# From Conception to Refinement in Mechatronics Systems Engineering

Georg Hackenberg, Christoph Richter, and Michael F. Zaeh

***Abstract*—The complexity of mechatronic systems increases constantly due to market requirements. Traditional engineering approaches have troubles coping with the desired functionality. A major problem is the early and continuous integration and coordination of engineering disciplines (i.e. mechanic, electric, and software). To address this situation and to enable a concurrent engineering of participating disciplines, a model-based approach to early conception and subsequent refinement of mechatronic systems is proposed in this paper. This approach allows specifying a high-level mechatronic concept within a single editor and the refinement of this concept by integrating it with established engineering tools. In particular, the paper outlines the technical and methodical integration of discipline-specific tools such that refinement steps can be tested automatically and continuously. The suitability of the approach is shown along selected examples. Finally, the paper finishes with a reflection on the current state and an outlook on future research activities.**

***Index Terms*—Model-based development, concurrent engineering, mechatronic systems engineering.**

## I. Introduction

The machine and plant engineering is currently facing a large variety of challenges including shorter innovation and production cycles [1]. Besides technological improvements, the optimization of engineering processes is one key factor to remain successful on the market [2]. Current engineering processes are still characterized by serial process steps [3], discipline-specific procedures [4] and a lack of synchronization between the involved disciplines [5]. In addition, the diversity of engineering tools that cope with discipline-specific engineering problems impedes the establishment of an integrated and interdisciplinary development approach [6].

A recent approach for coping with these challenges is the model-based development of mechatronic systems [7]. This development approach focuses the alignment of the entire process along one central and interdisciplinary system model. This model is constructed by engineers from all involved disciplines in an early development phase. Thereby a common understanding of the system to be developed can be derived [7], mistakes can be avoided and system requirements can be considered appropriately [8]. However, these advantages are countered by the effort needed to build an additional model. For this reason, model-based development is currently applied rarely within machine and plant engineering [3]. Therefore, it is a reasonable attempt to establish the system model during the entire development process in order to justify the early modeling effort by savings in later phases. This leads to the need for integrating seamlessly the system model and established engineering tools so that engineers from all disciplines can use the early interdisciplinary development insights without additional efforts [9].

Manuscript received May 9, 2015; revised July 18, 2015. This work was supported in part by the German Federal Ministry of Economics and Technology (BMWi), Funding code 435 ZN.

Georg Hackenberg is with the Chair IV: Software & Systems Engineering, Technische Universität München, 85748 Garching b. München, Germany (e-mail: hackenbe@ in.tum.de).

Christoph Richter is with the Fraunhofer Institute for Machine Tools and Forming Technology, 86153 Augsburg, Germany (e-mail: christoph.richter@iwu.fraunhofer.de).

Michael F. Zaeh is with the Institute for Maschine Tools and Industrial Management, Technische Universität München, 85748 Garching b. München, Germany (e-mail: michael.zaeh@iwb.tum.de).

## II. State of the Art in Mechatronic Systems Engineering

In the following subsections recent approaches in the fields of model-based development, data exchange and simulation within the context of mechatronic engineering are presented. In conclusion, remaining challenges are emphasized in order to derive potentials for advances.

### *A. Model-Based Development in Mechatronic Systems Engineering*

While model-based development is already well-known within the field of software engineering [10], the topic is still at an initial stage within mechatronic systems engineering [3]. But due to the rising complexity of mechatronic systems, such development methods will increase in their importance over the next years [11]. The most important approaches are outlined in the following.

Model-based development usually sets up on various modeling languages, which allow a consistent and abstract description of mechatronic systems. Examples include the Systems Modeling Language (SysML) [12] or function-oriented methods like the approach from Gehrke [13]. Based on these system modeling techniques, different metho- dologies for an integrated model-based development of mechatronic systems were elaborated. Eigner [6], for example, introduces three modeling layers in the left branch of the well-known V-Model from [14]: Qualitative models for modeling requirements or functions, quantitative models for an interdisciplinary simulation and discipline-specific models with discipline-specific engineering and simulation environments. A similar approach was focused within the research project "AQUIMO" [15]. Here, an engineering tool and method were developed, which define the formal modeling of interdisciplinary development information and the belonging process. Another comparable approach was elaborated in the research project "AutoVIBN" [16]: Zaeh *et al.* [17] and

DOI: 10.7763/IJMMM.2016.V4.227

Hummel [18] developed a component-based modeling technique for an easy virtual commissioning. On this basis, Hensel [19] introduced the so-called "Vi-Model" which defines a systematic procedure for a model-based development of automation solutions. A different approach for an easy modeling of mechatronic systems was illustrated by Gausemeier *et al*. [20]. Their specification technique "CONSENS" allows an interdisciplinary description of mechatronic systems using eight modeling views. In addition, several methodologies were developed within the field of model-based systems engineering. Estefan [21] summarizes important approaches including the INCOSE object-oriented systems engineering method [12].

### B. Data Exchange in Mechatronic Systems Engineering

In order to integrate model-based approaches into current development processes, a collaboration between the central system model and discipline-specific engineering tools has to be enabled and established [9]. In particular, model transformations are needed, which transfer several contents of the interdisciplinary system model to tools from the participating engineering disciplines (i.e. mechanics, electronics and software). For *mechanics* this essentially means to transfer geometry and assembly information to an MCAD tool, for *electronics* it means to transfer actor/sensor lists and further electronic equipment to an ECAD tool [16]. Finally, for the *software* the aim of model-transformations is to generate prototypic code from the modeled system behavior [22]. This code can either be executed on a controller or can be refined within a development environment if necessary.

