Component-based Hazard Analysis for Mechatronic Systems

Claudia Priesterjahn∗, Christoph Sondermann-Wölke†, Matthias Tichy‡ and Christian Hölscher§

∗Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn, Paderborn, Germany, Email: cpr@uni-paderborn.de
†Mechatronics and Dynamics, University of Paderborn, Paderborn, Germany, Email: chrisson@uni-paderborn.de
‡Organic Computing, University of Augsburg, Augsburg, Germany, Email: tichy@informatik.uni-augsburg.de
§Design and Drive Technology, University of Paderborn, Paderborn, Germany, Email: c.hoelscher@uni-paderborn.de

Abstract—One cannot image today’s life without mechatronic systems, which have to be developed in a joint effort by teams of mechanical engineers, electrical engineers, control engineers and software engineers. Often these systems are applied in safety critical environments like in cars or aircrafts. This requires systems that function correctly and do not cause hazardous situations. However, random errors due to wear or external influences cannot be completely excluded. Consequently, we have to perform a hazard analysis for the system. Further, the union of four disciplines in one system requires the development and analysis of the system as a whole.

We present a component-based hazard analysis that considers the entire mechatronic system including hardware, i.e. mechanical and electrical components, and software components. Our approach considers the physical properties of different types of flow in mechatronic systems. We have identified reusable patterns for the failure behavior which can be generated automatically. This reduces the effort for the developer. As cycles, e.g. control cycles, are an internal part of every mechatronic system our approach is able to handle cycles. The presented approach has been applied to a real-life case study.

Keywords-Mechatronics, Fault trees, Modeling, Failure analysis, System analysis and design;

I. INTRODUCTION

Mechatronic systems, which have to be developed in a joint effort by teams of mechanical engineers, electrical engineers, control engineers and software engineers, enable innovative design concepts. Each mechatronic system comprises four components. The basic system is a mechanical, electromechanical, hydraulic or pneumatic structure or a combination of these. The state of the basic system is measured by different types of suitable sensors. The information processing unit contains control algorithms, which calculate the desired values for the actuators. The actuators influence the behavior of the system. Additionally, three types of flow are necessary to describe the interaction between the four basic components. The information flow is used to exchange pieces of information like measured values. The energy flow describes the type and amount of energy transferred mainly from and to the basic system. The material flow comes into play, where raw material or semi-finished products are used to fabricate the desired product or where inherent processes use material like oil for hydraulic components. The four components and the different types of flow are depicted in Figure 1.

![Figure 1. Basic Structure of a Mechatronic System](image_url)

Although this abstract structuring of the components of a mechatronic system simplifies the interaction and dependencies of the components and control cycles, mechatronic systems are complex in the details. Hazard analysis within the design process is required to reveal hazards and their probability of occurrence. There are several known techniques for hazard analysis such as failure mode and effects analysis (FMEA, [2]) or fault tree analysis (FTA, [3]).

Mechatronic systems impose various requirements on the hazard analysis technique. As depicted in Figure 1 mechatronic systems contain different types of flow, which have to be regarded within the model. Further, mechatronic systems contain at least one large control cycle. The analysis should thus be able to handle cycles in the system architecture.

In previous works [4], [5] we presented a component-based hazard analysis approach. This approach mainly analyzes a software architecture that is deployed on a computing hardware (ECU - Electronic Control Unit). It determines the combinations of errors that lead to hazards and the hazards’ probabilities on the architecture level. It does not consider the propagation of failures between hardware components and treats all connectors as error-free.

In this paper, we present an extension of this approach that (1) also considers mechanical and electrical components
like valves, electric motors and sensors and (2) also takes the random errors and the failure propagation behavior of connectors into account. This is achieved by defining single components or component structures, including the failure propagation models, that replace the connectors in the model.

