Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-12 (2016) 1175–1180

# MaCon: Consistent Cross-Disciplinary Conception of Manufacturing Systems

**Georg Hackenberg** * **Mario Gleirscher** * **Thomas Stocker** *
**Christoph Richter** ** **Gunther Reinhart** **

* *Technische Universität München, Garching, Germany (e-mail: georg.hackenberg,mario.gleirscher,thomas.stocker@in.tum.de).*
** *Fraunhofer Institute, Augsburg, Germany (e-mail: christoph.richter,gunther.reinhart@iwu.fraunhofer.de).*

**Abstract:** *Conceptual designs* of *manufacturing systems* contain input from all engineering disciplines involved in the life cycle of such systems. For their efficient and consistent conception, information about the *elements* of a conceptual design and their *relationships* have to be *sufficiently precise* but *preferably abstract*. We employ a cross-disciplinary modeling technique to capture this information together with consistency checking including simulation to accomplish such a challenging task. In this paper, we demonstrate our approach with a *tool prototype*. Based on an industrial example, we highlight how the features of the tool prototype can effectively support tasks in conceptual design. To illustrate this effectiveness, we visualize the modeling work flow and the results of the consistency checks based on data collected during tool usage.



*Keywords:* Manufacturing, mechatronics, engineering, modeling, simulation, testing.

## 1. INTRODUCTION

In the manufacturing domain, engineers deal with the design of plants of the *systems of systems* type. Such systems are built from simple parts up to complex modules delivered by various independent equipment suppliers. These systems also require *cross-disciplinary collaboration* of, e.g. mechanical, electrical and software engineers. Their distributed control subsystem is not only strongly coupled via the modules of the plant but also via services from outside the plant, e.g. from the Internet. Reinhart and Wünsch (2007) investigated that *intensified software-based control* can increase the risk of failures during commissioning. Gausemeier et al. (2013) criticizes the expressiveness of existing engineering methods. Correspondingly, we perceive the *consistency* of *manufacturing system concepts*, i.e. the requirements specification and conceptual design, as an important prerequisite for successful commissioning. Additionally, the realization of such a concept in terms of a plant can be required to be *operable* for a *long time span*. In this context, engineers have to deal with *modifications* and *evolution* of legacy plants. Many of these tasks have to be supported by *virtual commissioning* to prematurely understand *critical plant aspects*, such as manufacturing resource optimization, occupational and system safety, security against unauthorized influence on the process.

These challenges require achieving an *early* and *clear understanding* of the manufacturing system concept. This implies an easy access to important relationships among the elements (i) of the *automated manufacturing process* (e.g. the nominal fabrication and assembly activities, plant incident handling, ramp-up and shut down procedures) and (ii) of the *technological* and *personnel setting* implementing this process (i.e. machine tools, industrial handling equipment, raw material supply chain, control system, operating personnel).

We perceive large potential for improvements in this direction: Section 2 discusses related engineering methods and tool suites. Sections 3 describes the problem we want to solve as well as our contribution.

## 2. RELATED WORK

Here, we review work in the field of cross-disciplinary model-based (mechatronic) systems engineering.

*MATLAB*[1] *Simulink* is probably the de facto standard for modeling and simulation of mechatronic systems. The tool supports specification of continuous system dynamics. The focus lies on control engineering (e.g. dealing with vibrations of mechanical components or the temperature of liquids). With *MATLAB Stateflow* the continuous system dynamics can be integrated with discrete-event logics (e.g. to model control software). The *MATLAB Simulink Design Verifier* supports formal requirements specification and verification. While the tool chain covers a wide range of cross-disciplinary engineering tasks, (technical) process and geometry specification (including their impact on simulation and verification) is not integrated properly.

In *hybrid systems theory* (Henzinger, 1996) numerous techniques have been proposed for cross-disciplinary modeling and verification. While these techniques provide a strong notion of consistency, their scope typically does not include manufacturing conception and work flow support for geometry and material flow specification. We exclude these techniques from detailed investigations because we discuss several applications thereof in the following.

[1] http://www.mathworks.com


10.1016/j.ifacol.2016.07.665

*Modelica*, described by Fritzson (2014), is a popular language and tool similar to the *MATLAB* tool chain, supporting continuous system dynamics and discrete-event logics. Additionally, *Modelica* provides a basic mechanism for integrating component geometry. However, the geometry information is only used for visualization and the other drawbacks of the *MATLAB* tool chain mostly remain.