To realize these transformations established data exchange formats can be applied. In the recent past several comprehensive data exchange formats were developed, which deal with engineering information from multiple disciplines. Therein, the Standard for the Exchange of Product Model Data (STEP) [23] is considered to be the most important one [24]. STEP contains physical and functional aspects, which can be used to map product information of the entire life cycle. A similar approach was introduced by Drath [5]. He developed the Automation Markup Language (AutomationML) as an open and standardized exchange format for a seamless automation engineering-workflow with various and substitutable engineering tools. Besides these approaches, data exchange formats exist, which cover only a specific part of a mechatronic system. For example, there are data formats like COLLADA [25] for geometries, like VHDL [26] for electronic components or like PLCopen-XML [27] for software aspects. Finally, there are data formats that can be used for various contexts like the well-known eXtensible Markup Language (XML) [28]. XML is used for platform-independent data exchange and defines a syntax, which can be used to represent any hierarchical structured data in a textual form.

### C. Simulation in Mechatronic Systems Engineering

For securing particular development decisions, simulation is one common approach in mechatronic systems engineering [29]. Along the development process, different simulation models are used [24], which can be classified according to their model granularity and scope [30]. The spectrum includes overall plant simulations [31] as well as simulations for detailed discipline-specific development tasks like the Finite-Element-Method (FEM) for mechanical calculations [32] or multibody systems (MBS) for motion analyzes [33]. To simplify the model building for the latter simulation method, recent approaches address the integration of physics engines [34]. These engines provide effective calculation methods (e.g. for collision detections) and thus decrease the modeling effort as physical behavior does not have to be defined explicitly [30]. Furthermore, comprehensive simulation environments, where simulation models can consist of components from many engineering domains, have been developed and established. An example is the commercial tool Dymola [35] based on the Modelica modeling language [36], which consists of various components represented by differential equations, ports and connectors.

But even comprehensive simulation environments do not claim to capture all relevant simulation aspects of mechatronic systems at once. As a result distributed and detached simulation methods are currently applied throughout the development process [30]. This in turn often leads to problems while integrating discipline-specific solutions [37]. To overcome this situation, recent approaches address a coupling of different simulation environments in order to secure development decisions across disciplines and simulation tools. Examples, which allow a coupling of specific simulation tools, include the approach from Groothuis *et al*. [37] or from Brezina *et al.* [38]. Additionally, approaches like the Functional Mock-Up Interface [39] are available, which offer standards for the co-simulation of a large variety of simulation tools.

### D. Remaining Challenges

The preceding sections show that various efforts have been made to improve development processes for mechatronic systems. Model-based development has great potential to encounter the difficult collaboration and synchronization of engineering disciplines in particular in early stages of engineering [7]. On the other hand, technical possibilities like standard data exchange formats or co-simulation improve synchronization especially in later phases. But an overall engineering methodology combining the advantages of both worlds, i.e. providing models for early stage analysis and design decisions while defining a clear path towards refined design documents and co-simulation at later stages is still missing. To meet this challenge, a novel integrated development approach is presented in the following.

## III. The IMoMeSA Mechatronic Systems Engineering Methodology: An Overview

The development methodology presented in this paper differentiates between two phases: The *conception* and the *refinement* phase (see Fig. 1). The *conception* phase starts with a product idea, from which subsequently a mechatronic concept is derived using a dedicated systems modeling technique. The general purpose of the conception phase is to gain a shared understanding of the system under development among the participating engineering disciplines. Integral parts

of the conception phase are the modeling of requirements, the definition of the high-level system structure/architecture, as well as the formulation of the approximate system behavior. Furthermore, mechatronic model simulation/testing is used for validation and verification as well as consistency checking throughout the development.

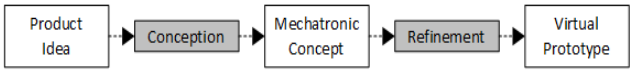


Fig. 1. Overview of the mechatronic systems engineering process [7], [9].

During *refinement* the mechatronic concept is elaborated from a mere concept to a complete virtual prototype including all relevant design documents required for manufacturing the mechatronic system. For elaboration the original modeling technique is extended with additional modeling concepts. At this point also the link to established engineering tools is achieved, which represents a necessary prerequisite for adoption of the methodology in practice. Finally, simulation is used again throughout development to reveal and manage potential problems of the system design.

The conception phase was described in detail in [7] and [9]. In this paper, only the necessary foundations regarding the systems modeling technique and the simulation approach are introduced. The main focus of this paper is the refinement phase. Therefore, a new methodology of how mechatronic concepts can be elaborated systematically is presented. In particular, the use of discipline-specific engineering tools and the systems modeling technique in combination is described and the application of state-of-the art data exchange and co-simulation principles is outlined.

## IV. The Conception Phase: From Product Idea to Mechatronic Concept

*Conception* refers to the development of a simplified mechatronic system model, which can be used to describe and analyze requirements as well as early design decisions across the involved engineering disciplines. As presented in [7], [9], a component-based modeling technique prototypically implemented in the "IMoMeSA Modeller" engineering tool is proposed (see Fig. 2).

