Tool Assisted Model Based Multi Objective Analyses of Automotive Embedded Systems

Saimir Baci[1], Henrik Kaijser[1], Henrik Lönn[1], Matthias Tichy[2], Wenjing Yuan[3]
[1]Volvo Group, Advanced Technology and Research, Gothenburg Sweden
[2] Software Engineering Division, Chalmers | University of Gothenburg, Gothenburg, Sweden
[3]Arccore AB, Gothenburg, Sweden

***Abstract*— Automotive embedded systems are increasingly critical for vehicle functionality. Their scope and impact increase and as a consequence their complexity and criticality. Cost is constantly in focus, which emphasizes the need for efficient development methods. Securing functional and non-functional properties in such setting requires systematic and stringent methods, including model based analysis. This paper introduces a methodology and tool support for embedded automotive systems, where multi-objective analyses drive iterative design. Tool support is developed based on the EATOP platform for EAST-ADL. Plugins supporting version consistency control, fault tree analysis and analysis of linear property annotations including power, energy and cost are reported. The tooling is validated on an example system design and the methodology and part of the tooling evaluated with engineering experts.**

Keywords—Embedded Systems; Real-time Systems; Model Based Development; AUTOSAR; EAST-ADL; Quality; Safety

I. INTRODUCTION

Automotive systems continuously face increasing expectations. Functionality and cost are constantly in focus, and linked to the latter is reduced development time. A recent trend is the emerging use of agile methods and continuous integration in automotive development, which calls for shorter development iterations and increased automation.

Securing functional and non-functional properties in such setting requires systematic and stringent methods, including model based analysis. Examples of important qualities to analyze are timing, safety, product cost, energy consumption and various correctness criteria.

The demand for short development time, consistency and correctness means that the required analyses should be performed on the original constructive model or on automatically generated models. The EAST-ADL [6] architecture description language provides the opportunity to represent several engineering aspects in the same model. It is based on a structural model capturing the electrical and electronic architecture in 4 levels of abstraction:

- Implementation Level: Concrete software architecture and code
- Design Level: Physical topology, concrete functional architecture with allocation.
- Analysis Level: Abstract functional architecture
- Vehicle Level: Feature content

Thus, the architecture description reduces the amount of software details in four steps: Design level description ignores code structure, interfaces to execution platform, interfaces to I/O, etc. Analysis level ignores allocation and topology, physical sensors and actuators. Vehicle level ignores solution and integration aspects and focus on individual features and their appearance to the customer.

The software details on implementation level are represented using the AUTOSAR [5] approach, which is a fundamental standard for automotive software.

EAST-ADL provides extensions to complement the structural model with additional aspects: Behavior, Timing, Dependability, Variability, Cost as well as:

- Energy and power: Mode-based steady state energy and power annotations
- Take rates: Definition of production volumes and take rates of vehicle types and features
- Requirements: Requirements allocation, traceability and formalization
- Verification and Validation: Definition of V&V procedures and results

The EAST-ADL is using the AUTOSAR meta modelling rules. Therefore, the above extensions are compatible to AUTOSAR models. Further, by prescribing syntax and semantics for these annotations, tools and engineers can interpret the information unambiguously and consistently.

In this paper, we present a methodology for working iteratively with embedded system design, in a setting with multi-objective model based analysis and design. Quality aspects covered include fault propagation, linear property annotations like energy, power and cost. The modelling and analysis approach is based on the EAST-ADL Open Tool Platform [2].

Model-based quality evaluation (MBQE) techniques have been developed for a variety of quality attributes, including safety, security, reliability, availability and performance. Similar to other engineering disciplines the core idea of MBQE techniques is to construct a quality evaluation model from a system model and use this model to gain knowledge

about the quality of the system by checking them against formally specified quality requirements. For the construction of quality evaluation models mostly architectural models are used, since the decisions taken in the architecture design phase have a significant impact on the system quality and MBQE provides an effective basis to choose an appropriate design. The generated quality evaluation models are different for each quality attribute [7]. As examples Discrete Time Markov Chains (DTMCs) are used for reliability evaluation [8], Layered Queueing Networks (LQN) and Continuous Time Markov Chains (CTMC) are used to predict performance attributes [9], and Fault Tree (FT) and Failure Propagation Models (FPM) are commonly used for evaluating system safety [10][11][12].