To simultaneously enhance formal semantics and increase expressiveness, Botaschanjan et al. (2009); Botaschanjan and Hummel (2010) proposed a formal specification technique including geometric aspects as well as tool support. Their tool is based on *AutoFOCUS*, which has been developed originally by Broy et al. (1999) for modeling software architectures and their discrete-event logics. *AutoFOCUS* provides requirements management capabilities, which have been introduced by Teufl et al. (2013). However, the requirements and geometry modeling extensions are not compatible with each other. Consequently, respective consistency checks are not supported. Moreover, aspects like specification of the (technical) manufacturing process are not integrated properly.

A comprehensive tool chain is provided by *Siemens*[2] *Team Center (TC)* and *Siemens Mechatronics Concept Designer (MCD)*. While TC provides means for requirements, technical process, and mechatronic architecture specification, MCD supports modeling component geometry, continuous system dynamics, and discrete-event logics. Consequently, the tool chain covers many engineering tasks properly. However, especially the TC models remain informal hindering rigorous automated consistency checks.

*SysML*, as explained by Friedenthal et al. (2014), is the de facto standard for cross-disciplinary requirements specification and system design. *SysML* provides a wide range of diagrams covering many aspects of early engineering phases. However, *SysML* by itself is not based on formal theory, which hinders rigorous automated syntactic and semantic consistency checking. Moreover, geometry specification is not supported, which represents an important aspect in the manufacturing domain.

To overcome the problem of formal semantics, Burmester et al. (2005) proposed *MechatronicUML*. Adding formal semantics enables the authors to implement automated consistency checks and, for example, code generation. However, the formal semantics comes with reduced language expressiveness, such that only a subset of engineering tasks can be supported effectively. In particular, the geometry of mechanical components cannot be specified properly.

To overcome the problem of reduced expressiveness, Rehage et al. (2014) proposed the method *CONSENS*, while Gausemeier et al. (2010) provided respective tool support. With a similar goal in mind, Eigner (2013) proposed an integration of *SysML* with product life cycle management solutions. Their approaches aim to cover a wide range of engineering tasks including geometry specification but lack seamless formal semantics. Consequently, many aspects of the system under development can be specified, but the overall consistency of the specification cannot be checked automatically.

[2] http://www.plm.automation.siemens.com

## 3. PROBLEM AND CONTRIBUTION

As sketched in the previous section, recent tools are capable of integrating (formal) requirements, (technical) process, (mechatronic) architecture, mechanics, electrics and software modeling. However, automated consistency checking support is limited due to the gaps within the underlying formalisms. Consequently, inconsistencies within mechatronic system designs cannot be discovered easily leading in the "best" case to (undocumented) quick fixes and in the "worst" case to expensive iterations encompassing the advanced development and realization process.

To overcome this situation, we have been developing the novel tool prototype MaCon[3] based on the (conceptual) mechatronic/manufacturing modeling and consistency checking technique described in Hackenberg et al. (2014b,a, 2015). In this paper we briefly explain the prototypical tooling in Section 4. Then, we show the tool in action based on an industrial example in Section 5 as well as an analysis of the engineering work flow in Section 6. Finally, we provide a conclusion and outlook to further work in Section 7.

## 4. PROTOTYPICAL TOOLING

The prototypical tooling, explained in this section, includes a representation of the meta-model explained in Hackenberg et al. (2015), an engine checking the syntax rules, a simulation/test engine checking the semantics rules, and a user interface for modeling and test execution. The user interface of the prototypical tooling is shown in Figure 1. The interface consists of six *tabs*: The *explorer*, *editor*, *scene*, *attribute*, *marker*, and *change* tabs.

The *explorer* tab shows all model elements as a tree or containment hierarchy (e.g. workspace contains projects, projects contain components). In case of multiple child elements of the same type (e.g. requirements inside components), folder tree nodes are introduced to improve readability. Ultimately, the tree view provides all aspects of the approach: Informal and formal requirements, (technical) process, (mechatronic) architecture, as well as mechanics, electrics, and software (Hackenberg et al., 2015).

