# Cyber-physical manufacturing system Development: A test-driven design method and exploratory case study

Georg Hackenberg, Jakob Mund

TUM-I1664

# Cyber-physical manufacturing system development: A test-driven design method and exploratory case study

Georg Hackenberg[1,a] and Jakob Mund[1]

[1]*Technische Universität München, Fakultät für Informatik, 85748 Garching bei München, Germany*

**Abstract.** Today, manufacturing engineering companies still struggle developing *cyber-physical* solutions. Studies suggest that inappropriate engineering methods dominated by mechanical design decisions and incompatible engineering tool landscapes are key problems that lead to isolated and potentially inconsistent specifications. To overcome the current situation, in this paper we propose a *test-driven* approach to cyber-physical manufacturing system design based on an integrated modeling technique and engineering tool called MACON. Then, we explore the applicability and potential benefits of the approach for an industry-close example. Therefore, we analyze empirical data about the development process collected during tool usage. The data indicates that test-driven development is applicable in principle to the cyber-physical manufacturing systems domain resulting both in early test specification at various levels of the component architecture as well as in early identification of design flaws.

## 1 Introduction

The transition from purely mechanical to cyber-physical manufacturing systems is proceeding for a few decades now. In the last decade Reinhart and Wünsch [1] observed that many manufacturing engineering companies are struggling with high commissioning cost due to design flaws, which remain undiscovered until late in the projects. At the same time, Schäfer and Wehrheim [2] suggest that existing discipline-specific modeling techniques, analysis tools, and engineering methods are not sufficient to cope with the multidisciplinary nature and complexity of the engineering problems. Similar thoughts have been expressed by other authors as well, e.g. [3]. Consequently, researchers have been working on extending and integrating existing approaches to tackle the described challenges.

### 1.1 Related work

The first attempts originate from the *Product Design* (PD) community. For example, Umeda et al. [4] propose the *Function, Behavior, Structure* approach, where first the product functionality is defined before deriving the expected behavior as well as the mechanical structure and the actual behavior. Similarly, Suh [5] proposes the *Axiomatic Design* approach, where first the functional requirements are defined before driving design parameters and process variables, while their relationship is expressed using matrices. More recently, Sitte and Winzer [6] propose the *Demand-Compliant Design* approach, where first a requirement structure is defined before deriving a function structure, a process structure, and a component structure, while the relationships are expressed again using matrices. An advantage of the PD approaches is their general applicability as well as the systematic mapping from requirement to solution characteristics. However, the design information typically is not captured in a semantically meaningful way preventing advanced analyses.

Further attempts originate from the *System Design* (SD) community. For example, Burmester et al. [7] propose the *MechatronicUML* approach, where mechatronic systems can be described in terms of components, ports, channels and (hybrid) state machines. More recently, Friedenthal et al. [8] describe the *Systems Modeling Language* (SysML) approach, where also the customer needs and technical constraints are considered and additional perspectives on the system behavior are provided. Based on SysML, Thramboulidis [9] proposes the *Model-Integrated Mechatronics* approach, where mechatronic systems are described from a control application, a communication and processing system, and a mechanical process perspective. Finally, Gausemeier et al. [10] propose the *CONSENS* approach, where also the system geometry can be considered. An advantage of the SD approaches is their wide-spread use and tool support. However, the approaches typically lack either formal semantics for advanced analyses or important design information such as geometry.

To compensate for the above-mentioned problems, additional attempts originate from the *Formal Method* (FM) community. For example, Hummel [11] proposes the *Spatio-Temporal Engineering Models* (STEM) approach, where mechatronic systems can be described in terms of components, ports, channels, and (hybrid) state

[a] Corresponding author: hackenbe@in.tum.de

machines as well as mechanical parts, material detectors, material entries, material exits, and collision-based interaction. Based on STEM, Hackenberg et al. [12, 13, 14] propose an extended approach, where additionally customer needs, technical constraints, and manufacturing processes can be considered. An advantage of the FM approaches is that they capture important design information and enable advanced analyses through formal model syntax and semantics. However, the approaches still lack a practical methodology, which promises to develop higher quality specifications in shorter time.

### 1.2 Problem statement

Despite the progress explained previously, developing high-quality specifications for complex cyber-physical manufacturing systems in early phases of engineering projects still remains a major challenge. Our brief literature review suggests that existing approaches lack either (1) important design information such as system geometry and dynamic interaction based on spatial arrangement, or (2) formal semantics for advanced specification analyses such as model checking [15], or (3) practical methodologies that are able to cope with the complexity and multidisciplinary nature of respective engineering problems. We believe that all three problems have to be solved in combination before high-quality specifications can be developed efficiently.

### 1.3 Contribution

To overcome the present situation, in Section 2 we describe a test-driven methodology for cyber-physical manufacturing system design based on existing FM techniques, which already capture a wide range of design information and come with formal semantics for advanced analyses. In particular, we try to map ideas and principles behind test-driven (software) development [16] to the cyber-physical manufacturing system domain, which include (1) specifying test cases first and (2) developing the system in increments. Then, in Section 3 we describe a case study that has been carried out to explore the applicability and benefits of the approach. Finally, in Section 4 we conclude the presented work and outline future research.

