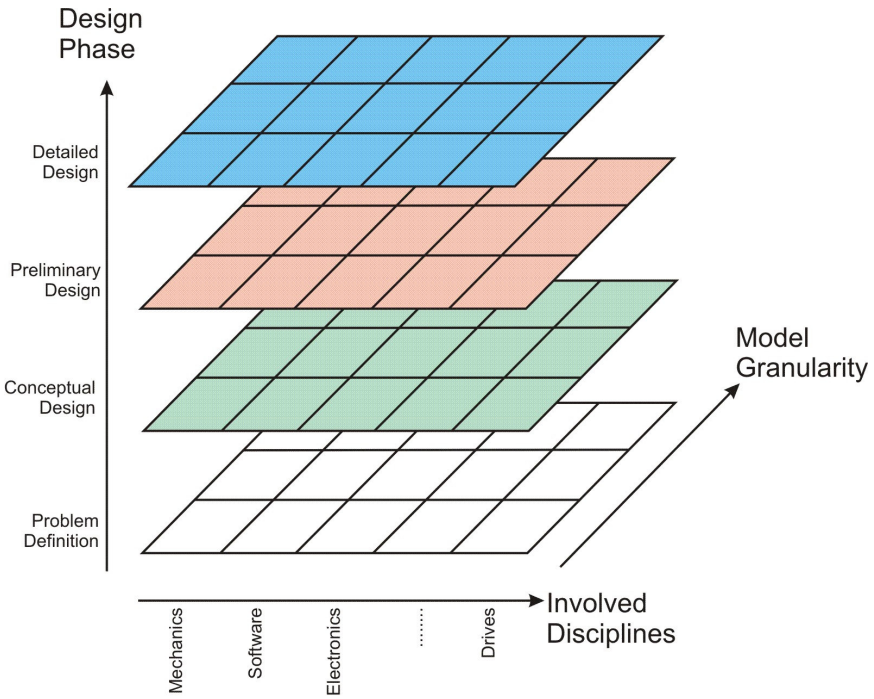

Fig. 1. Viewpoint of product models and data [8]

The aim of a behavioral model, i.e. a model of the (physical) system behavior, is to serve as a means of finding an answer to a design question. Each model is unique and it has a specific purpose. The preliminary design phase is often characterized by a cascading series of what-if questions. Many of these questions, which may be of divergent character, are related to the complex dependencies between geometry (shape), topological structure and (physical) behavior. The complex nature of engineering design, as well as the time and cost-constraints on this process, require highly efficient and flexible procedures to configure system models (overall models) for non-routine simulations. The modeling challenge may be addressed by a modular or an integrative model design. A modular sub-system has interfaces that are well defined and are shared with only a few other sub-systems. An integrative system has interfaces that may be more complex and shared across the whole system model. The

physical behavior of a technical system depends on the properties of the sub-systems and their (internal and external) interactions [9]. An interface describes the relationship between a pair of mating features. Product models are important containers of significant product properties such as shape and material.

3.2 Consistency

Consistency rules are simply conditions on a model which evaluate to true or false – depending on whether the rule is satisfied by the model or not [8,10]. There are typically many such consistency rules and these rules need to be re-evaluated whenever the model changes. This re-evaluation is computationally expensive if done exhaustively with every model change. During a rule's evaluation, a consistency rule investigates typically a portion of the model only. We define the accessed portion of a model as the scope of a rule. For example, we want to analyze different design models during the development process of a gripper. Figure 2 shows different simplified models from different design stages.

The function structure is used to describe product functionality because in engineering design, the final goal is the creation of an artifact, product, system, or process that performs a function or functions under certain constraints to fulfill customer need(s). The conceptual design model is used to select, define and fix the main parameters of the gripper. In detailed design we have a CAD-model which describes the geometry (and other properties) of the gripper.

Consistency rules are written for a context element (a type of model element, e.g. a function) from where its evaluation starts. For consistency rule 1 (Fig. 2), the interaction between the function structure and the conceptual design model is described as "applies to <fix workpiece> requires(this).contains(<pneumatic cylinder>)". This means that for the function "fix workpiece" a drive like a "pneumatic cylinder" is required. In the second rule the interaction between the model parameter "distance piston" and the geometric representation in the CAD-model is defined as equal.