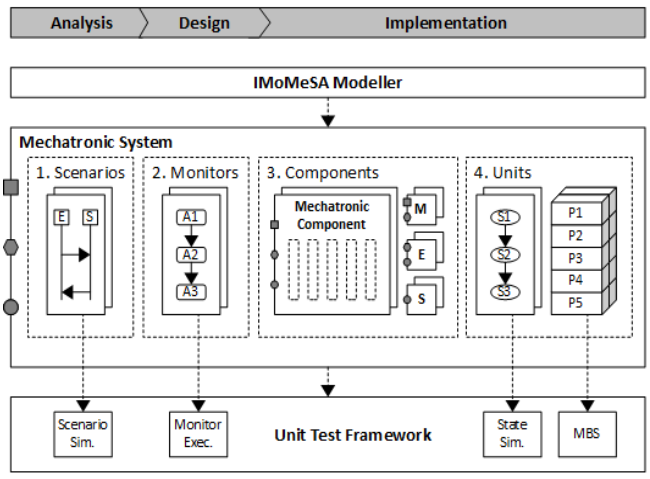


Fig. 2. Development activities, editors, contents [7], [9], and simulators during the conception phase.

This technique provides different views for each development activity. During system analysis the *interface* of the mechatronic system as well as scenarios of interaction between environment and system are captured. The interface is modeled in terms of material, energy and data ports, while scenarios consist of steps, actions and conditions. Subsequently, during system design *monitors* can be added to specify the desired system behavior. As detailed in [40], monitors are composed of activities, transitions and constraints. Then, during system implementation the system can be decomposed into mechatronic or discipline-specific *components*. For each component the same views can be used as during analysis, design and implementation creating a hierarchical structure of the mechatronic system. Finally, the implementation of atomic components (i.e. components without sub-components) can be defined in terms of *units*. For electronic and software components discrete-time I/O automata [41] can be used describing data and energy flow in terms of states, transitions, guards and actions. For mechanic components rigid parts and joints are provided.

For validation and verification purposes the implementation is tested with respect to the scenarios as described in [7]: Each scenario is transformed into a test case. The test case simulates the respective scenario as well as the behaviors and mechanics added during implementation. In the meantime, the monitors are used to track the simulation and check for constraint violations. In case constraints are violated, the test case fails and the violations are reported to the unit testing framework. Otherwise the test case succeeds. Note that at this development stage physics simulation is intentionally limited to base geometries (e.g. spheres or boxes) and rigid body dynamics in order to focus on interdisciplinary development insights rather than detailed mechanical design.

## V. The Refinement Phase: From Mechatronic Concept to Virtual Prototype

As described in the previous section, during conception a simplified model of the system is built neglecting, in particular, actual physical and temporal behavior. The model should be sufficient to validate and verify critical design decisions, while correctness with respect to real-world behavior cannot be guaranteed. To address this issue, during refinement modeling constructs, editors and simulators are added which allow to describe and analyze the mechanic, electric/electronic and software behavior more accurately (see Fig. 3). In the following, the *model extensions* required with respect to the original approach are presented, before the necessary *editor* and *simulator integrations* are outlined. Finally, a systematic *development procedure* on top of the revised modeling and simulation framework is proposed.

### A. Model Extensions

The enhancements of the original modeling technique do not concern the *scenarios*, *monitors* and *components* views of the mechatronic system model. Rather a number of *units* are added. Depending on the purpose of the unit within the mechatronic systems engineering process a distinction between *simulation* and *deployment units* is proposed: As the name suggests, simulation units are meant for computer-based

simulation and testing of the mechatronic system model only. The simulation units include hybrid rather than the original discrete-time I/O automata [42], differential equations as known from the Modelica language [36] and custom code (e.g. using the Java programming language [43]). Both hybrid I/O automata and differential equations allow describing discrete- and continuous-time behavior. However, hybrid I/O automata need to specify the direction of data flow between components, while Modelica-style differential equations allow defining equation systems across component boundaries without flow direction. Note that the choice of modeling technique depends on the actual problem at hand. Finally, custom code can be used to integrate all sorts of simulations not supported by the other techniques.

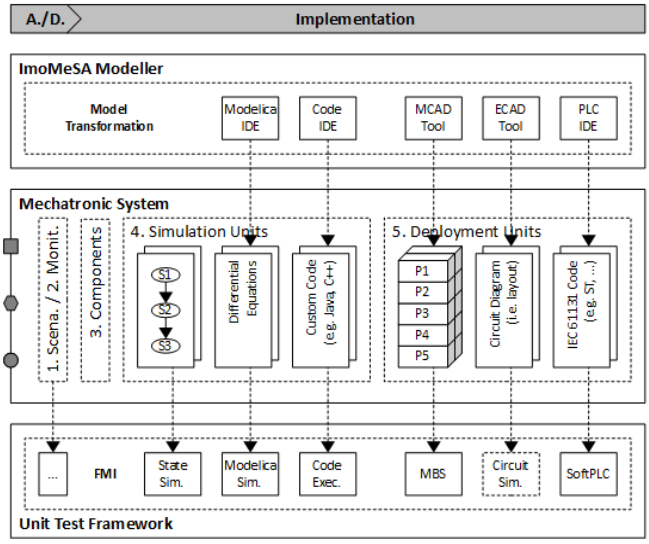


Fig. 3. Development activities, editors, contents and simulators during the refinement phase.