## 2 Test-driven design method

According to test-driven development principles [16], our test-driven design method promotes a fundamentally iterative and incremental approach to complex cyber-physical manufacturing systems development. Hereby, *iterations* are used to revise inaccurate problem understanding as well as incompatible or inadequate design decisions with respect to the current problem understanding. *Increments*, on the other hand, are used to reduce the problem scope considered in each iteration. Consequently, the entire system does not have to be designed at once, but the engineers can concentrate on (selected) parts of the engineering problem only. Increments are one key principle to cope with problem complexity. Furthermore, increments enable early stakeholder integration into the development process and, hence, increase the validity of the problem understanding developed and the design decisions taken throughout the process. While iterative approaches have been suggested for mechatronic systems already in the 1990s, e.g. [4, 5], the idea of increments appears to be novel to us in the cyber-physical manufacturing systems domain.

Figure 1 provides an overview of our test-driven design method in UML activity diagram notation [17]. According to established manufacturing engineering practices, the method distinguishes between a preparation and an implementation phase. In the preparation phase the requirements are specified including input and output flows, an efficient manufacturing process is design, and relevant test procedures are determined. In contrast, in the implementation phase the system architecture is defined including the system components and their interactions based on material, energy, and data flow [18]. In the following, we describe both phases in more detail. Thereby, we explain the relation between the activities and the elements of the MAnufacturing CONception (MACON) modeling technique [12, 13, 11].

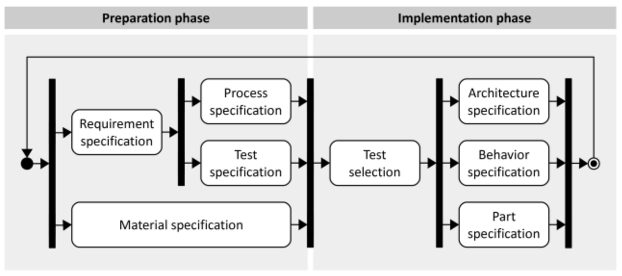

**Figure 1.** Illustration of the test-driven design method divided into a preparation and an implementation phase.

### 2.1 Preparation phase

The preparation phase is divided into four activities, namely requirement specification, material specification, process specification, and test specification. Furthermore, the corresponding elements of the MACON modeling technique [12, 13] are depicted in Figure 2. The elements include (informal / textual) requirements, (material / energy / data) ports, constraints, (requirement / process) monitors, and (test) scenarios.

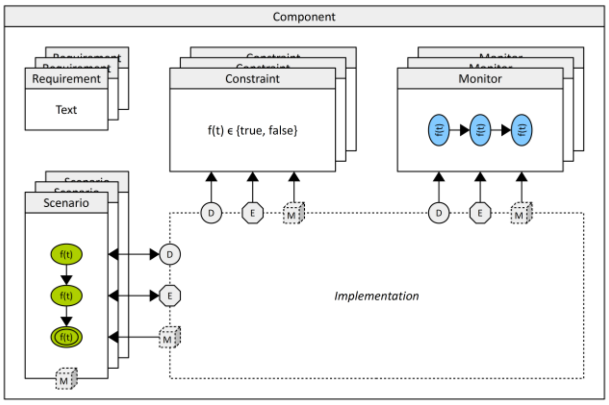

**Figure 2.** Illustration of the MACON modeling elements for the preparation phase [12, 13].

During **requirement specification** the customer needs are collected and documented using (informal / textual) *requirement* elements. Then, the requirements can be formalized using *constraint* and *monitor* elements over *port* elements. According to Liang and Paredis [18], ports define the data, energy, and material interface of the cyber-physical system and its components. Then, constraints allow one to express invariants for the entire lifetime of the system such as the maximum amount of energy flow per time unit over some energy port. In contrast, monitors allow one to express invariants for certain activities only such as the maximum amount of energy for material transportation or material shape manipulation. Note that the formal semantics of monitors is described in [12].

Then, during **process specification** the manufacturing process is derived - again - using *monitor* elements. Similar to established process planning tools [19], the *activity* elements represent the process steps such as grinding or milling and the *transition* elements describe the possible activity sequences (e.g. first mill, then grind). Then, invariants can be attached to the activities and guards must be defined for the transitions. The invariants describe constraints that must hold while performing the process step such as the maximum allowable duration. In contrast, the guards describe the conditions that must hold to finish a process step such as the target material shape after milling and grinding respectively. Note that monitors can be used to formalize (informal / textual) requirements also. However, during process specification design decisions can be added regarding intermediate process steps, which have not been prescribed by the customer.

In contrast, during **test specification** the test cases are derived using *scenario* elements. Scenarios essentially describe environments, within which the system is expected to operate. Scenarios are divided into *entry*, *exit*, *step*, and *transition* elements. Entries define the position and orientation in space where the environment may add material to the simulation during test execution. In contrast, exits represent the locations in space where the environment may remove material from the simulation during test execution. Finally, steps and transitions model the environment behavior, which can read component output ports and write component input ports as well as scenario entry and exit ports. Again, invariants can be added to steps such as the maximum allowable duration, and guards have to be specified for transitions such as the presence of material at a particular location. Finally, a test case is passed if the final step is reached during test execution without violating any invariants within considered constraints, monitors, and the scenario itself. Note that the formal semantics of scenarios is similar to test automata [20].