In comparison to the related work (cf. section V) our approach enables to handle the connections between hardware components that are part of a mechatronic system. Particularly, we consider the special physical characteristics of the different types of flow. For these flows we have identified reusable patterns for the failure propagation and error behavior. This behavior can consequently be generated automatically and does not need to be modeled by the system developer. This reduces the effort for the developer as well as error-proneness of the analysis. Last but not least, our approach is able to handle cycles.

We applied our analysis to a real-life system: The RailCab – a vehicle that is developed in the RailCab project\(^1\) at the University of Paderborn. We regarded a subsystem of the RailCab (cf. section III-C) which contains all flow types that are part of a mechatronic system as defined in [1]. This allows to study the different physical properties of the connections that are part of such systems.

The paper is structured as follows. In section II we present our application example. Section III introduces our new approach for the component-based hazard analysis of mechatronic systems followed by further ideas in section IV. In section V we reflect the related work before we conclude in section VI.

II. APPLICATION EXAMPLE

As a real world example we use the RailCab project for our analysis. The vision of the RailCab project is a mechatronic rail system where autonomous vehicles called RailCabs work with the linear drive technology, as used by the Transrapid system, but travel on the existing passive track system of a standard railway system.

![Figure 2. Air Gap over Time between primary and secondary part](image)

1http://www.railcab.de/en

The proposed analysis approach is applied to the Air Gap Adjustment System (AGAS) of the RailCab. The AGAS, as the name indicates, serves for the adjustment of the air gap \(\delta(t)\) between primary and secondary part of the linear drive. Different influences such as incorrectly laid tracks, setting processes, wear of tracks and rails cause a fluctuating air gap (Figure 2). The AGAS consists essentially of the primary part of the linear drive, an adjustment actuator, a mechanical structure with springs and a controller. To reach optimal efficiency the air gap has to be adjusted at a minimum value. To improve the efficiency of the linear drive the optimal compromise between required adjustment energy and a minimum value of the air gap has to be found. Further information on the AGAS is given in [6].
III. HAZARD ANALYSIS FOR MECHATRONIC SYSTEMS

Our approach is based on the architecture of the system. Thus, we first specify the architecture consisting of components, their ports and connectors which connect the ports. For each component, we determine the flaws and build a failure propagation model (as a set of fault trees) which relates failures at the ports of the components with internal errors. Failure propagation models for the connectors are automatically generated.

The failure propagation for the whole system is a combination of the failure propagation models of all components and connectors. We specify the hazard as a combination of failures of individual components in form of a fault tree. Finally, we compute the combinations of individual components’ errors which lead to the hazard based on the system failure propagation and the hazard specification.

In the following, we present our approach in more detail. We start with the specification of the architecture in Section III-A. In Section III-B, we present the specification of the failure propagation for components and connectors and discuss the particular characteristics of mechatronic systems. Finally, we present the hazard analysis in Section III-C.

A. Architecture Model

We define the system architecture using internal block diagrams of SysML [7]. Block diagrams consist of blocks, a modular system unit either hardware or software. Each block has a number of ports. Ports specify interactions between blocks. Connectors describe interconnections between ports. They are denoted following the schema \((c_1, p_1 \rightarrow c_2, p_2)\) for port \(p_1\) of component \(c_1\) connected to port \(p_2\) of component \(c_2\) and the flow from \(c_1, p_1\) to \(c_2, p_2\).

Blocks represent any kind of module that is a part of the system, i.e. any kind of hardware and software. In our example (cf. Figure 3) we model mechanical parts, e.g. the hydraulic pump and the valve, electrical parts, e.g. the sensors, computing hardware, e.g. ECUs and software parts, e.g. the control software.

Block diagrams provide so called flow ports that enable to model the flow of items between blocks. This allows modeling the specific flows of mechatronic systems, i.e. material, energy and information (data) as defined in [1]. This provides us with an instrument to describe the propagation of failures in hardware and software components.

In the remainder of this paper we use “components” rather than “blocks” as component is the more general concept. But we keep on using SysML internal block diagrams.