The *editor* tab shows an editor for the element selected in the *explorer* tab. E.g. for the workspace, projects, and components a management view is presented, which allows one to execute all contained scenarios and generate test reports. For scenarios, monitors, and behaviors a state machine view is presented instead, which allows one to create and layout steps, activities, states, and transitions. For a component's (sub-)components, an architecture view is presented, which allows to connect the ports of (sub-)components and material interaction ports. Note that material interaction ports are presented as slots in the architecture view (see *orange* boxes in Figure 7), where other components can dock spontaneously upon collision.

The *scene* tab shows the parts as well as material entry, exit, and interaction ports of the selected component and all its subcomponents. The concept of ports is described in Hackenberg et al. (2015).

[3] https://youtu.be/aPYt2ajO-FQ

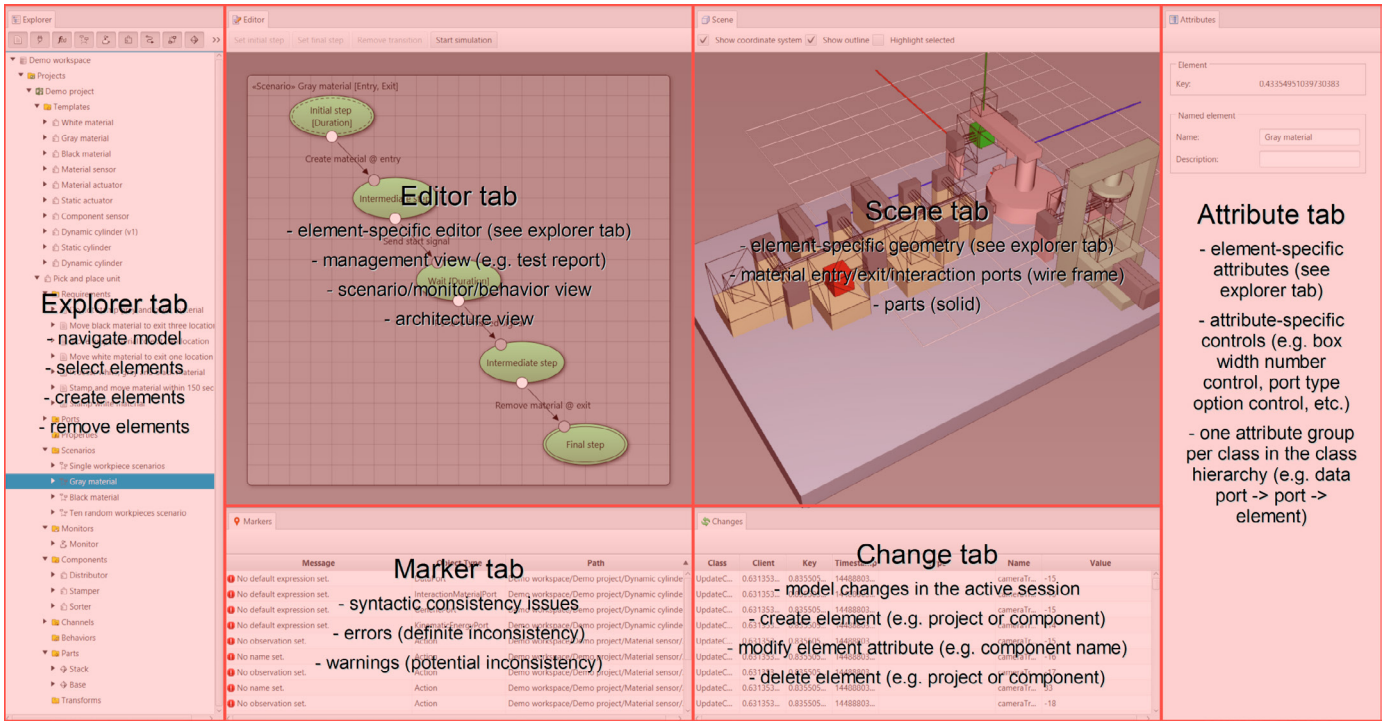



Fig. 1. User interface of the prototypical tooling including *explorer*, *editor*, *scene*, *attribute*, *marker*, and *change* tabs.

The *attribute* tab shows attributes of the element selected in the *explorer* or *editor* tabs. In particular, attributes such as the width, length, and height of geometries or the type and direction of ports can be modified. Furthermore, the attributes are grouped according to the inheritance hierarchy of the respective element type (e.g. the class *DataPort* inherits from class *Port*, which inherits from class *Element*).