Our approach differs from the mentioned ones that it uses complete models for both the architectural configuration [17] as well as, in the case of safety, the error behavior, i.e., the error behavior can be modeled and analyzed independently of the architecture. The other approaches typically embed the quality evaluation relevant behavior into the components of the architectural configuration and are therefore restricted by the architecture. In contrast, the model of the error behavior can evolve independently of the architectural configuration in our approach to give the engineers (e.g., safety engineers) more freedom to improve and refine the error model. For this, we provide specific refactoring operations and version consistency support to handle inconsistencies between the models.

The paper will next introduce the modeling and analysis capabilities used, followed by a methodology description for iterative design guided by multi-objective analysis. The concepts are then illustrated using a particular analysis, error propagation applied on a brake by wire example system. Before concluding, a description of the prototype tooling is provided as well as an evaluation of the approach with intended users.

II. Multi-Objective Models

System development is always a trade-off among a large set of capabilities. In order to make the right design decisions, reliable assessment of system properties is required. Model-based analysis provides means for well-defined and semantically sound assessment. Because the input is a model, system specification can be captured in a systematic and understandable way.

Because many analyses are highly specific and because most analysis tools are tailor-made for a specific purpose, input models often need to be unique for the specific analysis. This has several consequences:

- Efficiency: Common aspects of the input are duplicated
- Consistency: System definition may deviate between analyses
- Tool and data: Same system needs to be managed in different tools with different learning curves, data management, etc.

In order to mitigate these problems, a common model, annotated with information for different analyses can be used. This way, variants and versions are consistent for all analysis purposes, and the amount of redundant modeling is minimized.

Examples of analysis purposes that can share the same core model include:

Timing – Response time: Annotate existing functional structure with timing properties and requirements.

Dependability- Fault Tree analysis: Complement architectural configuration with error propagation models

Cost – Piece and development cost: Take rates and cost annotations complement functional and hardware elements.

Energy – Steady state and mode based energy and power consumption: Can annotate functional and hardware elements.

In the tooling accounted for in this paper, the latter three categories are supported, as well as several other linear properties. Figure 1 shows the hardware architecture of a fictitious brake by wire system. The lower part shows the hardware architecture containing 5 instances of ECU types as well as sensors and actuators. The type declarations are annotated with power consumption as shown at the top. The actuators' power consumption varies with operating mode. With the linear properties analyzer, the desired mode is selected and the total power is summed up to 4*60W+5*17W=325W which is less than the prescribed maximum power of 400W. The sensors' power consumption is ignored for simplicity. Note also that bus and power lines as well as most type relations are suppressed in the diagram.

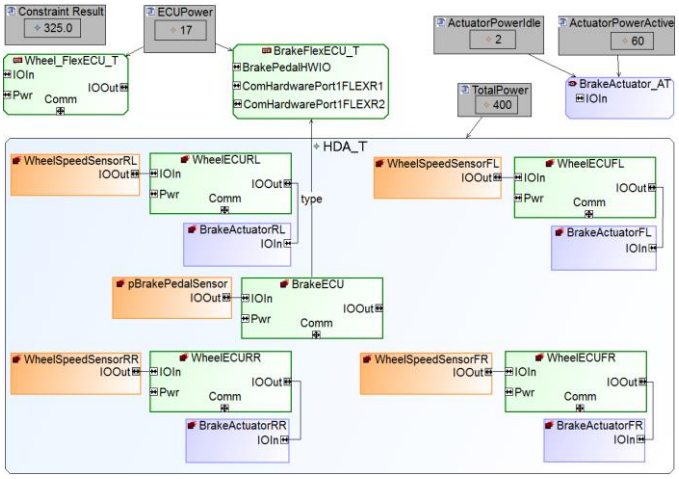

Figure 1. Prescribed and computed power consumption for components of a brake by wire system in active and idle mode.