On the other hand, deployment units actually need to be elaborated to manufacture and assemble the real-world system. The deployment units include rigid parts and joints already known from conception for mechanic components. The parts and joints are complemented by circuit diagrams for electric/electronic components defining their geometric layout as well as standard IEC 61131code [44] for example in Structured Text (ST) format for software components.

### B. Editor Integrations

The major focus of this article is the integration of mechanical (i.e. MCAD) and electrical/electronical (i.e. ECAD) computer-aided design tools. For this purpose the tool integration strategy is shown in Fig. 4.

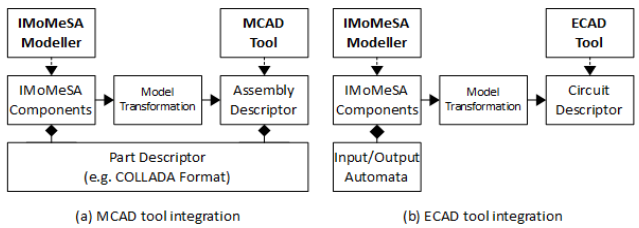


Fig. 4. Integration of the "IMoMeSA Modeller "with MCAD/ECAD tools using model transformations.

In case of MCAD an *assembly descriptor* [16] is generated from the mechatronic component hierarchy including an identical assembly hierarchy, joint descriptors and references to part descriptors. Joint descriptors define the linkage between parts. Part descriptors define the rigid building blocks of the mechanical structure and are stored in the COLLADA format [25]. Note that the mechatronic components reference the same part descriptors as the assembly descriptors. Consequently, modifications on parts are directly synchronized between the "IMoMeSA Modeller" and the MCAD tools. In contrast, modifications on the assembly hierarchy must be performed on the mechatronic component hierarchy inside the "IMoMeSA Modeller" and synchronized by means of model transformation. This policy has been introduced to prevent inconsistencies between the mechatronic component and the assembly hierarchy. However, in the future automated synchronization strategies might be used instead allowing modifications to the assembly hierarchy inside the MCAD tools directly.

In case of ECAD a *circuit descriptor* [16] is built from the mechatronic component hierarchy including circuit diagram macros for all mechatronic, electric and electronic components. Note that during conception the electrical/ electronic design can be reduced to clamps as well as sensors and actuators. Clamps provide data input and electric energy output ports or vice versa. In contrast, sensor and actuators use electric energy input or output and material ports. However, in few cases intermediate components with only electric energy input and output ports might be used during conception as well. During refinement the ECAD tool is used to define the geometric layout of the components. The geometric layout includes the position and orientation of electric or electronic building blocks as well as the wire tracks. Similar to the MCAD case at the current stage new electric/electronic building blocks and logical connections have to be introduced in the "IMoMeSA Modeller" to ensure consistency. Finally, note that the behavior specification of electric/electronic components is not part of the transformation. This limitation is due to the fact that ECAD tools typically are concerned with the geometric circuit layout only, while circuit behavior is omitted.

A minor focus of this article is the integration with PLC IDEs and simulation unit editors (i.e. Modelica and Code IDEs). Concepts for the integration between the "IMoMeSA Modeller" and PLC IDEs based on model transformation were presented previously [9]. The application of similar concepts for the seamless integration of simulation unit editors is intended in the future. For now, a manual transformation and integration step is to be assumed.

### C. Simulator Integrations

For validation, again a simulation-based approach is employed reusing the *scenarios* and *monitors*, which have been developed during conception and need to be adapted potentially during refinement. Due to the increased number of units and their diverse simulation semantics a number of different simulation tools have to be integrated during refinement. More specifically, the state and multi-body simulation tools already used during conception can be applied again. But, these tools now are complemented by a SoftPLC [44] for IEC 61131 code execution as well as a Modelica simulator for differential equation solving [35] and an environment for custom code execution [45]. Optionally,

also a dedicated circuit simulator such as SPICE [46] can be employed in case the ECAD circuit descriptor format is supported or a trans- formation to such format exists. In case the circuit simulator cannot be used, I/O automata, differential equations or custom code can be employed instead for describing and simulating electric/electronic behavior. For simulator tool integration the Functional Mockup Interface (FMI) [39] is applied.

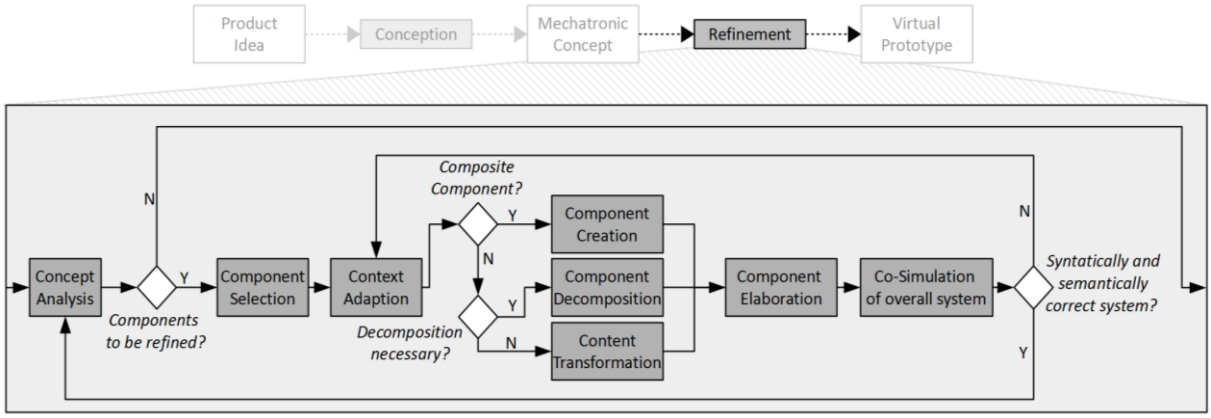


Fig. 5. Systematic development procedure for the refinement phase.