The *marker* tab shows active syntactic warnings and errors (e.g. channel connects ports with incompatible types), which are created and removed by the *syntax* module mentioned below. Thus, the developer is provided with immediate feedback in case of syntactic inconsistencies, which have to be resolved before simulation/testing.

Finally, the *change* tab shows the history of model modifications in current modeling session. Note that the software tracks all modifications for synchronization between several tool instances as well as engineering process analysis.

*Technology* The prototypical tooling is written in *Java* [4], and uses *JavaFX* for visualization and *JBullet* [5] for collision detection. The implementation consists of a *model* module representing the modeling technique (Hackenberg et al., 2015), a *syntax* module for checking models for syntactic consistency, a *semantics* module for testing models for semantic constraint violations, and a *workbench* module adding the user interface.

## 5. INDUSTRIAL EXAMPLE

Here, we present the model of an exemplary industrial manufacturing system: The pick and place unit as described by Legat et al. (2013). In the following, we first explain the reusable templates of the system model in Section 5.1 before describing the model of the pick and place unit itself in Section 5.2.

### 5.1 Templates

Templates typically include models of the workpieces, which are being processed by the manufacturing system. Furthermore, models of sensors and actuators, such as light barriers, pneumatic and hydraulic cylinders, or electric drives, might be included. Due to space limitations we only present the *workpiece* model.

*Workpiece* As explained later, the pick and place unit is responsible for sorting and optionally stamping workpieces based on their type which we distinguish in the model by their colors, i.e. white, gray, and black. For each type of workpiece a separate, but structurally identical template is provided. The interface of each template includes ports for transferring kinematic and stamp energy as well as for observing the workpiece type and state (i.e. unstamped or stamped). Then, each workpiece template contains one behavior with the states *unstamped* and *stamped* as well as one transition *stamp* shown in Figure 2.

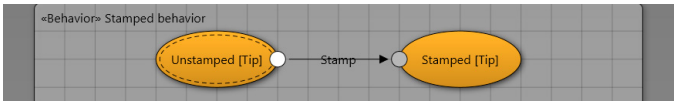
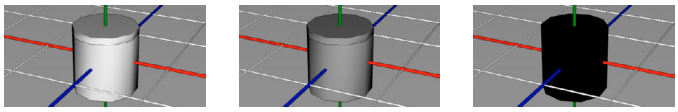



Fig. 2. Workpiece behavior view.

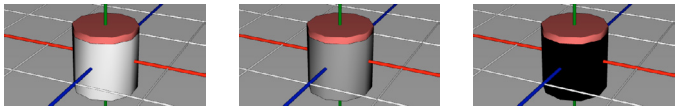
The two states are used for informing the environment about the constant workpiece type and the potentially changing workpiece state. Furthermore, each state contains a separate geometric model of the tip of the workpiece visible in Figure 3. In the *unstamped* state the tip

[4] http://docs.oracle.com/javase/8/index.html
[5] http://jbullet.advel.cz

color corresponds to the type color (i.e. white, gray, or black), while in the *stamped* state the tip color is red. Finally, the workpiece state changes if stamp energy is received from the environment.



(a) White *a* (b) Gray *a* (c) Black *a*



(d) White *b* (e) Gray *b* (f) Black *b*

Fig. 3. Unstamped *a*/stamped *b* workpiece scene view.

Additionally, the workpiece model includes a constant geometric part representing the base of the cylinder already depicted in Figure 3. Again, the color of the base part corresponds to the type color (i.e. white, gray, or black).

### *5.2 Components*

The model of the *pick and place unit* (i.e. the system under development) itself is explained in the following. In particular, the second level of decomposition is included containing a *distributor*, a *stamper* and a *sorter* component. Further levels of decomposition are omitted due to space limitations.

*Pick and place unit* As mentioned previously, the pick and place unit is responsible for sorting and optionally stamping (i.e. labeling) workpieces. Figure 4 shows the geometric model of the system including respective material entry *green* and exit *red* ports. All workpieces are handed over from the environment at the same material entry port. Then, white workpieces have to be stamped and handed over to the environment at the first material exit port. Gray and black workpieces do not have to be stamped and can be handed over to the environment at the second and third material exit ports respectively.