B. Error and Failure Behavior in Mechatronic Systems

We extend the components by an abstract failure propagation which relates random errors in the components to failures at the components’ ports. We follow the terminology of Laprie [8] by associating failures – the external visible deviation from the correct behavior – to the ports where the components interact with their environment. Errors – the manifestation of a fault in the state of a component – are restricted to the internal of the component.

As we only want to study the higher level failure propagation, it is sufficient to consider the failure modes and the errors that trigger the failure modes which are relevant at the architecture level. We thus restrict our attention to the failure propagation taking place at specific ports.

We use Boolean logic with quantifiers (cf. [9]) to formally encode the failure propagation of the system and the occurrence of hazards. The failure propagation relates incoming failures with internal errors, basic Boolean operators (and, or) and resulting outgoing failures. If the outgoing failure \(f_{\text{out}}\) occurs due to both internal error \(e_1\) and incoming failure \(f_{\text{in}}\), this is formalized as \(f_{\text{out}} \Leftrightarrow e_1 \land f_{\text{in}}\). Failures and Errors are typed using an (extensible) failure classification like the one from [10]. For our application example we distinguish the general error and failure classes service and value. We refine the class value by the HAZOP guide words more and less to describe the error and failure classes value more and value less representing value errors and value failures of which the values are greater or less than the expected value.

Failure and error variables are named according to the following schema: \(f_{c,p,t,d}\) and \(e_{c,t}\) for a component \(c\), port \(p\), failure or error type \(t\) and \(d\) for incoming or outgoing.

\[
\begin{align*}
\text{f}_{\text{valve, outcyl1, more, o}} & \iff \text{f}_{\text{valve, inhp1, more, i}} \lor \text{e}_{\text{valve, stcyl}} \\
\text{f}_{\text{valve, inhp1, more, i}} & \iff \text{f}_{\text{valve, outcyl1, more, o}} \land \text{e}_{\text{valve, stcyl}}
\end{align*}
\]

Figure 4. Specification of Errors and Failures in the Component Valve

Figure 4 shows an extract of the specified failures and errors for the valve. The variable \(e_{\text{valve, stcyl}}\) models the error of the valve being stuck open and only allowing the flow of oil towards the hydraulic cylinder and not backwards. \(f_{\text{valve, inhp1, more, i}}\) models the failure that more oil is flowing into the valve than specified. \(f_{\text{valve, outcyl1, more, o}}\) models the failure that more oil is moving out of the valve than specified. Figure 5 shows the fault tree for this outgoing failure. The corresponding Boolean formula is

\[
\text{f}_{\text{valve, outcyl1, more, o}} \iff \text{f}_{\text{valve, inhp1, more, i}} \lor \text{e}_{\text{valve, stcyl}}
\]
Figure 5. Extract of the Fault Tree for an Outgoing Failure of the Valve

For the analysis of the complete mechatronic system we have identified special characteristics concerning the modeling of the failure propagation concerning connectors. This concerns (real-world) physical connectors and (virtual) connectors between software components.

1) Physical Connectors: In every system items are interchanged between the components that build the system. Otherwise the components could not work together. In mechatronic systems the items that are transferred are information, energy and material. Every item that is transferred from one component to another needs a connector. These connectors have to be implemented as hardware e.g. wires, tubes or belts. We have identified four different types of physical connectors in mechatronic systems classified by the items they transport.

Kinetic Energy: The first type of connectors transfers kinetic energy as e.g. a belt that connects two pulleys. Connectors of this type can change their mechanical behavior due to wear. For example the belt can wear out and thus transfer less energy than intended. This correspond to a value error. Additionally, the connector can break off and the energy transfer stops. The belt connecting the pulleys could rip apart and stop transferring the rotation from one pulley to the other. This corresponds to a service error.