### D. Development Procedure

Based on the outlined model extensions and editor/ simulator integrations, finally a development procedure, which defines how to use the modeling technique during refinement, is presented. Basically, an incremental procedure is proposed, where several components are refined successively (see Fig. 5). Therefore, the analysis of the mechatronic concept is recommended as a first step. During this step one can browse the system model and look for components, which are either not deployable directly (e.g. because of a simplified geometry) or where automata do not represent the components' behavior sufficiently in order to secure further development decisions. As a result, one can answer the question whether components exist that need to be refined. If so, a specific component can be selected typically beginning with the most critical one regarding its dependencies within the overall mechatronic system. This component can be refined subsequently by carrying out the following steps of the development procedure.

Regardless of the component type (i.e. composite or atomic), usually it is necessary first to adapt the component's context for the respective refinement task. This adaption may include an enhancement of the component's interface or the modification of belonging scenarios and monitors [1]. Subsequently, the component itself can be refined. Therefore, three general refinement options can be distinguished depending on the component type and the complexity of the refinement task. If the component under consideration is composite already, refinement means the creation of one or more components within the existing sub-component structure and the elaboration of these components using simulation or deployment units. If on the other hand the component is atomic, one has to decide whether the particular refinement task is complex enough such that further decomposition is required. If that is the case, the respective component has to be divided into sub-components and the conception units have to be re-implemented as a composition of these sub-components again using simulation and deployment units. If alternatively no decomposition is needed, refinement means the substitution of the conception units by a more detailed specification. Therefore, the original conception units can be transformed to the preferred simulation or deployment units. The transformation result can be used as a basis for the component elaboration. This case could occur, for example, if the I/O automaton of a software component only captures the desired behavior partially. This automaton can be transformed to IEC 61131 code, which then can be refined in a PLC IDE using so called entry points (see [9]).

Once the component under consideration is elaborated completely, a co-simulation of the overall system can be performed to check for syntactic and semantic correctness [47]. In particular, the compilation and simulation shows whether new components were embedded sufficiently (i.e. syntactic correctness) and whether the refined mechatronic system model still obeys the scenarios, monitors and constraints (i.e. semantic correctness). If one of these aspects is not met, the procedure proposes an iterative process beginning with a revision of the context adaption. If on the other hand the simulation succeeds, the refinement for this component is completed and the incremental procedure can start again with the analysis of the mechatronic concept. Once there are no more components to be refined (i.e. all deployment units are elaborated), the refinement is complete resulting in a fully functional virtual prototype of the mechatronic system.

## VI. Conception and Refinement of Mechatronic Systems in Practice: Exemplary Application

To demonstrate the ideas presented in the previous sections, three practical examples are outlined in the following (see Fig. 6). These examples were derived from the three refinement options of the development procedure presented above. Furthermore, the examples illustrate the refinement of

[1] Especially the interface adaption already contributes to consistency preservation since the enhanced interface has to be embedded appropriately in the overall system context resulting in refinement tasks for other disciplines.

mechatronic as well as pure mechanic, electric/electronic and software components.

In case of Fig. 6(a) the geometry of a *standard workpiece* component is modified when going from the conception to the refinement phase. During conception the basic geometry is sufficient to simulate the mechanic behavior of the workpiece within the mechatronic system. In particular, the model granularity allows to rule out collisions between certain parts or to prove the adequacy of kinematic forces. However, during refinement more detailed manipulations on the surface of the standard workpiece need to be designed and tested. For this reason, the geometry of the workpiece is refined using a standard MCAD tool. From then on the refined geometry can be used in multi-body simulations instead of its coarse approximation.

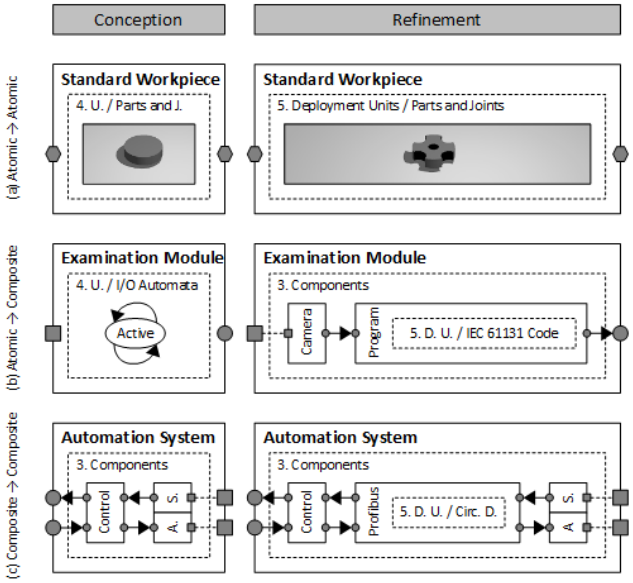


Fig. 6. Overview of different refinement scenarios resulting from the development procedure.