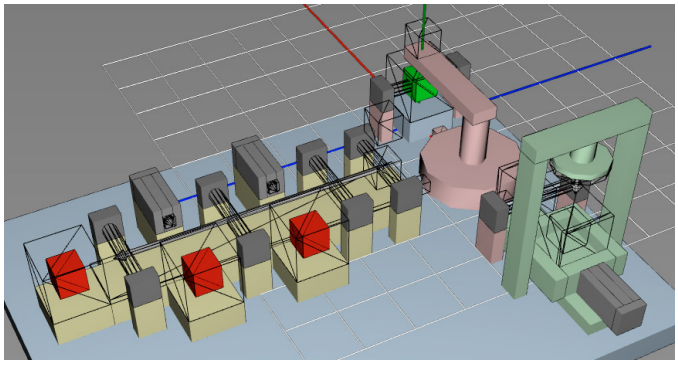


Fig. 4. Pick and place unit scene view (including material entry ports *green* and material exit ports *red* used for scenario specification).

These requirements are translated into three separate, but structurally identical scenarios, out of which one is shown in Figure 5. Each scenario waits for the pick and place unit to send a *ready* signal within 50 time steps. Then, a white, gray, or black workpiece is created at the material entry port location and a *start* signal is sent. Thereafter, the scenario waits for the pick and place unit to send a *finished* signal within 150 time steps. Finally, the scenario tries to remove an unstamped/stamped workpiece of the respective type at the desired material exit port location within one time step. If the removal succeeds, the implementation model of the pick and place unit is consistent with the scenario.

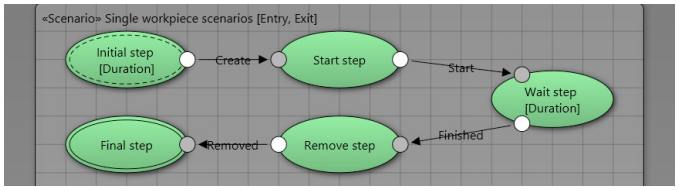



Fig. 5. Pick and place unit scenario view.

From the scenarios one monitor is derived as shown in Figure 6, which captures the (technical) processes for all three types of workpieces and decides upon the *stamp* location (not yet defined in the requirements/scenarios). Starting from the *wait* activity the monitor supports two alternative activity sequences. In case of gray and black workpieces the first activity sequence only includes moving the workpieces to the respective material exit port. In case of white material the second activity sequence includes moving the workpiece to some *stamp* location, stamping the workpiece, and moving the workpiece from the stamp location to the material exit port location.

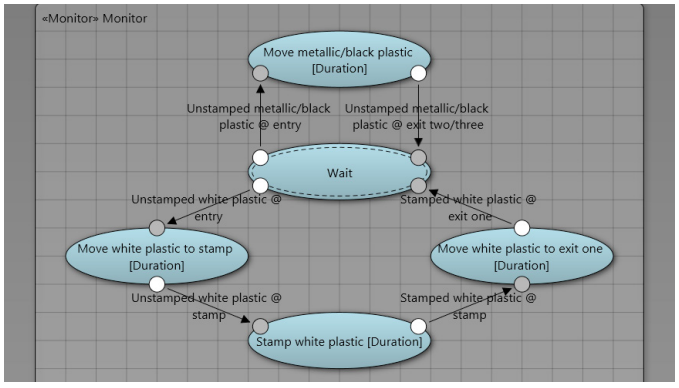



Fig. 6. Pick and place unit monitor view.

Finally, from the scenarios and the monitor a decomposition into three mechatronic components is derived shown in Figure 7. Note that the architecture view also includes material interaction ports *orange* used for scenario and monitor formalization. The mechatronic components are the *distributor*, the *stamper*, and the *sorter*, which are connected by data ports/channels for signaling *ready*, *start*, and *finished*.