III. Methodology

Engineering models will be defined in several ways, typically depending on the degree of reuse and the kind of tooling available. Below, a simple pattern will be described, which would motivate and explain which kind of tool support is typically needed.

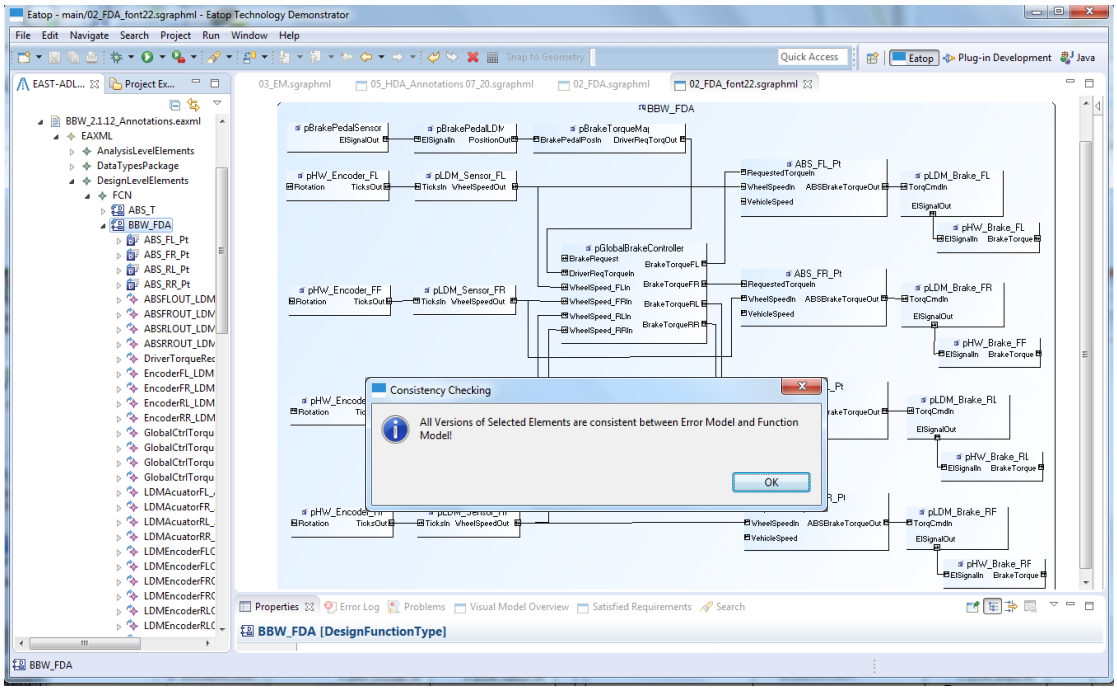

Figure 2. Version consistency check on Functional Design Architecture of Brake By Wire

A. *Defining*

Defining the model which is subject to analysis may be done manually, or in cases where a source model exists, automatically or semi-automatically. Such annotated model needs to contain the fundamental properties of the element to be analyzed, for example timing annotations or failure rate.

B. *Analyzing*

Analysis often involves transferring the annotated model to a format acceptable to a particular analysis tool. Ideally, this should be completely automatic to avoid mistakes and allow the annotated model to be the documentation of the analyzed entity. In case assumptions or engineering decisions need to be provided to the analysis tool, these need to be documented to secure repeatability and traceability.

C. *Understanding*

Many analysis results are non-trivial and thus difficult to understand. Effective views representing the source model and analysis result are thus required.