In case of Fig. 6(b) the I/O automaton of an *examination module* is elaborated when moving to the refinement phase. During conception the I/O automaton is sufficient to describe the behavior of the examination module, i.e. examining the state of some workpiece found at a particular location. On a conceptual level modeling the state of the workpiece and its examination can be achieved by means of state machines and material ports easily. However, during refinement a solution has to be developed which can be deployed to a real-world system. In this case a camera-based solution complemented with a computer vision algorithm is selected. For computer-based testing the camera is implemented using a custom code simulation unit integrating Pov-RAY [48] for realistic 3D image rendering. The computer vision algorithm is specified in IEC 61131 code instead. Consequently, the algorithm can be tested against synthetic images in early stages while moving to real-world samples later.

Finally, in case of Fig. 6(c) the sub-components of an *automation system* are extended while working on the refinement of the mechatronic system model. During conception only the software-based control component as well as mechatronic sensors and actuators are defined. This way the entire event chain from mechanics over electrics/electronics to software can be described and tested on a conceptual level. While this level of detail is sufficient for designing the coarse structure and behavior of the mechatronic system, critical real-world phenomena such as communication delays are not considered appropriately. Due to this limitation during refinement a Profibus [49] is added between the control and sensor/actuator components. This component can be used to describe the respective communication delays, such that the new phenomenon can be accounted for in control software design and verification. Note that the behavior of the Profibus can be described using, for example, I/O automata or custom code simulation units. Additionally, a circuit diagram deployment unit can be added to describe its geometric layout for manufacturing the mechatronic system.

## VII. Conclusion and Outlook

In this paper a model-based engineering methodology for mechatronic systems leading from a product idea over a mechatronic concept to a complete virtual prototype was presented. During conception critical design decisions are taken to form the mechatronic concept, before elaborating a complete virtual prototype during refinement. For conception a systems modeling technique is used that is able to capture requirements as well as the high-level mechatronic system structure and the approximate behavior. Subsequently, a dedicated methodology for refinement was introduced, which extends the original modeling technique by various units for deployment (e.g. circuit diagrams, IEC 61131 code) and simulation (e.g. hybrid I/O automata, differential equations). Finally, the refinement was shown for three practical examples derived from the defined refinement options.

The presented engineering methodology supports the complete development process of mechatronic systems by seamlessly combining interdisciplinary modeling aspects with discipline-specific refinement. In particular, this approach allows the continuous and automated evaluation of modeled scenarios, monitors and constraints in order to secure decisions and avoid mistakes across discipline borders. Since model transformations and editor integration are proposed as one essential step, a high degree of automation during the transition from conception to refinement can be guaranteed and thus the initial modeling effort can be justified by later savings. Finally, this approach proposes and defines the use of various state-of-the-art technologies (e.g. co-simulation) within one development procedure and thus combines their individual advantages for mechatronic systems engineering.

However, it should be noted that the presented refinement methodology has not been implemented completely so far: Various state-of-the-art technologies are used, but the possibilities for integration within the presented approach have to be clarified. For this reason, only three practical examples could be presented for evaluation purposes while a complete case study is currently missing. Thus, the functionality of the methodology could be demonstrated only with restrictions. Further research tasks will address these drawbacks: The technical realization of the introduced model transformations and simulator integration is currently elaborated. Furthermore, the case study of an industrial-like stamping component will be carried out in order to evaluate and verify this approach within a practical development task.

## REFERENCES


[1] M. Garetti and M. Taisc, "Sustainable manufacturing: Trends and research challenges," *Production and Control*, vol. 23, pp. 83-104, 2012,

[2] K. Clark and S. Wheelwright, *Managing New Product and Process Development: Text Cases*, the Free Press, 1993.

[3] J. Gausemeier, R. Dumitrescu, and D. Steffen, "Systems engineering in der industriellen praxis," Heinz-Nixdorf-Institut and the Fraunhofer IPT — Projektgruppe Entwurfstechnik Mechatronik, 2013.

[4] W. Dohmen, "Interdisziplinäre Methoden für die integrierte Entwicklung komplexer mechatronischer Systeme," Ph.D. dissertation, Technical University of Munich, 2002.

[5] H. Diehl, "Systemorientierte Visualisierung disziplinübergreifender Entwicklungsabhängigkeiten mechatronischer automobilsysteme," Ph.D. dissertation, Technical University of Munich, 2009.

[6] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML — the glue for seamless automation engineering," in *Proc. IEEE International Conference on Emerging Technologies and Factory Automation*, Hamburg, 2008.

[7] M. Eigner, "Modellbasierte virtuelle produktentwicklung auf einer plattform für system lifecycle management," in *Industrie 4.0 — Beherrschung der industriellen Komplexität mit SysLM*, U. Sendler, Ed. Springer, 2013, pp. 91-110.

[8] G. Hackenberg, C. Richter, and M. F. Zäh, "A multi-disciplinary modeling technique for requirements management in mechatronic systems engineering," in *Proc. the 2nd Joint Internal Conference on System-Integrated Intelligence*: *New Challenges for Product and Production Engineering*, Bremen, 2014, pp. 5-16

[9] C. Richter, G. Hackenberg, and M. F. Zäh, "Interdisziplinäre Funktionsmodellierung für die Generierung von robustem Steuerungscode für fehlertolerantes Systemverhalten," in *Proc. Industry meeting of Measurement and Automatic Control*, 2014.