Finally, during **material specification** the material flowing through the manufacturing system is modeled using separate *component* elements. In parallel to requirement specification typically initial and final material states are considered only. Then, in parallel to process specification typically intermediate material states as well as transitions and interaction ports are added. Hereby, the states, transitions, and ports are derived from the respective process steps. Note that for each material component the test-driven engineering method can be applied recursively. Consequently, cyber-physical "material" can be considered in principle as well, which might be equipped with appropriate sensing, actuating, and digital information processing units such as RFID [21].

## 2.2 Implementation phase

Similar to the preparation phase, the implementation phase is divided into four activities, namely test selection, architecture specification, behavior specification, and part specification. For reference, the corresponding elements of the MACON modeling technique [11] are depicted in Figure 3. The elements include components, material ports respectively slots, channels, behaviors, and parts.

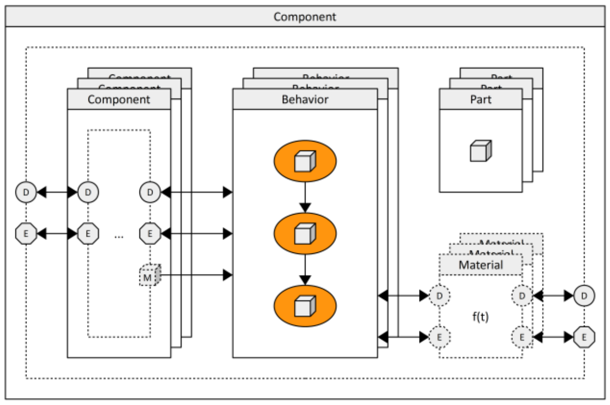

**Figure 3.** Illustration of the MACON modeling elements for the implementation phase [11].

During **test selection** the test cases, that are considered within the current iteration or increment, are selected defining the partial engineering problem to be solved (i.e. passing the selected test cases only). Hereby, one can select an arbitrary non-empty subset of test cases from the set of all possible test cases. The selected test cases drive the following specification activities (hence the name *test-driven*). Different strategies can be used for test selection. For example, one could select the test cases based on estimated / perceived architectural impact such that important design decisions are likely to be in early iterations or increments. Alternatively, one could select the test cases based on suspected ambiguity / misinterpretation to foster requirement validation based on early customer feedback. At this point, we leave the question of suitable test selection strategies as well as their impact on the development process and the product quality to future research.

Then, during **architecture specification** the modular structure of the cyber-physical manufacturing system is defined using *component*, *material slot* and *channel* elements. Herein, components represent cyber-physical subproblems / subsystems, which allows one to reduce the considered problem / system complexity further. Note that for each component the test-driven engineering method can be applied recursively including both the preparation and the implementation phase as well as the fundamentally iterative and incremental process. In contrast, material slots are derived from material ports directly and allow one to bind colliding cyber-physical components dynamically during test execution such as material added at some entry (see Section 2.1). The formal semantics of the dynamic interaction based on collision is described in [11]. Finally, channels define the interaction between the components of the architecture as well as components dynamically bound at material ports respectively slots.

Subsequently, during **behavior specification** the reactions of the component respectively system to input from its environment are defined using *behavior* elements. Behaviors are divided into *state* and *transition* elements. Both states and transitions may contain actions for writing output ports of the surrounding component respectively system such as some data signal or energy flow. Following [22] energy flow is modeled using scalars, where the sign indicates the direction of the energy flow. Hereby, a negative sign indicates that the component requires the energy from the environment, a positive sign indicates that the component provides the energy to the environment instead. In contrast, transitions may contain guards over the component input ports. The guards define the conditions for switching between the source and the target state of the transition such as receiving a data signal from the environment or observing an energy flow. Finally, each state may contain individual *part* elements. Parts allow one to model the mechanical shape of cyber-physical components. Consequently, a component reaction might include changing its shape, e.g., due milling or grinding energy obtained from the environment. Note that the formal semantics is based in input / output automata [23].

Finally, during **part specification** the static portion of the mechanical shape of a component is defined again using *part* elements. Each part contains exactly one *volume* element. Hereby, we distinguish between atomic volumes and composite volumes. Atomic volumes represent basic shapes such as spheres, cylinders, and boxes. In contrast, composite volumes represent the union of child volumes, where child volumes can be both atomic volumes and composite volumes. Consequently, we rely on a subset of constructive solid geometry (CSG) [24] excluding volume intersection and difference operators rather than using the full set of CSG operators or using boundary representations such as non-uniform rational basis splines [25]. The current reason is that the collision-based dynamic interaction semantics [11] is more easy to implement using the selected CSG subset. In the future we might extend our approach to other, more powerful representations.

## 3 Exploratory case study

For demonstrating and evaluating the test-driven design method (see Section 2) we re-designed a miniaturized cyber-physical manufacturing system - the so-called pick-and-place unit - which is located at the Institute for Automation and Information Systems, Technische Universität München [26, 27]. Subsequently, we first explain the objectives of the study in Section 3.1 as well as the tool support used during the study in Section 3.2. Then, we explain the method of data collection in Section 3.3 and the system design obtained while performing the study in Section 3.4. Finally, we evaluate the data collected during tool usage with respect to the study objectives in Section 3.5 before discussing threads to the internal and external validity of the study in Section 3.6.