D. *Evolving*

The purpose of analysis is often to evaluate a design in order to iteratively improve it. On the basis of analysis results, both the architecture model and its analysis annotations may be changed. When such changes are performed, it is important that architecture models and analysis annotations stay consistent. It is also necessary to iteratively modify and asses models and annotations with respect to all relevant properties.

IV. Dependability Analysis Concepts

In the following, we describe the instantiation of the four steps discussed in Section III for the analysis of a system's dependability, particularly, safety. Automotive systems are highly safety-critical and, thus, processes require that safety is properly addressed during system development. A particular employed technique is fault tree analysis (FTA) as part of hazard analysis. Fault tree analysis is a top-down approach to identify, which errors (or combinations thereof) of system parts can result in a certain hazard.

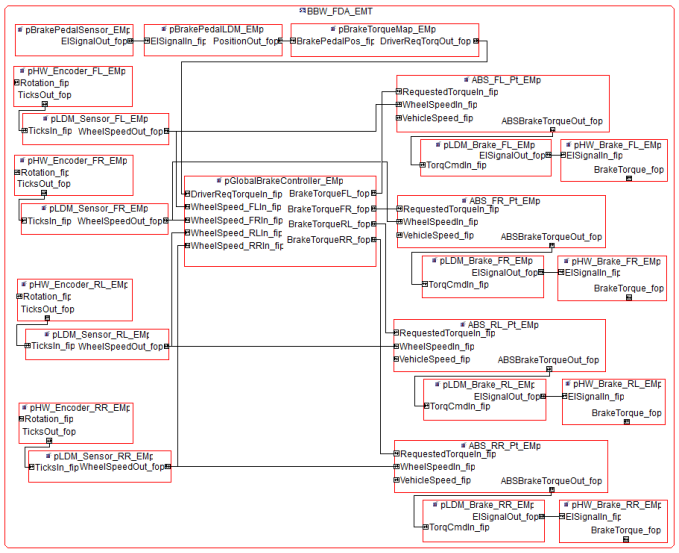

Figure 3. Error Propagation Model of brake system

A. *Defining*

EAST-ADL supports a specific view, called error model, for modeling the errors and how errors of one component influence other components, and eventually result in a hazard, see . This influence of one component's errors to other components has been termed "failure pathology" by Avizienis

et al. [14] and is the basis of many architecture based safety analysis methods [10][11][12]. As part of the goal to support the engineer to analyze the system's safety, our approach supports an automatic generation of an initial error model from the architectural model. The generated error model reflects the structure of the functions in the design model of EAST-ADL and is based on one to one mapping patterns between function models and error models, e.g. component types are mapped to error types, ports of the components are mapped to ports at the error type.

Furthermore, the generated behavior of the error model is pessimistic, i.e., it assumes that errors of one component always result in errors of other components.

As this initial error model is only based on the structure, it does not resemble the real error behavior. Hence, the error model, particularly the behavior, needs to be refined by the engineer to properly resemble the system's error behavior.

To support the engineer, we provide two refactoring operations on the error model. As the initial mapping is quite verbose, the refactoring operations deal with merging different parts of the architectural configuration, i.e., merging multiple error types inside a containing error type and merging of multiple ports of an error type. As an example, the joining of multiple error types inside a containing error type results in a simpler structure, which might also be easier to analyze and understand. Note that changes in this step only concerns the correct representation of error propagation, and does not address system changes.

As those refactorings change the error behavior, they are not automatically executed but instead provide the safety engineer more complex editor operations compared to the normal graphical editor operations.

B. Analyzing

In order to analyze the specified error mode, the EAST-ADL error model created by the previous step is automatically transformed to a fault tree model, which is subsequently analyzed by the HipHops tool [18] with respect to minimal cut sets (see Figure 4). Cut sets are sets of failures whose combination result, in the context of safety, in a hazard. Minimal cut sets are cut sets where no failure can be removed without the cut set not being a cut set anymore. Minimal cut sets are important from a safety perspective, as they are the "simplest" scenarios how failures in the system can result in a hazard and, thus, should be addressed first.