Electrical Energy and Information: The second type of connectors transports electrical energy and information. They are typically implemented by wires. Value errors occur due to side effects from the environment e.g. electric fields. Service errors occur due to tear off or breaking.

Wireless Information: The third type of connectors transfers wireless information. Wireless connectors always are exposed to noise from e.g. other wireless connectors disrupting the signals. This noise can lead to value errors when single signals are lost. If all signals are lost the connection breaks causing a service error.

Liquids and Gases: The fourth type of connectors transports liquids and gases (or simply material) as, e.g. tubes. This type of connectors can become leaky due to wear and lose some of the material that is moving through it. This corresponds to a value error. In contrast to the others connectors this kind of value error can also propagated along the opposite direction of the material flow (backpropagation). This fact is discussed in more detail in section III-B2.

Service errors happen, if the tube is teared off or rips apart. Then no more material can be transferred.

In summary, all types of connectors are exposed to value errors and service errors causing outgoing value failures and service failures. They all show a similar failure propagation and error behavior (except from the backpropagation of failures in material flow). Consequently, it is possible to provide patterns that define this behavior. These patterns can be exploited to support the system developer. The developer specifies the kinds of the connectors in the model. The components that model the connectors and their failure propagation and error behavior are automatically generated from patterns according to the kind of the connector.

Figure 6 shows an example for the application of such a pattern. It shows an extract of the AGAS where oil flows from the valve to the hydraulic cylinder. The tube that connects these components in the physical system is modeled as a separate component.

This tube can fail in different ways: Due to wear it can become leaky and lose oil. As a consequence the flow rate decreases and less oil reaches the hydraulic cylinder. This fact is modeled as the value error \( e_{\text{tube,less}} \) in the component \( \text{tube} \) that causes the outgoing value failure \( f_{\text{tube,tu1,less,o}} \) at port \( \text{tu1} \). This failure behavior is modeled by the Boolean formula \( f_{\text{tube,tu1,less,o}} \Leftrightarrow e_{\text{tube,less}} \).

For different reasons the tube can be pulled off. Then, no more oil is transfered to the hydraulic cylinder. This fact is modeled as the service error \( e_{\text{tu1,su}} \) that causes the outgoing service failure \( f_{\text{tube,tu1,su,o}} \) at port \( \text{tu1} \). This failure behavior is modeled by the Boolean formula \( f_{\text{tube,tu1,su,o}} \Leftrightarrow e_{\text{tu1,su}} \).

2) Backpropagation of Errors in Material Flow: In many cases a hardware connector is designed to carry a flow in one direction. For example a tube that carries oil into the rod side chamber of the hydraulic cylinder and another tube that carries oil out of the rod side chamber of the hydraulic cylinder. Consequently, the propagation of failures is defined for this one direction only. But errors that occur in such connectors can also propagate in the opposite direction of the actual flow. The tube carrying oil away from the rod side chamber can become leaky. While it is loosing oil the pressure in the rod side chamber drops. This causes the piston to extend further. This corresponds to a too high extension of the cylinder. As a consequence the air gap becomes too small. To include this fact into our analysis we have to model each connector that carries material into one direction in a way that failures can propagate against the actual material flows. The failure propagation of the
connected component decides whether this failure effects this same component.

![Figure 7. Connector with Backpropagation](image)

Figure 7 shows the components RodSideChamber and Tube. The latter carries the oil away from the rod side chamber. The error $e_{\text{tube, leak}}$ models the case that the tube becomes leaky. The component RodSideChamber provides ports for incoming and outgoing flows thus allowing for the backpropagation. Error $e_{\text{tube, leak}}$ propagates via the failures $f_{\text{tube}, tu1, \text{backprop}, o}$ and $f_{\text{rs}, r1, \text{backprop}, i}$ into the component RodSideChamber. The fault tree in this component propagates it to the outgoing failure $f_{\text{rs}, r3, \text{more}, o}$. As a consequence the air gap becomes too small.