### 3.1 Study objective

The main objective of this study is to evaluate the general *applicability* of the test-driven design method to cyber-physical manufacturing systems. Consequently, we want to demonstrate the successful use of the method at least for one selected and representative case (i.e. the pick-and-place-unit). In this context, we consider the experiment to be successful, if we can follow the prescribed method closely (called *process feasibility*), while obtaining a valid system design (called *result validity*). Note that here we do not consider the performance, with which the system design can be obtained. Consequently, we will not derive any statement about the efficiency of the method with respect to other design methods (e.g. test-driven versus classical top-down).

### 3.2 Tool support

A prototypical tool support for the MACON modeling technique has been described in [14]. Note that herein the challenge lies in integrating the different views onto the cyber-physical manufacturing system intuitively. Here, we describe the prototypical tool only briefly. The graphical user interface of the tool consists of two screens, namely (1) a modeling screen and (2) a testing screen, which are explained both in the following.

As the name suggests, the *modeling* screen (Figure 4) allows one to edit the design of the cyber-physical manufacturing system. Change events are triggered when editing attribute values of, adding / removing children to / from, and adding / removing references between model elements. Furthermore, after each change event syntactic rules are re-evaluated and issues appear or disappear (e.g. missing child). Also, the syntactic issue appearance and disappearance events are recorded for analysis. Finally, test execution can be started from within the modeling screen for uncovering the semantic issues in the model. Upon the test execution start event the testing screen shows up.

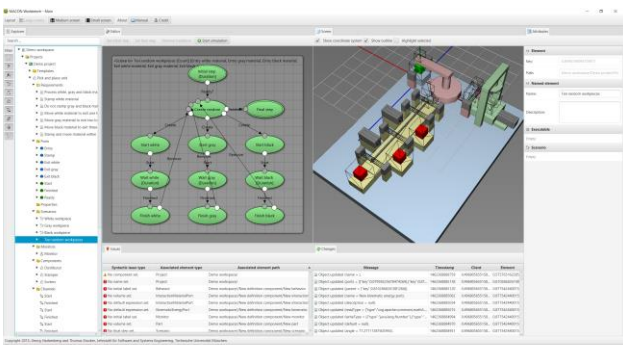
**Figure 4.** Modeling screen of the prototypical tool support [14].

In contrast, the *testing* screen (Figure 5) executes the simulation engine in the background and allows one to inspect the simulation results. During test execution the engine can raise semantic issues, which are displayed in the testing screen. Furthermore, the simulation engine triggers a simulation stop event in case a severe semantic issue was raised (i.e. the test execution *failed*, e.g., due to the violation of an invariant as explained in Section 2.1), the scenario final step was reached (i.e. the test execution *succeeded*), or a timeout issue appeared (i.e. the test execution could not be finished within a predefined time frame). Finally, the testing screen allows one to step through the simulation and inspect the system state. The system state includes the component translations and orientations in space, the port valuations, the current scenario step, the current monitor activities, and the current behavior states.

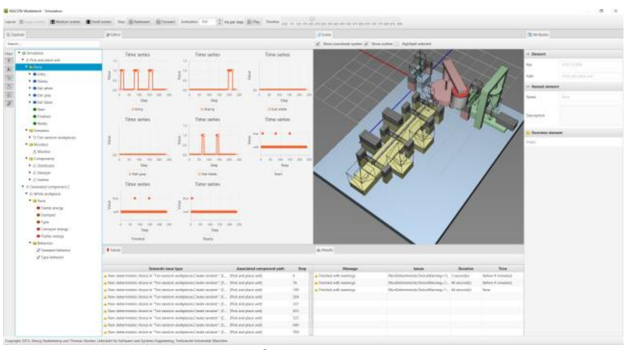
**Figure 5.** Testing screen of the prototypical tool support [14].

## 3.3 Data collection

For evaluating the *process feasibility*, we instrumented the prototypical tool (see Section 3.2) to track all modeling and testing activities. Specifically, the tool records each model change event (i.e. *adding / removing children* of, *adding / removing references* between, and *modifying attributes* of model elements) and associates it with the affected component of the system architecture. In addition, the tool records each test execution event and associates it with the respective scenario and execution result (i.e. *success*, *failure*, or *timeout*). From the records we reconstruct the development process and compare the reconstruction with the method described in Section 2.

In contrast, for evaluating the *design validity*, we can use the developed system design directly. In particular, we can compare the developed system architecture (i.e. the components and their interactions) with the SysML documentation [27]. Furthermore, we can compare the developed system behavior (i.e. the states and the transitions) with the behavior of the physical system, which we are provided access to. Consequently, we can derive the design validity from the similarity between the developed and the documented system architecture as well as the developed and the physical system behavior. Herein, the developed behavior might be simplified.

## 3.4 System design

The developed system design of the pick-and-place unit consists of the top-level system component itself as well as three second-level components, namely the distributor, the stamper, and the separator. In the following we explain each of these components in more detail. Herein, we focus on requirements and scenarios respectively. In contrast, we omit information about the manufacturing processes and the implementation.

### *3.4.1 Pick and place unit*

The *pick-and-place unit* is responsible for receiving white plastic, metallic, and black plastic material at a pre-defined entry location. Then, white plastic material must be stamped due to, e.g., contamination and moved to a pre-defined exit location. In contrast, metallic and black plastic material must be moved to different pre-defined exit locations directly. Furthermore, each material must be processed within a certain duration. Consequently, the pick and place unit is responsible for separating (or sorting) and conditionally stamping material, depending on its type (i.e. white plastic, metallic, or black plastic). After requirement and process specification, one scenario is derived for each material type during test specification. The three scenarios are illustrated in Figure 6, in which green and red boxes represent entry or exit locations. Note that the mechanical parts are not part of the model originally, but they have been added after many iterations and increments.