C. Understanding

In order to understand results, a combination of views of the architectural configuration, the analysis input and the analysis results is required. In this case, the error propagation model and the resulting fault tree are key elements. Also, the minimal cut set, i.e. the sets of faults leading to each system failure, are fundamental and thus highlighted in the tree view of the error propagation model.

D. Evolving

An important part of the development lifecycle is the evolution of the system. In the context of the safety

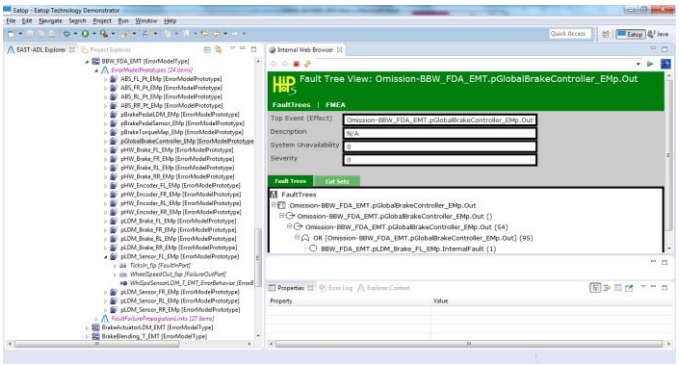

Figure 4. Fault Tree from HiP-HOPS analysis tool

assessment, both the architectural configuration as well as the error model will be updated regularly. Furthermore, the updates of one model will lead to updates in the other, e.g., the analysis of the error model might result in the addition of a redundant sensor in the architectural configuration, which in turn requires a change in the error model to properly capture the new failure behavior. Such changes are manual in our current approach, although semi-automatic approach can be used where a new error propagation model is generated and compared/merged with the existing error propagation model. New or modified annotations to represent non-functional properties like timing or energy consumption may also be required in the architectural configuration.

To support version management of individual model elements, a fine-grained version annotation approach was introduced. Each model element gets a version annotation that is manually increased if a new stable version of that model element has been developed. As the model elements of the architectural configuration and the error model are internally linked, the system can simply compare the two versions of the model elements and check whether one model has been updated, see Figure 2.

As an example, when analyzing the error propagation model, it was found that there was a single point of failure of the brake pedal sensor leading to brake system failure. The flaw was corrected, and version consistency control applied to identify which parts of the error propagation model was affected by the change.

V. PROTOTYPE TOOLING

The prototype tooling is implemented in Java as a set of six Eclipse [1] plugins. Eclipse is an open source platform that facilitates rapid development of integrated features based on a plugin architecture. The platform comprises a runtime and many plugins that provide support for tool development. On top of Eclipse the prototype tooling makes use of the EATOP platform [2], which is an emerging *infrastructure* platform for EAST-ADL. This means that EATOP provides serialization, deserialization and validation of the multi-objective system model. In addition, EATOP supplies a tree view for model navigation and editing, as well as programmatic access to the model. The prototype tooling plugins mainly rely on EATOP functionality, but also take advantage of some functionality in the Eclipse platform, see Figure 5.

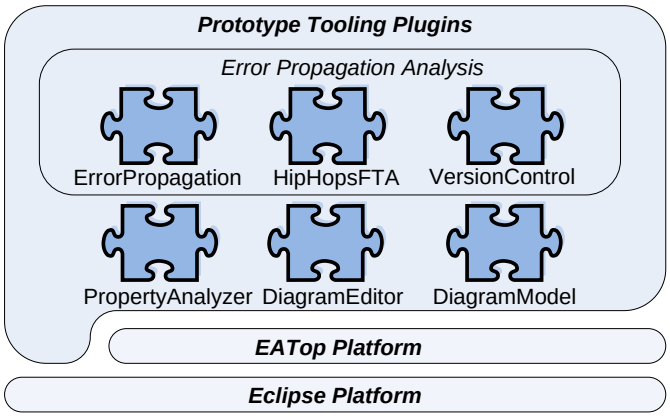

Figure 5. Software architecture of the prototype tooling

There are three plugins for the error propagation analysis. First, the `ErrorPropagation` plugin creates the error model from the model of the architectural configuration (i.e., the structure of the architecture defined by components and connectors [17]) and also supports the refactoring operations. Second, the `HipHopsFTA` plugin exports the error model to HiP-HOPS format. Third, the `VersionControl` plugin handles version control and consistency between the architectural configuration and the error model.