The *distributor* (see Figure 8a) is responsible for communicating with the environment, detecting workpiece presence and type autonomously, and moving workpieces between the entry material port, the *stamper* and the *sorter*. To achieve this task, the *distributor* employs workpiece sensors and a crane with a pneumatic gripper. In contrast, the *stamper* (see Figure 8b) is responsible for stamping workpieces received at the *stamp* location. To implement

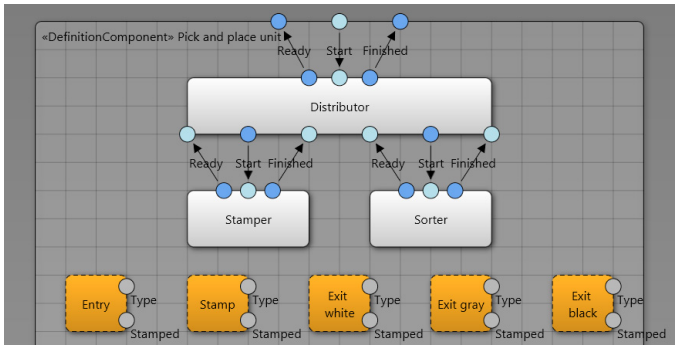
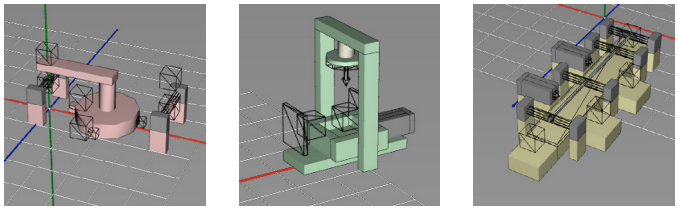



Fig. 7. Pick and place unit architecture view (including material interaction ports *orange* used for scenario and monitor specification).

this task, the *stamper* includes a basket and a stamp with respective pneumatic cylinders. Finally, the *sorter* (see Figure 8c) is responsible for detecting workpiece presence and type autonomously and moving workpieces to their respective exit material ports. To perform this task, the *stamper* contains four material sensors, a conveyor belt, and two pneumatic cylinders. Furthermore, all *distributor*, *stamper*, and *sorter* provide their own software controller.



(a) Distributor (b) Stamper (c) Sorter

Fig. 8. Pick and place unit subcomponent scene views.

The final model of the pick and place unit does not contain any syntactical and semantical inconsistencies. This fact is illustrated also by the test report depicted in Figure 9. The model of the pick and place unit contains twelve scenarios in total. Four scenarios belong to the *distributor*, one belongs to the *stamper*, and three belong to the *sorter*. The remaining four scenarios belong to the pick and place unit itself. The test report shows that eleven out of twelve scenarios terminate successfully (i.e. scenario and system model are consistent), while only one scenario terminates with time-out. However, when increasing the maximum simulation time per scenario the time-out can be resolved.

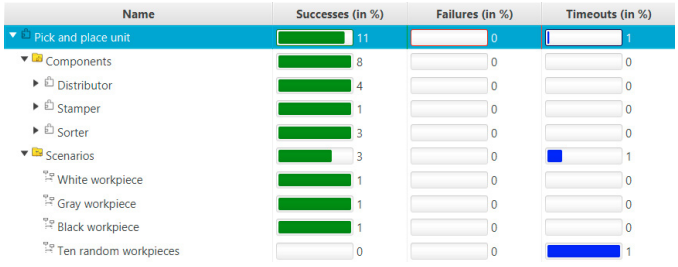
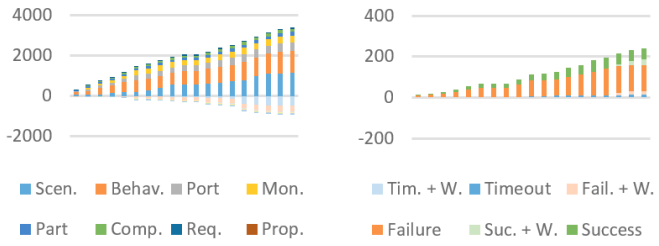



Fig. 9. Pick and place unit management view.

## 6. ENGINEERING WORK FLOW ANALYSIS

To understand the *effectiveness* of our approach for consistent conceptual design of mechatronic manufacturing systems, we provide an analysis of the engineering work flow based on data collected during tool usage. In total, developing a consistent model of the pick and place unit from Section 5 using the prototypical tooling from Section 4 took approximately 19 hours. In the following, we present selected data on the modeling activity and the simulation activity (see Figure 10) as well as the modeling work flow (see Figure 11).




(a) Modeling activity (b) Simulation activity

Fig. 10. Cumulative activity charts over time (in hours).