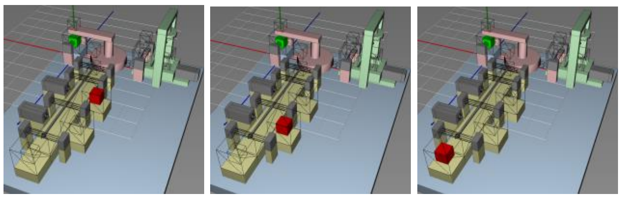
**Figure 6.** Three scenarios of the pick and place unit component.

### *3.4.2 Distributor*

The *distributor* component is responsible for moving white plastic, metallic, and black plastic material between the entry location, the stamper, and the separator. In particular, white plastic material must be moved from the entry location to the stamper location and from the stamper location to the separator location. In contrast, metallic and black plastic material must be moved from the entry location to the separator location directly. Furthermore, each operation must be performed within a certain duration. Again, after requirement and process specification, one scenario is derived for each combination of material type and start location during test specification. Three out of four scenarios are shown in Figure 7. For the sake of brevity, the presentation of a further decomposition as done in the implementation phase (Section 2.2) is omitted here.

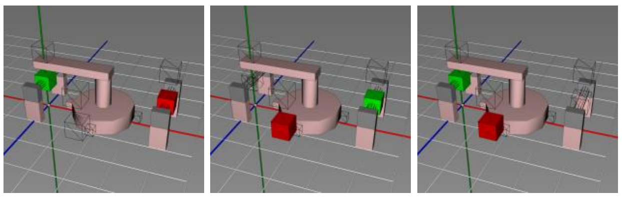
**Figure 7.** Three scenarios of the distributor component.

### *3.4.3 Stamper*

Then, the *stamper* component is responsible for receiving white plastic material at the stamper location, stamping the material (i.e. transmitting "stamp" energy through collision-based dynamic interaction), and delivering the stamped material back at the original location. In contrast, metallic and black plastic material does not have to be considered by the stamper component. Furthermore, the stamping procedure must be performed within a certain duration. Consequently, one scenario is derived during test specification, which is shown in Figure 8. Note that for this component, the entry and exit locations coincide such that only the exit is shown.

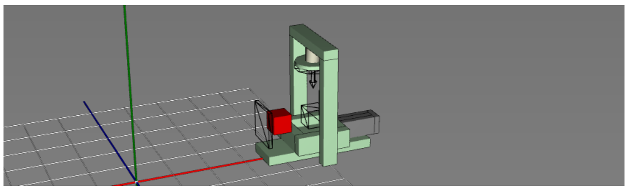
**Figure 8.** One scenario of the stamper component.

### *3.4.4 Separator*

Finally, the *separator* component is responsible for receiving white plastic, metallic, and black plastic material at the separator location. Then, white plastic material must be moved to the first exit location. In contrast, metallic material must be moved to the second and black plastic material to the third exit location. Note that the three exit locations correspond to the exit locations defined during requirement specification for the pick-and-place-unit. Furthermore, analogous to the top-level pick-and-place-unit component and the other lower-level components each operation must be performed within a certain duration. Again, after requirement and process specification, one scenario is derived for each material type, resulting in three scenarios as illustrated in Figure 9. Note that the implementation uses a conveyor belt with several position sensors and push cylinders.

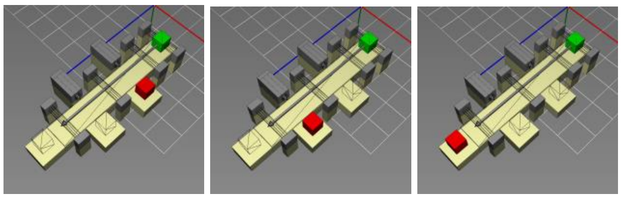
**Figure 9.** Three scenarios of the separator component.

## 3.5 Data evaluation

In total, the experiment comprised 49 tool sessions and 18.44 hours of tool usage. During those sessions, 3,397 model elements were created, 145,785 element attribute modifications were recorded (note that each key stroke is recorded as a separate modification event), and 903 elements were deleted (suggesting the revision of design decisions and / or the current problem understanding). Furthermore, 241 scenario test execution events have occurred and 2,592 syntactic as well as 180 semantic issues have been discovered. Note that these numbers represent absolute performance measures. If such numbers could be obtained for other methods and / or modeling techniques / tools, also relative performance measures could be derived. However, we leave this evaluation to future research.

In the following, we analyze the collected data (see Section 3.3) with respect to the *process feasibility* before discussing the obtained system design (see Section 3.4) with respect to the *design validity*.

### *3.5.1 Process feasibility*

To answer the question of *process feasibility*, we aggregate and visualize the data collected during tool usage. In particular, we analyze the model change events and the test execution events independently.