In addition to the plugins for error propagation, there is a `PropertyAnalyzer` plugin. This plugin can be used for analyzing linear properties like power consumption, current, cable length, piece cost etc. as mentioned above.

Finally, there are two diagram plugins used by the tooling prototype. These are general in the sense that they are applicable for any modeling objective that benefits from diagram representation. The `DiagramModel` plugin represents a diagram model and has been auto-generated from an xml schema using the Eclipse Modeling Framework [3]. The `DiagramEditor` plugin is an editor based on the Graphical Editor Framework [4]. Diagrams are created by using drag-and-drop of system model elements in the tree view to the editor canvas. In order to keep the system and diagram models synchronized, there is a reference from each node and edge in the diagram to the underlying system model element. Model element listeners are utilized on both models to detect any changes in one of the models that need to be reflected in the other.

VI. Evaluation

In order to evaluate the presented method, we performed a confirmative case study at Volvo. In the following sections, we discuss the evaluation criteria, the method of the evaluation, as well as the results.

A. Evaluation Criteria

Carvalho [15] provides validation criteria for measuring the outcome of a design research: artifact success, generalization, and novelty. Artifact success, as shown by name, is used for measuring whether the artifact is a success or not. Generalization defines the range of applicability for the outcome, so whether the applicability is restricted to specific situations or not. Novelty indicates the improvements of the measured outcome comparing with existing knowledge, i.e., the outcome is better compared to before.

B. Evaluation Method

We used semi-structured interviews as a method to perform data collection on the evaluation criteria. Semi-structured interviews are built on a common set of questions but the interviewer can react to the background and questions of the interviewee by adaptions of the questions during the interview. The interviews have been performed individually in order to avoid bias by a group interview as well as ensure that all feedback from all participants is gathered.

Each interview starts with an introduction, which provides a general concept of the research and background. Afterwards, each part of the approach is described using both a presentation as well as a demo of the tools. Thereafter, the interviewees answer the questions and rate the features of our approach with respect to the evaluation criteria. Seven interviews have been performed each around one hour. Four of the interviewees are experts in automotive architecture; three of the interviewees are safety experts. We chose these two groups of experts to cover both parts of our models, e.g., architecture and error models.

After the interviews, the answers from the interviews have been summarized in relation to our evaluation criteria and subsequently analyzed.

C. Results

In the following, we discuss the chosen evaluation criteria artifact success, generalization, and novelty in more detail for the four activities in our method.

We measure artifact success in terms of “Ease of Use”, “Ease of Understanding” and “Achieving its goal” on a likert scale of 1 to 5, with 1 being poor and 5 being excellent.

	Defining	**Analyzing and Understanding**	**Evolving**
Ease of Use	4	3.8	3.9
Ease of Understanding	3.4	3.8	4
Achieving its Goal	4.1	4	3.9

Table 1 Artifact Success

Overall, we can see that the artifact success is generally rated highly (around 4). The only criteria rated less good is “Ease of Understanding” for the “Defining” step. The specific reason for that is that the interviewees did not find it as easy to understand the refactoring operations and their purpose on the error model, compared to other parts.