3) Combined Material and Kinetic Energy Flow: Inherent processes of the mechatronic system use material flows to transport energy, e.g., cooling systems or hydraulic components. In the formula for the kinetic energy $E_{\text{kin}} = \frac{1}{2}mv^2$ this combination gets obvious. The transported kinetic energy is proportional to the transported mass, i.e. the amount of material. Referring to the previous section this means, we have to model two different connectors (here liquids and gases as well as kinetic energy) in parallel (cf. Figure 6). Additionally, we have to model the interconnection between the different flow types.

![Figure 8. Failure Propagation Model of the Failure Dependencies between Material Flow and Energy Flow in the Tube](image)

In our application example the hydraulic cycle is investigated as an example for the combined material and energy flow. The hydraulic oil constitutes a material flow that also transport energy through the cycle via tubes. The kinetic energy is transformed into hydraulic energy to move the piston of the hydraulic actuator. A loss in the material due to leakage also reduces the transported energy and thus the energy required to move the piston. An example of a failure propagation model describing these dependencies is shown in Figure 8. This failure propagation model can be automatically derived from the equation presented above. This is achieved by abstracting from concrete values to the failure classes value, less and more.

4) (Virtual) Software Connectors: We clearly distinguish between software and hardware components, i.e. we model the computer hardware (ECU) and the software as separate components. Software can only communicate via hardware with other system components. Especially, software components that are deployed on different ECUs have to communicate via the hardware connections between these ECUs. The direct connections between the software components are only virtual and not used in the real application. When analyzing the failure propagation between such software components, it is necessary to take the hardware connectors into account as well. We do this by automatically generating separate hardware connectors and associated ports on the ECU for each software connections. The system developer only has to specify the routing of software connections over the hardware components. The failure propagation of the hardware components is adapted accordingly.

![Figure 9. Failure Propagation via Virtual Software Connectors and Corresponding Hardware Connectors](image)

The architecture shown in Figure 9 consists of the three software components SWHall, SWPos and SWCtrl deployed on the two ECUs ECUEval and ECUCtrl. There are two software connectors that only exist at design time: $(\text{swhall}, \text{sh1} \rightarrow \text{swctrl}, \text{sc1})$ and $(\text{swhall}, \text{sh2} \rightarrow \text{swctrl}, \text{sc2})$. In order to distinguish the failure propagation of the two separate software connectors we also have to model two separate connectors between the ECUs: $(\text{ecueval}, \text{ev}3 \rightarrow \text{ecuc}, \text{sc1})$ and $(\text{ecueval}, \text{ev}4 \rightarrow \text{ecuc}, \text{sc2})$.

In the example the error $e_{\text{swhall, sv}}$ is raised in the component SWHall. It is propagated to the software connector $(\text{swhall}, \text{sh2} \rightarrow \text{swctrl}, \text{sc2})$. As this connector does not exist in the real-world system we have to model the corresponding path to the component SWCtrl via the two ECUs. This path is illustrated by the bold solid lines and its adjacent connectors.

If we had only modeled one connector describing the information flow between the ECUEval and ECUCtrl, a
failure going out of ECUEval port ev4 would not only trigger the incoming failure on ECUCtrl port ec2 but also an incoming failure on ECUCtrl port ec1. The propagation of an incoming failure on ECUCtrl port ec1 following the dotted line would trigger the incoming failure on SWCtrl port sc4 and satisfy the failure propagation of SWCtrl. As a result an outgoing failure on SWCtrl port sc5 would be produced. In the real system this failure would only be triggered, if an outgoing failure on ECUEval port ev3 occurred.

C. Component-based Hazard Analysis

To describe hazards and the combinations of failures at ports that can cause them we employ standard fault tree analysis (FTA [3]). In a fault tree the hazardous event is shown as top of a fault tree. This top node is caused by a combination (and, or) of its child nodes. This continues until the leaf nodes of the tree are reached. These leaf nodes describe the basic events which indirectly caused the hazardous event on the top. In our case, the basic events are failures of the system components. This part of the analysis is done manually.