We describe the modeling activity using the number of model elements created (= 3,397) and deleted (= 903) over time shown in Figure 10a. We distinguish the *component*, *requirement*, *port*, *property*, *scenario*, *monitor*, *behavior*, and *part* element categories. The chart reveals that most elements belong to scenarios (= 817) and behaviors (= 674), which describe the system and environment dynamics. Then, average numbers of elements belong to ports (= 372), monitors (= 265), parts (= 179), and components (= 133), which describe the component interfaces, technical processes, physical geometries, and logical decomposition. Finally, fewest elements belong to requirements (= 54) and properties (= 0), which describe informal customer expectations and formal state invariants. Furthermore, the chart shows that removed elements mainly belong to behaviors (= 489) and scenarios (= 262). Note that these numbers indicate that inconsistencies and other quality issues can be found primarily in the system dynamics. Interestingly, issues can be found also in test cases.

In addition, we describe the simulation/test activity using the number of simulation runs (= 241) and their outcomes over time shown in Figure 10b. We distinguish the outcomes *timeout (+ warning)*, *failure (+ warning)*, *success (+ warning)*. The chart reveals that simulation is used frequently during the development process (≈ 13 simulation runs per hour). Furthermore, all successful, timed-out, and failed simulations with and without warnings can be observed. In total, failed simulations take the largest share (= 147), followed by successful simulations (= 81). In contrast, timed-out simulations take the smallest share (= 13), Note that time-outs can be resolved easily by increasing the maximum simulation duration. The number of failed simulations indicates that inconsistencies are discovered frequently in the model. Finally, the growing number of successful simulations indicates that the inconsistencies are resolved towards the end of the project.

Finally, we describe the cross-disciplinary modeling work flow using the transitions between the element categories shown in Figure 11. The directed graph encodes the transition probabilities using the edge width and color. The graph reveals that in principle transitions between all

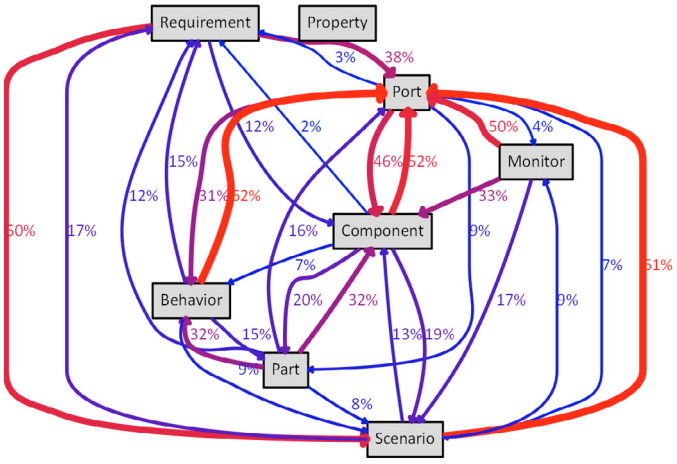



Fig. 11. Modeling work flow graph.

categories are possible. The highest probabilities have been measured for transitions from behaviors to ports (= 62%), from scenarios to ports (= 61%), and from components to ports (= 52%). Instead, the lowest probabilities have been measured for transitions from ports to monitors (= 4%), from ports to requirements (= 3%), and from components to requirements (= 2%). Note that the probabilities indicate that the tool allows one to jump seamlessly between different element categories respectively views onto the manufacturing system. Consequently, one can react quickly on arising (cross-disciplinary) inconsistencies.

Due to brevity we omit a more detailed discussion of the data. Still, we conclude that our tool prototype provides effective means for cross-disciplinary and consistent conceptual design of manufacturing systems. However, we need to determine the degree of effectiveness as soon as approaches with comparable coverage become available.

## 7. CONCLUSION AND FURTHER WORK

In this article, we described flaws in cross-disciplinary modeling and tool support for manufacturing system design: The main difficulty lies in *handling* the *manifold relationships* between the elements of conceptual designs of manufacturing systems, mainly resulting from a *lack* of *phenomena coverage* by *system models* (e.g. physical or spatial effects among plant components, intertwining of plant physics, mechatronics and software-based control) and a *lack* of *precision* of these models (i.e. formal foundation, clear model semantics) in order to run automated analyses to disclose and track such relationships.

Our contribution consists of (i) the use of a cross-disciplinary and precise modeling technique, based on previous work in Hackenberg et al. (2015), paired (ii) with a tool prototype for modeling, simulation and consistency checking. This prototype indicates how we think future tools should support the conceptual design of manufacturing systems. We also provide insight into how the *engineering work flow* (i.e. cross-disciplinary model creation, simulation and evolution) can be tracked by the tool to highlight our *notion* of *seamlessness* and *consistency*.