Figure 9 shows the model change events over time assigned to the components of the system design (i.e. pick-and-place-unit, distributor, stamper, separator; see Section 3.4) and classified by the elements of the MACON modeling technique (i.e. requirements, ports, scenarios, monitors, components, behaviors, and parts; see Section 2). The diagram shows that for each component of the design indeed the preparation (shades of red) and the implementation (shades of blue) phases can be distinguished. In particular, the scenarios (or test cases; orange) are specified before working on the implementation, which represents a core principle of test-driven development [16]. Furthermore, the diagram shows that the process indeed proceeds in iterations because work on the components, behaviors and parts (i.e. implementation phase elements) might be followed by work on the requirements, ports, scenarios, and monitors (i.e. preparation phase elements).

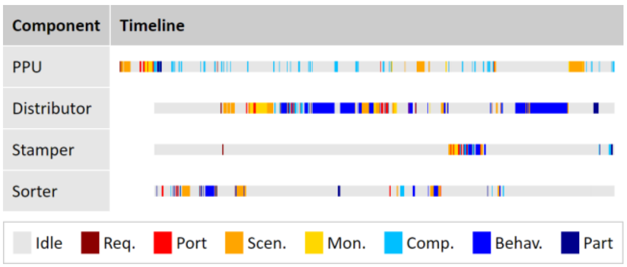

**Figure 9.** Element creation, modification, and deletion events associated with the individual components over time.

The previous diagram allows us to reconstruct the order, in which the activities of the test-driven method (see Section 2) have been executed, from the order, in which the elements of the MACON modeling technique have been touched. However, the diagram does not tell us about the test selection practices and, thus, whether the system has been developed incrementally. To answer this question, Figure 10 shows the test execution events over time assigned to the components and scenarios of the system design and classified by the result of the test execution. The diagram shows that first the sorter component, then the distributor component, and finally the stamper component is finished before the pick-and-place-unit is completed. Furthermore, first the metallic and black plastic scenarios of the pick-and-place-unit are passed before passing the white plastic material scenario.

The reason is that the stamper component is required to pass the white plastic material scenario. However, the stamper component is implemented last. Consequently, passing the white plastic material scenario is delayed for about two hours. From this delay we conclude that a first increment only considered the metallic and the black plastic scenarios. A second increment included the white plastic scenario and, hence, the stamper component. Consequently, we conclude that indeed an incremental approach was applied.

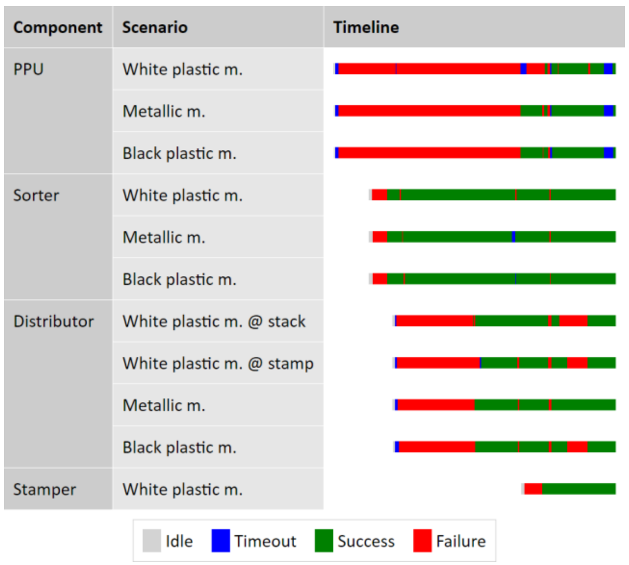

**Figure 10.** Test execution events and results associated with the individual components and scenarios over time.

### 3.5.2 Design validity

To answer the question of *design validity*, we compare the obtained system architecture with the architecture from the SysML documentation of the pick-and-place-unit [27]. Furthermore, we compare the developed system behavior to the behavior of the real physical system.

Fundamentally, both the SysML documentation as well as the developed system design decompose the pick-and-place-unit into three modules, i.e. the sorter, the stamper, and the separator. The distributor is modelled almost identically in the SysML documentation and in our system design, though our system design assigns two material sensors to the sorter component, while the SysML documentation assumes that the sensor signals are provided from the environment. Consequently, we have chosen a more autonomous module design, which we do not consider a critical deviation of the two system architectures. Then, in the SysML documentation the stamper includes only two cylinders and an operator panel component. In contrast, the developed system design omits the operator panel, but includes a stamp head component as well as the interactions between the stamp head, the cylinders and the material. Consequently, the realization of the stamp operation becomes more obvious. Finally, the separator is modeled almost identically in the SysML documentation and the developed system design. Again, the SysML document-tation includes an operator panel, which is omitted in our system design. Also, the SysML documentation distinguished different sensor types (i.e. presence sensor for detecting general material presence, inductive sensors for distinguishing metal, and optical sensors for distinguishing white and black plastic), while we combined all three functions into one sensor component. Note that we could have split the single sensor component into three independent sensor complements easily. However, we consider the single component to be a valid simplification during conceptual design. Finally, from this comparison we conclude that we developed a valid system architecture. Also, we were able to describe certain aspects of the interaction (i.e. the stamping operation) more detailed than in the SysML documentation.