A hazard (top event) corresponds thus to a hazard condition in form of a Boolean formula which employs only the operators for ∨ and ∧ and a subset of the outgoing failure variables of the system components.

It is, thus, a symbolic encoding of all combinations of errors which lead to the hazard. Then, we compute the implicants of this Boolean formula. For example, an implicant for the hazard air gap too small is service error in hall1 ∧ more error in hall2. We refer the interested reader to [4], [5] for a more detailed presentation and formalization of the hazard analysis.

The architecture of mechatronic systems does typically contain cycles. The main reason for this is that mechatronic systems contain feedback controllers which use previously computed values as inputs. Additionally, a mechatronic systems does interact with the environment and the environment interacts with the mechatronic system. Consequently, cycles can neither be eliminated from the system nor encapsulated into a single component.

We use Binary Decision Diagrams (BDD) [12] to encode the failure propagation using Boolean logic. Without any special handling of cycles, the resulting Boolean failure propagation would degenerate to the single BDD node true. For the case of monotonic increasing failure propagation formulas, a special BDD operator has been presented by Rauzy in [9]. This operator avoids the mentioned degeneration. Consequently, we employ this special operator in our approach to handle cycles in our failure propagation models. A detailed description of this step can be found in [4].

It is often the case that a hazard cannot be excluded. If (independent) probabilities for all internal errors for a given mission time are available, the hazard probability for the given mission time is computed. For efficient computation of the hazard probability please see [5].

We have computed the probability of the hazard air gap too small for our application example. This hazard occurs 9.0740 times per 10^6 hours.

This relatively high failure rate is due to the chosen tube components, whose failure rates are approximately 2 times per 10^6 hours. In our example their are four tube components, which contribute to the increased hazard. Choosing different tubes using materials with a lower failure rate, e.g. with a failure rate of 0.5 times per 10^6 hours. The overall hazard is reduced to 8.8483 times per 10^6 hours. This example clarifies that it is important to model the connectors explicitly.

IV. FURTHER IDEAS

In this section we present two further ideas that improve the accuracy of our approach. The material that flows through the system has to be explicitly considered during the hazard analysis. This is due to the fact that in reality material is a separate component of the system. Further, we propose a specialization of value errors into subclasses that represent intervals of values. This specialization is needed as the system can react differently to value errors of different values.

Figure 10. Fault Tree for the Hazard Air gap too small

Figure 10 presents a simplified extract of the fault tree for the hazard air gap too small. According to the mechanical and control engineers of the RailCab project, it is a hazardous state when the air gap is too small. In this situation the rotor could contact the stator which would destroy the rotor.

The hazard specification as well as the failure propagation models of components and connectors are Boolean formulas consisting of failure and error variables. We build the conjunction of all these formulas and use the exist-quantification to remove the failure variables as they are only required for the propagation of failures between components. The resulting Boolean formula does only contain error variables.
A. Modeling Failure Properties of Flowing Material

Liquids or gases that move through the system are subjected to aging and change their consistency or contaminate and thus change their properties. For example, if oil ages it starts to clump and becomes more viscous. When moved with a certain intended velocity it will be slower than expected.

This kind of error differs from the kind of random errors that have been considered so far. A failure due to aging and contamination does happen randomly. But they also add a systematic error to the system as all components will exhibit the same faulty behavior over a certain time span once the material (liquid or gas) has changed. For example, the oil flows more slowly.

We propose to model liquids or gases as separate components of the system. Errors in these components propagate as failures through the system. Figure 11 shows an extract of the AGAS where hydraulic oil flows from the hydraulic pump to the valve and back. The hydraulic oil is modeled as a separate component that is connected to all other hydraulic components. The error of the oil changing its properties is modeled as a value error in the component HydraulicOil. Once the error is activated a value failure is propagated to the multi port and thus to all connected components.