Large design models contain thousands of model elements. Designers easily get overwhelmed maintaining the correctness of such design models over time. Not only is it hard to detect new errors when the model changes but it is also hard to keep track of known errors. In the software engineering community, this problem is known as the consistency problem and errors in models are known as inconsistencies. Inconsistencies are detected through reasoning processes that require the existence of rules that describe correctness. Such consistency rules describe conditions that a model must satisfy to be considered a valid model (e.g., syntactic well-formedness and even coherence between different models). Such consistency rules obviously vary for different models. The validity (correctness) and invalidity depend on the modeling notation (language) and its semantics. However, for many mainstream modeling languages (e.g., SysML, UML) such semantics are well-defined and engineers have been able to identify consistency rules.

During the development stages of complex mechatronic systems many models are generated either manually or in an automatic manner. The nature of these models

ranges from requirements expressed in text documents down to mathematical description of a physical effect.

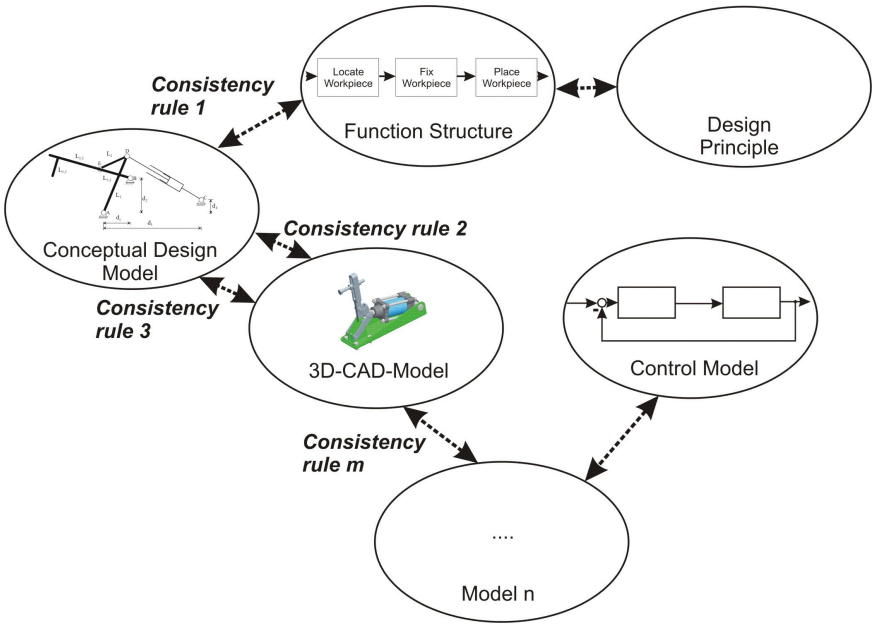

Fig. 2. Consistency rules between different models [8]

3.3 Information Flow

Understanding the relationships among different models is a nontrivial task and becomes harder the more complex a system is. Therefore one important topic is the analysis of these relationships. Traceability refers to the completeness of the information flow every step in a process chain.

The lack of information flow is most problematic in context of change management. If some (intermediate) solution changes - i.e., changed data - then engineers must be able to understand where this data was consumed because other solutions that were computed based in part on this changed data may change also. Fig. 3 illustrates this in a simple manner. We see that engineer 1 produces output $Data_B$ through some processing that requires $Data_A$ as input. $Data_B$ is then consumed by Engineer 2 who, through some other processing, produces $Data_C$. If $Data_A$ changes then $Data_B$ directly and $Data_C$ indirectly may be affected by this change where $Data_C$ should only be affected if $Data_B$ is affected. Change propagation thus requires knowledge about the flow of information where, in this particular case, $Data_B$ is a $Processing_{A\text{-}B}$ function of $Data_A$ ($Data_B = Processing_{A\text{-}B}(Data_A)$).

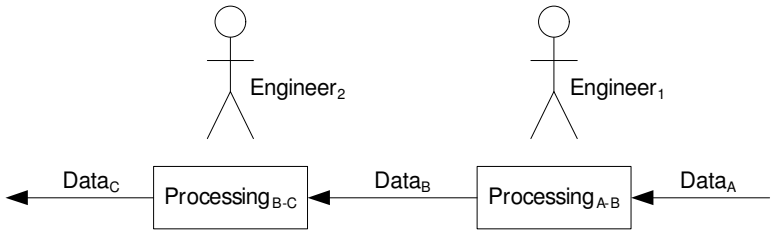

Fig. 3. The impact of successive processing onto the information flow.