Regarding the system behavior, we have observed the same causalities in the physical system and the developed system design. By causalities we mean the order in which sensor measurements are received, actuator values are controlled, and the system states evolve. The most important state variables include the angular and vertical position of the distributor crane, the position of the stamper cylinders, the position of the separator cylinders, and the position of material being processed by the pick-and-place-unit. One can observe the causalities easily when watching the physical system in operation and the developed system design during simulation. Furthermore, we have observed similar critical states in the physical system and the developed system design such as misplacement of material by the dispatcher due to angular crane position sensor delays. On the other hand, when comparing the behaviors one also can see easily that the timing behaviors do not match. However, we did not consider timing behavior to be critical in this experiment. Furthermore, we have observed critical states in the physical behavior, which could not be observed in the developed system design such as misplacement of material due to spring “push-back” forces of contact sensors. Such aspects of the physical behavior have to be modeled explicitly to be able to observe relate effects. However, we considered these effects to be irrelevant during the experiment. Note that such judgement is left to the expertise of the engineers because the method (see Section 2) does not provide any guideline at the moment. Hence, one must consider whether the effects have a considerable impact on the system design, e.g. because respective countermeasures have to be included, which might take a certain amount of space that is not available or change the behavioral causalities significantly. Still, in summary we conclude that we have obtained a valid system behavior neglecting minor physical effects.

## 3.6 Study validity

According to common practice in experimental research [28], we distinguish between the internal and the external validity of the study. Thus, the *internal validity* is concerned with the reliability of the conclusions that we have drawn about the research questions (i.e. *process feasibility* and *design validity*) from the collected data. On the other hand, the *external validity* is concerned with the degree to which the conclusions can be generalized to the entire cyber-physical manufacturing systems domain.

### 3.6.1 Internal validity

To address the *internal validity* with respect to the *process feasibility*, we (1) used tool instrumentation for collecting data automatically during tool usage, (2) developed appropriate aggregations and visualizations of the collected data, and (3) interpreted the results of these visualizations. Note that we tried to achieve a high degree of automation in the process to remove the bias of subjective interpretation. However, one can question whether the right data has been collected, whether the aggregations and visualizations are appropriate, and whether the interpretation of the visualizations is valid. For data collection we used only the most basic events, which can be obtained from tool usage. In particular, the events cover all possible changes to the system design as well as all possible outcomes of test execution. Then, the aggregation by component of the system design and by element of the MACON modeling technique / result of the test execution was performed automatically as well. Again, the aggregation can be performed unambiguously. The same holds for the visualization using time lines. Finally, the interpretation included mapping the time line information back to the activities and phases of the test-driven design method (see Section 2). In particular, the mapping to phases can be achieved unambiguously, because the phases do not share any model elements. Some ambiguity can be found in the mapping to the activities, as the requirement specification (respectively formalization) and the process specification potentially share the monitor elements. However, we can foreclose this ambiguity because we omitted the formalization of requirements in the experiment.

To address the *internal validity* with respect to the *design validity*, the authors have had access to the SysML documentation [27] as well as the real physical system. Again, the comparison of the system architectures is rather straight forward. In SysML, the architecture is specified using block diagrams including blocks (which correspond to the MACON components) and containment relationships (which are also defined in the MACON modeling technique). In contrast, the interactions between the components are defined only implicitly as part of the textual description and the behavior diagrams in the SysML documentation, while in the developed design interactions are defined explicitly using channels. The comparison of the system behaviors, on the other hand, is more difficult to achieve; we tried to focus the attention on the causalities and the observable critical states. Furthermore, we are aware of the simplifications that have been made with respect to the real physical behavior (e.g. the slight effect of spring forces in contact sensor). As stated previously, we consider such simplifications to be valid during the conceptual design phase and we think that such effects can be modeled with sufficiently large effort. Also, we have observed and addressed critical system states in the developed design, which can be observed in the physical system as well (e.g. the effect of sensor and actuator signal delays on the correct positioning of components).

Finally, a general deficiency of the study with respect to *internal validity* is that the target design existed prior to executing the study. Hence, the developed design is influenced by a priori knowledge. To circumvent this deficiency an experimental setup is required, where the participants are not aware of the target design.

### 3.6.1 External validity

In contrast, the *external validity* of the study is limited mainly due to the academic case (i.e. the pick-and-place-unit). Originally, the case has been designed to resemble industrial plants closely [26]. Consequently, the case comprises a number of important features that can be found in industrial systems. Such features include the transportation, selective manipulation, and separation of different types of material within certain time constraints. However, the case mainly lacks functional, structural, and behavioral complexity. In fact, the individual functions to be performed by the pick-and-place-unit (i.e. processing white plastic, metallic, and black plastic material) can be separated rather easily and have limited influence on each other. It will be interesting to see how the test-driven approach performs on systems, where the different functions cannot be separated that easily. For example, one could think about a system where multiple materials can be processed in parallel to increase productivity. We leave such investigations to future research.

## 4 Conclusions

In this paper we have presented an adaptation of test-driven software development principles to the cyber-physical manufacturing systems domain. In particular, we have integrated additional activities such as input / output material, manufacturing process, and mechanical part specification, which are not relevant for pure software systems. Then, we conducted a first exploratory study which indicated the general applicability of the test-driven approach. In particular, the study demonstrates that (1) it is feasible to specify test cases for cyber-physical manufacturing systems first and (2) the system indeed can be developed incrementally.