[10] T. Stahl and M. Völter, *Model-Driven Software Development*, John-Wiley & Sons Ltd., 2006.

[11] T. Weilkiens, "Zukunftsdisziplin Modellbasiertes Systems Engineering," in *Proc. Paderborner Workshop — Design of mechatronic systems*, 2011, p. 8.

[12] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML — The Systems Modeling Language*, Morgan Kaufmann, 2012

[13] M. Gehrke, "Entwurf mechatronischer systeme auf basis von funktionshierarchien und systemstrukturen," Ph.D. dissertation, University of Paderborn, 2005.

[14] VDI, *Entwicklungsmethodik Für Mechatronische Systeme*, Verein Deutscher Ingenieure, Beuth-Verlag, 2004.

[15] M. Litto, "AQUIMO — Ein Leitfaden für maschinen- und anlagenbauer," Verein Deutscher Maschinen — und Anlagenbau, VDMA-Verlag, 2010.

[16] J. Botaschanjan, T. Hensel, B. Hummel, and A. Lindworsky, "AutoVIBN: Automatische Generierung von Verhaltensmodellen für die qualitätsorientierte Virtuelle Intbetriebnahme," Technical Report TUM-I1012, Technische Univsersität München (Abschlussbericht — AiF-Forschungsvorhaben ZN 279), 2010.

[17] M. F. Zaeh and A. Lindworsky, "Automatic model generation for virtual commissioning," in *Proc. the International Conference on Competitive Manufacturing*, Paris, 2010, pp. 27-32.

[18] B. Hummel, "Integrated begavior modeling of space-intensive mechatronic systems," Ph.D. dissertation, Technical University of Munich, 2011.

[19] T. Hensel, "Modellbasierter Entwicklungsprozess für Automatisierungs-lösungen," Ph.D. dissertation, Technical University of Munich, 2011.

[20] J. Gausemeier, U. Frank, J. Donoth, and S. Kahl, "Specification technique for the description of self-optimizing mechatronic systems," *Research in Engineering Design*, vol. 20, pp. 201-223, 2009.

[21] J. Estefan, *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, INCOSE MBSE Focus Group 25, 2008.

[22] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proc. the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003, Anaheim, CA, USA.

[23] M. J. Pratt, "Introduction to ISO 10303 — The STEP standard for product data exchange," *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, pp. 102-103, 2001.

[24] U. Bracht, S. Wenzel, and D. Geckler, *Digitale Fabrik: Methoden und Praxisbeispiele*, VDI-Buch, Springer, 2009.

[25] R. Arnaud and M. C. Barnes, *COLLADA: Sailing the Gulf of 3D Digital Content Creation*, Wellesley, AK Peters, 2006.

[26] P. J. Ashenden, *The Designer's Guide to VHDL*, Morgan Kaufmann, 2010.

[27] M. Marcos, E. Estevez, F. Perez, and E. van der Wal, "XML exchange of control programs," *Industrial Electronics Magazine*, vol. 3, no. 4, pp. 32-35, 2009.

[28] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. (2006). Extensible markup language (XML). World Wide Web Consortium Recommendation. [Online]. Available: http://www.w3.org/TR/xml11/>

[29] R. Sinha, V. C. Liang, C. J. J. Paredis, and P. K. Khosla, "Modeling and simulation methods for design of engineering systems," *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, pp. 84-91, 2001.

[30] P. Stich and G. Reinhart, "Mechatronic sketching of manufacturing systems using physically based models," in *Proc. IEEE Symposium on Industrial Electronics and Applications* (*ISIEA*), Kuching, September 2013, pp. 1-6.

[31] S. Völker and P. M. Schmidt, "Simulationsbasierte Optimierung von Produktions- und Logistiksystemen mit Tecnomatix Plant Simulation," in *Integrationsaspekte der Simulation: Technik, Organisation und Personal*, G. Zülch and P. Stock, Eds. KIT SCI Publication, 2010.

[32] S. Röck and G. Pritschow, "Real-time capable finite element models with closed-loop control — A method for Hardware-in-the-Loop simulation of flexible systems," *Production Engineering-Research & Development*, vol. 1, no. 1, pp. 37-43, 2007.

[33] G. Reinhart and M. Weissenberger, "Multibody simulation of machine tools as mechatronic systems for optimization of motion dynamics in the design process," in *Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Atlanta, GA, USA, 1999, pp. 605-610.

[34] G. Reinhart and F. F. Lacour, "Physically based virtual commissioning of material flow intensive manufacturing plants," in *Proc. 3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production*, 2009, pp. 377-387.

[35] Dymola. (2014). [Online]. Available: http://www.modelon.com/products/dymola/

[36] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, John Wiley & Sons, 2010.

[37] M. A. Groothuis, A. S. Damstra, and J. F. Broenink, "Virtual prototyping through co-simulation of a Cartesian plotter," in *Proc. IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 697-700.

[38] T. Brezina, Z. Hadas, and J. Vetiska, "Using of co-simulation ADAMS-SIMULINK for development of mechatronic systems," in *Proc. 14th International Symposium on Mechatronika*, IEEE, 2011, pp. 59-64.

[39] T. Blochwicz, M. Otter, J. Akesson, M. Arnold, C. Clauß, H. Elmquist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel, "Functional mockup interface 2.0: The standard for Tool independent exchange of simulation models," in *Proc. 9th International Modelica Conference*, 2012.