B. Specialization of Value Errors

In some cases the same type of error, mainly the value error, could cause different impacts on the system. Considering the aging of the hydraulic oil, the hydraulic actuator needs more force to move the more slowly moving oil. Depending on the age of the oil (resulting in a certain viscosity) the controller is able to compensate a failure in the position of the hydraulic actuator. If the oil gets too tough the controller is not able to compensate that failure in the current moment of time and this leads to a stationary failure in the control loop. This example shows that the controller is able to compensate value failures of a certain range, but if a threshold is exceeded the remaining value failure could lead to a hazard in the system.

To further improve the hazard analysis for mechatronic systems, we suggest to define classes of value errors that cover certain ranges of values. This is due to the fact that the system behaves differently under the influence of deviations of different extents.

V. RELATED WORK

Several approaches for the component-based hazard analysis are proposed in the literature [13], [14], [10], [15], [16], [17], [18]. They all have in common that they establish a relation between the inputs and outputs of a component by abstracting from the system behavior. Whereas the outputs consist of a logical combination of propagated failures from the inputs and inherent errors of the component. The analysis is performed on the architecture level. Consequently, these approaches enable the hazard analysis for complex systems. The reuse of components and their failure propagation behavior is possible. But in contrast to our approach it is not considered.

Approaches that take properties of connections into account are [17], [15], [18]. In Hierarchically-Performed Hazard Origin and Propagation Studies (HiP-HOPS) the data (e.g. pressure, temperature, etc.) is covered by a parameter [17]. This policy is also applied in the generalization annotation of failure patterns [18]. The Failure Propagation and Transformation Calculus (FPTC) suggests a remodeling for the flow, which is exemplary justified by a communication protocol, that could transform the failure type [15]. The remodeling consists of the substitution of the flow by an extra component. The connections between components only model the flow of information. In addition, our approach takes into account the physical properties.

In this paper we have pointed out that the handling of cycles is essential for modeling and analyzing mechatronic systems. Besides our approach, only Wallace [15] propose a solution by means of a fixed-point algorithm to handle cycles within the system architecture.

An approach that considers the complete system behavior rather than an abstraction is the Deductive Cause-Effect-Propagation Origin and Propagation Analysis (DCCA) [19]. Unlike the aforementioned techniques the DCCA uses the specified behavior of the system instead of specifying the models for the failure behavior. The behavior is modeled by state automata and extended by failure automata. On the basis of this formal model modelchecking is used to verify, if a hazard might occur and which errors have to occur for this hazard. In contrast to our approach no part of the system can be reused for the analysis and physical properties are ignored.

VI. CONCLUSION

In this paper, we present an approach that allows for the component-based hazard analysis for (entire) mechatronic
systems including hardware and software components as well as the connections between them. Our approach enables to handle the connections between hardware components that are part of a mechatronic system. Particularly, we consider the special physical characteristics of the different types of flow. For these flows we have identified reusable patterns for the failure propagation and error behavior. This behavior can consequently be generated automatically and does not need to be modeled by the system developer. This reduces the effort for the developer as well as error-proneness of the analysis. Further, our approach is able to handle cycles.

In the future, we plan to further develop the modeling of failure properties of flowing material and the specialization of value errors as sketched in section IV.

To increase the accuracy of our analysis we plan to implement a temporal hazard analysis that also considers propagation times and calculates time dependencies between errors. This allows to prevent false negatives and false positives that do occur because of real-time properties of mechatronic systems.

In order to analyze reconfigurable mechatronic systems we plan to extend our approach by the hazard analysis of the reconfiguration behavior [20].

ACKNOWLEDGMENTS

This work was developed in the course of the Special Research Initiative 614 – Self-optimizing Concepts and Structures in Mechanical Engineering – University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

Matthias Tichy was on leave from the Software Engineering Group at the University of Paderborn while carrying out this research.

REFERENCES