Therefore, we see ourselves encouraged to conduct future research on test-driven and agile development methodologies for the cyber-physical manufacturing systems domain. In particular, we plan to explore the importance of syntactic and semantic issues and their impact on system development more deeply. Then, we plan to support the different specification and selection activities with more advanced decision support. Finally, we plan to extend our experiments to more complex systems as well as larger and more heterogeneous groups of engineers and practitioners. Hereby, we are interested also in performing comparative studies with respect to other design methodologies and modeling techniques.

## Acknowledgements

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design for Future – Managed Software Evolution.

## References

[1] G. Reinhart and G. Wünsch. Economic application of virtual commissioning to mechatronic production systems. *Springer Journal of Production Engineering - Research and Development*, 1(4):371–379, 2007.

[2] W. Schäfer and H. Wehrheim. The challenges of building advanced mechatronic systems. In *2007 Future of Software Engineering*, FOSE ’07, pages 72–84, Washington, DC, USA, 2007. IEEE Computer Society.

[3] R. Isermann. On the design and control of mechatronic systems-a survey. *IEEE Transactions on Industrial Electronics*, 43(1):4–15, Feb 1996.

[4] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama. Supporting conceptual design based on the function-behavior-state modeler. *Ai Edam*, 10(4):275–288, 1996.

[5] P. N. Suh. Axiomatic design theory for systems. *Research in Engineering Design*, 10(4):189–209, 1998.

[6] J. Sitte and P. Winzer. Demand-compliant design. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(3):434–448, May 2011.

[7] S. Burmester, H. Giese, and M. Tichy. *Model Driven Architecture: European MDA Workshops: Foundations and Applications, MDAFA 2003 and MDAFA 2004, Twente, The Netherlands, June 26-27, 2003 and Linköping, Sweden, June 10-11, 2004. Revised Selected Papers*, chapter Model-Driven Development of Reconfigurable Mechatronic Systems with Mechatronic UML, pages 47–61. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

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

[9] K. Thramboulidis. The 3+1 sysml view-model in model integrated mechatronics. *Journal of Software Engineering and Applications*, 3(02):109, 2010.

[10] J. Gausemeier, U. Frank, J. Donoth, and S. Kahl. Specification technique for the description of self-optimizing mechatronic systems. *Research in Engineering Design*, 20(4):201–223, 2009.

[11] B. Hummel. A semantic model for computer-based spatio-temporal systems. In *Engineering of Computer Based Systems, 2009. ECBS 2009. 16th Annual IEEE International Conference and Workshop on the*, pages 156–165, April 2009.

[12] G. Hackenberg, A. Campetelli, C. Legat, J. Mund, S. Teufl, and B. Vogel-Heuser. *System Analysis and Modeling: Models and Reusability: 8th International Conference, SAM 2014, Valencia, Spain, September 29-30, 2014. Proceedings*, chapter Formal Technical Process Specification and Verification for Automated Production Systems, pages 287–303. Springer International Publishing, Cham, 2014.

[13] G. Hackenberg, C. Richter, and M. F. Zäh. A multi-disciplinary modeling technique for requirements management in mechatronic systems engineering. *Procedia Technology*, 15:5 – 16, 2014. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering.

[14] G. Hackenberg, M. Gleirscher, T. Stocker, C. Richter, and R. Gunther. Macon: Consistent cross-disciplinary conception of manufacturing systems. In *Manufacturing Modelling, Management and Control, 8th IFAC Conference on*, 2016.

[15] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999.

[16] K. Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[17] G. Booch, J. Rumbaugh, and I. Jacobson. The unified modeling language. *Unix Review*, 14(13):5, 1996.

[18] V.-C. Liang and C. J. J. Paredis. A port ontology for conceptual design of systems. *Journal of Computing and Information Science in Engineering*, 4(3):206–217, 2004.

[19] L. Alting and H. Zhang. Computer aided process planning: the state-of-the-art survey. *The International Journal of Production Research*, 27(4):553–585, 1989.

[20] M. Krichen and S. Tripakis. *Model Checking Software: 11th International SPIN Workshop, Barcelona, Spain, April 1-3, 2004. Proceedings*, chapter Black-Box Conformance Testing for Real-Time Systems, pages 109–126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[21] S. Shepard. *RFID: radio frequency identification*. McGraw Hill Professional, 2005.

[22] G. Hackenberg, M. Irlbeck, V. Koutsoumpas, and D. Bytschkow. Applying formal software engineering techniques to smart grids. In *Proceedings of the First International Workshop on Software Engineering Challenges for the Smart Grid*, pages 50–56. IEEE Press, 2012.

[23] N. Lynch and M. Tuttle. An introduction to input/output automate. *CWI quarterly*, 2(3):219–246, 1989.

[24] A. A. G. Requicha and H. B. Völcker. Constructive solid geometry. 1977.

[25] L. Piegl and W. Tiller. Curve and surface constructions using rational b-splines. *Computer-Aided Design*, 19(9):485 – 498, 1987.

[26] C. Legat, J. Folmer, and B. Vogel-Heuser. Evolution in industrial plant automation: A case study. In *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pages 4386–4391, Nov 2013.

[27] Birgit Vogel-Heuser, Christoph Legat, Jens Folmer, and Stefan Feldmann. Researching evolution in industrial plant automation: Scenarios and documentation of the pick and place unit. Technical report, Institute of Automation and Information Systems, Technische Universität München, 2014.

[28] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and quasi-experimental designs for generalized causal inference.* Houghton, Mifflin and Company, 2002.