The data may also be abandoned if undesirable. The processing of data and hence the flow of information can be arbitrarily complex and also hierarchical. The flow of information in engineering processes is typically meant to accentuate information flow among tools and users (e.g., communication between two users, between two tools). We distinguishe the information flow between engineering tools (external flow) and information flow within engineering tools (internal flow). The main focus of this work is on external information flow. This focus is useful and necessary because engineering tools internally

1) may already have (limited) ability to track information flow and
2) even if they do not it is the tool builder's responsibility to providing this capability.

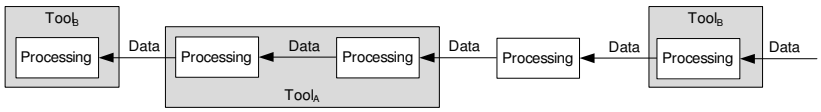

Fig. 4. Relationship between tools, processing, and data**.**

Fig. 4 depicts the relationship between tools, processing, and data. Engineering tools mainly serve as means for processing data and internal data exchanges. Results produced by these tools are usually not forwarded to other engineering tools but rather stored persistently. Depending on the complexity of the tool, it may support a range of processing activities. In some cases, a tool may support multiple related processing activities where the flow of information among the activities is internal to the tool (as in $Tool_A$). In other cases, a tool may support one or more independent processing activities where the flow of information among these and other activities is external (outside the tool as in $Tool_B$). Not all processing may be tool supported and even for the ones that are, engineering tool support typically involve extensive manual activities. Individual tools may support multiple processing steps. These tools are typically well integrated and within a single tool there may even be support for the capture of the information flow.

4 Conclusion and future work

To understand the flow of information this approach characterize direct and indirect, internal, external, and transitive information flow - within and across user domains (network, verbal). It is important to note that this approach is not about tool integration or model integration. Both areas are well studied. Moreover, they require tool vendor support to be successful. The future work will treat tools as black boxes mostly and only characterizing certain properties about these tools which are important for understanding information flow and change propagation.

Acknowledgments

We gratefully acknowledge that this work was supported in part by the Austrian Center of Competence in Mechatronics (ACCM), a K2-Center of the COMET/K2 program, which is aided by funds of the Austrian Republic and the Provincial Government of Upper Austria and in part by the Austrian FWF grant P21321-N15.

References

1. De Silva, CW.: Mechatronics – an integrated approach. CRC Press Boca Raton, London, New York, Washington DC (2005)
2. Isermann, R.: Mechatronic systems. Fundamentals. Springer, London (2005)
3. Hehenberger, P., Zeman, K.: The role of hierarchical design models in the mechatronic product development process. In: Proceedings of TMCE International Symposium Series on Tools and Methods of Competitive Engineering, Izmir, Turkey (2008)
4. Eckert, C.M., Zanker, W., Clarkson, P.J.: Aspects of a Better Understanding of Changes. In: Proceedings of the 13th Int. Conference on Engineering Design: Design Applications in Industry and Education, Glasgow, UK, pp. 147-154, (2001)
5. Jarratt, T.A.W., Eckert, C., Caldwell, N.H.M., Clarkson, P.J. Engineering change: an overview and perspective on the literature, Res Eng Design, DOI 10.1007/s00163-010-0097-y, (2010)
6. Suh, E.S.: Flexible Product Platforms. Doctoral Dissertation, Massachusetts Institute of Technology, Engineering Systems Division, Cambridge, MA, USA, (2005)
7. de Weck, O., Bounova, G., Keller, R., Eckert, C., Clarkson, P.J.: Change Propagation Analysis in Complex Technical Systems. ASME Journal of Mechanical Design, Vol. 131, August (2009)
8. Hehenberger, P., Egyed, A., Zeman, K.: Consistency checking of mechatronic design models, In: Proceedings of IDETC/CIE 2010, ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 15.-18.August, Montreal, Quebec, Canada, Paper-Nr. DETC2010-28615, (2010)
9. Hehenberger, P., Poltschak, F., Zeman, K., Amrhein, W.: Hierarchical design models in the mechatronic product development process of synchronous machines, Mechatronics ISSN 0957-4158, Volume 20, pp. 864-875, http://dx.doi.org/10.1016/j.mechatronics.2010.04.00, (2010)
10. Egyed, A.: Automatically Detecting and Tracking Inconsistencies in Software Design Models, IEEE Transactions on Software Engineering (TSE), to appear (2010)