[40] G. Hackenberg, A. Campetelli, C. Legat, J. Mund, S. Teufl, and B. Vogel-Heuser, "Formal technical process specification and verification for automated production systems," in *Proc. the 8th International Conference on System Analysis and Modeling*, 2014.

[41] N. A. Lynch and M. R. Tuttle, "An introduction to input/output automata," Technical Report MIT/LCS/TM-373, MIT Laboratory for Computer Science, 1988.

[42] N. A. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Information and Computation*, vol. 185, no. 1, pp. 105-157, 2003.

[43] K. Arnold, J. Gosling, and D. Holmes, *The Java Programming Language*, Reading: Addison-Wesley, vol. 2, 1996.

[44] H. Lepers, *SPS-Programmierung Nach IEC 61131-3*, Franzis Verlag, 2005.

[45] B. Venners, *Inside the Java Virtual Machine*, McGraw-Hill Inc., 1996.

[46] S. M. Sandler and C. E. Hymowitz, *SPICE Circuit Handbook*, McGraw-Hill Professional, 2006.

[47] M. Broy and K. Stølen, *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*, Springer, 2001.

[48] Persistence of Vision Raytracer. (2004). [Online]. Available: http://www.povray.org/

[49] E. Tovar and F. Vasques, "Real-time fieldbus communications using Profibus networks," *IEEE Transactions on Industrial Electronics*, vol. 46, issue 6, pp. 1241-1251, August 2002.

**Georg Hackenberg** was born in Füssen, Germany on August 18, 1983. He holds a bachelor of science degree in software and internet technology from University of Mannheim, Mannheim, Germany, in 2007 and a master of science degree in software systems engineering from RWTH Aachen University, Aachen, Germany, 2010.

From 2004 to 2006, he worked as a student assistant at the Database Group, University of Mannheim, Mannheim, Germany. From 2006 to 2007, Georg worked as a trainee at the Integrated Data Systems Group, Siemens Corporate Research, Princeton, USA. From 2008 to 2009, he worked as a student assistant at the Chair for Information Systems, RWTH Aachen University, Aachen, Germany. From 2009 to 2010, he worked as a student assistant at the Cooperation Systems Group and as a researcher at the Risk Management and Decision Support Group, Fraunhofer Institute for Applied Information Systems FIT, Sankt Augustin, Germany. From 2011 onwards, Georg works as a researcher assistant at the Chair for Software and Systems Engineering, Technische Universität München, Garching, Germany. His current research interests include model-based development of manufacturing and energy systems as well as large-scale discrete-time optimal control.

Mr. Hackenberg is a student member at the Institute of Electrical and Electronics Engineers (IEEE). In 2011 he won the Hugo Geiger Price for an outstanding Master thesis at the Cooperation Systems Group, Fraunhofer Institute for Applied Information Systems FIT, Sankt Augustin, Germany. In 2012 he earned a Software Campus Fellowship for his research on model-based development of smart energy systems.

**Christoph Richter** was born in Bamberg, Germany on September 12, 1986. He holds a mechanical engineering diploma from Technische Universität München, Garching, Germany, 2012.

During his studies from 2006 to 2012 he temporarily worked as a student assistant at the Chair of Automatic Control and the Chair of Thermodynamics at the Department of Mechanical Engineering, Technische Universität München, Garching, Germany. From 2010 to 2012, he worked as a student trainee at the ITQ GmbH, Garching, Germany. After finishing his studies in 2012, he worked as a research assistant at the Institute for Machine Tools, Technische Universität München, Garching, Germany. His main research focus was on the model-based development of mechatronic systems. Since January 2015, he works as a research assistant at the Fraunhofer Institute for Machine Tools and Forming Technology, Augsburg, Germany, where he concentrates his research activities on the model-based development of user interfaces in the field of mechanical engineering.

**Michael F. Zaeh** was born in Coburg, Germany on December 11, 1963. He is a full-time professor for machine tools and manufacturing technology at the Department of Mechanical Engineering, Technische Universität München, Garching, Germany.

After his studies in mechanical engineering at Technische Universität München from 1984-1989, he assumed a position as a graduate research assistant at the Institute of Machine Tools and Industrial Management of Technische Universität München (iwb). He received his doctorate degree in 1993 with a thesis about a dynamic process model for circular sawing. Form 1994 to 1995, he had been the head of a department under the new director of the institute, Prof. Dr. Gunther Reinhart. In 1996 Michael Zaeh joined Gleason-Pfauter Maschinenfabrik GmbH, a builder of machine tools for manufacturing of gears, where he worked in the research laboratory and in the design department during the first year. Later he was promoted head of the order management department. Since 1997 he was a member of the management group. From 1998 to 2002, he worked in an ERP-System-Implementation Project at several production sites and sales agencies of the Gleason Corporation. During that time he stayed in Rochester, N.Y., USA, for two years and he worked also in South Korea, Great Britain, Brazil and Japan. In the year of 2002 he accepted the position of full-time professor for machine tools and manufacturing technology at Technische Universität München, Garching, Germany.

Prof. Dr. Michael Zaeh is a member of ACATECH, the German academy of technical science, of the WGP (German Scientific Society for Production Technology), of the WLT (Scientific Society of Laser Technology) as well as of several other institutions. He also holds an associated membership at CIRP. Suggested by professor Dr. Joachim Milberg the Diploma Thesis of Prof. Dr. Zaeh was awarded the student prize of the VDW, the German machine tool builders' association.