Next, we want to conduct a modeling experiment or a pilot study, together with manufacturing practitioners, to evaluate tool usability in an industrial cross-disciplinary and collaborative (i.e. multi-user) setting. Insight from such an experiment can help us to evaluate the usefulness of our modeling technique. Furthermore, we are interested in using the work flow data from Section 6 for a modeling recommender system.

## REFERENCES


Botaschanjan, J., Hummel, B., Hensel, T., and Lindworsky, A. (2009). Integrated behavior models for factory automation systems. In *Emerging Technologies Factory Automation – ETFA. IEEE Conf.*, 1–8.

Botaschanjan, J. and Hummel, B. (2010). Material flow abstraction of manufacturing systems. In A. Cavalcanti, D. Deharbe, M.C. Gaudel, and J. Woodcock (eds.), *Theo. Aspects of Comp. – ICTAC*, volume 6255 of *LNCS*, 153–67. Springer.

Broy, M., Huber, F., and Schätz, B. (1999). AutoFocus – Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme. *Informatik Forschung und Entwicklung*, 14(3), 121–34.

Burmester, S., Giese, H., and Tichy, M. (2005). Model-driven development of reconfigurable mechatronic systems with mechatronic uml. In U. Aßmann, M. Aksit, and A. Rensink (eds.), *Model Driven Architecture*, volume 3599 of *LNCS*, 47–61. Springer.

Eigner, M. (2013). Modellbasierte virtuelle produktentwicklung auf einer plattform fr system lifecycle management. In U. Sendler (ed.), *Industrie 4.0*, Xpert.press, 91–110. Springer Berlin Heidelberg.

Friedenthal, S., Moore, A., and Steiner, R. (2014). *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann.

Fritzson, P. (2014). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley & Sons.

Gausemeier, J., Tschirner, C., and Dumitrescu, R. (2013). Der weg zu intelligenten technischen systemen. *GITO Prof. J. Industrie Management*, 29(1).

Gausemeier, J., Dorociak, R., and Nyßen, A. (2010). The mechatronic modeller: A software tool for computer-aided modeling of the principle solution of an advanced mechatronic system. In *11th International Workshop on Research and Education in Mechatronics*.

Hackenberg, G., Campetelli, A., Legat, C., Mund, J., Teufl, S., and Vogel-Heuser, B. (2014a). Formal technical process specification and verification for automated production systems. In D. Amyot, P. Fonseca i Casas, and G. Mussbacher (eds.), *System Analysis and Modeling: Models and Reusability*, volume 8769 of *LNCS*, 287–303. Springer.

Hackenberg, G., Richter, C., and Zäh, M.F. (2014b). A multi-disciplinary modeling technique for requirements management in mechatronic systems engineering. *Elsevier J. Procedia Tech.*, 15(0), 5–16. IC System-Integrated Intelligence: Challenges for Product and Production Eng.

Hackenberg, G., Richter, C., and Zäh, M.F. (2015). Imomesa – abschlussbericht. Technical report, Technische Universität München.

Henzinger, T.A. (1996). The theory of hybrid automata. In *Proc. 11th Annual IEEE Symp Logic Comp Sci*, LICS '96, 278–. IEEE Computer Society, Washington, DC, USA.

Legat, C., Folmer, J., and Vogel-Heuser, B. (2013). Evolution in industrial plant automation: A case study. In *Ann. Conf. IEEE Ind. Electronics Society*, IECON'13, 4386–91.

Rehage, G., Bauer, F., and Gausemeier, J. (2014). Specification technique for the consistent description of manufacturing operations and resources. In M.F. Zäh (ed.), *Enabling Manufacturing Competitiveness and Economic Sustainability*, 47–53. Springer.

Reinhart, G. and Wünsch, G. (2007). Economic application of virtual commissioning to mechatronic production systems. *Springer J. Prod. Eng. - Research and Devel.*, 1(4), 371–9.

Teufl, S., Mou, D., and Ratiu, D. (2013). Mira: A tooling-framework to experiment with model-based requirements engineering. In *Requirements Eng. Conf. (RE), 2013 21st IEEE Int.*, 330–1.