With respect to the generalization and novelty criteria, we collected qualitative answers from the interviewees. Overall

both generalization and novelty are confirmed. For example, the interviewees stated on generalization: “It should be a part of [our] development process in the future.”, “This feature is easy to use, but setting the logic probability is hard to achieve. But it achieves its goal quite well.”, “Yes, this feature is useful everywhere and highly needed.”, ”The version control is definitely applicable for the current work and it is always a big issue to handle.”. “Yeah, there are a lot [of] modifications needed to be done manually. When need to update the model, add or move hardware or functions, those changes are quite common. But it may not be able to automate since the machine cannot make decision for that. But it would be good to have something automatic for a complex project.”.

Examples for statements on novelty include “I don’t think there is any current tool which has this feature, as far as I know.”, ” There are lots of tools [that] are doing the version control.”, “Other tools have similar features, but not for this purpose.”

In summary, we conclude that the approach has been positively evaluated.

D. Threats to Validity

Threats to validity are usually categorized into internal and external. Internal validity refers to the extent that the evaluation result is warranted and external validity refers to the extent that the evaluation result can be generalized to other situations and to other people [16]. In this evaluation, the threat to internal validity is instrumentation. The instrumentation here refers to the measuring scale for scoring different artifacts. Since different interviewees could have different understanding of the score, it means the same evaluation feedback may differ for different interviewees.

There are several threats to external validity in this research. Purposive sampling, only the ones who are familiar with the research background in Volvo are selected as interviewees, this makes the evaluation result hard to be generalized to other people in other organizations. Similarly, only seven interviews have been conducted during the evaluation. However, in this evaluation, experts from industrial practice with different backgrounds and expertise are selected in order to avoid selection bias. So we conclude that the result is adequate valid from generalization perspective. In interviews, the Hawthorne effect might result in more positive feedback, when the researcher states that the whole interview will be recorded. To counter this threat, the interviewer asks for honest feedback at the beginning of the interview and the individual recordings were kept anonymous and not shared within the organization. Concerning the influence of Mono-method bias, we performed both subjective measurement (interview) and objective measurement (quantitative data on a Likert scale on the valuation criteria) to make sure a valid conclusion is drawn.

VII. Conclusion and Future Work

We presented an approach for model-based analysis of automotive systems with respect to multiple different quality attributes like performance, timing, costs, and safety. The approach is based on a common modeling language, EAST-ADL, with extensions to model the different quality aspects. We explained in more detail the four steps of our approach by applying it to safety as quality attribute. The evaluation by a case study at Volvo using semi-structured interviews showed that the approach is applicable in practice.

Currently, our approach is restricted that we support only a full regeneration of the error model based on the model of the architectural configuration. We are planning to improve the approach working on implementing incremental model transformations [13] [19] in order to incrementally adapt parts of the error model if only parts of the model of the architectural configuration change.

References

[1] Eclipse Foundation: https://eclipse.org/

[2] EATOP EAST-ADL TOol Platform: https://www.eclipse.org/EATOP/

[3] Eclipse Modelling Framework: http://www.eclipse.org/modeling/emf/

[4] Eclipse Graphical Editing Framework: https://eclipse.org/gef/

[5] AUTOSAR Partnershio: www.autosar.org

[6] ÉAST-ADL Association: www.east-adl.info

[7] L. Grunske. Early quality prediction of component-based systems–a generic framework. Journal of Systems and Software 80.5 (2007).

[8] K. Goseva-Popstojanova and K. S. Trivedi. Architecture-based approach to reliability assessment of software systems. Perform. Eval, 45(2-3):179–204, 2001.

[9] S. Balsamo, A. DiMarco, P. Inverardi, and M. Simeoni. Model-Based Performance Prediction in Software Development: A Survey. IEEE Transactions on Software Engineering, 30(5):295–310, 2004.

[10] M. Güdemann and F. Ortmeier. Probabilistic model-based safety analysis. In Alessandra Di Pierro and Gethin Norman, editors, Proceedings Eighth Workshop on Quan- titative Aspects of Programming Languages, volume 28 of EPTCS, pages 114–128, 2010.

[11] H. Giese and M. Tichy. Component-based hazard analysis: Optimal designs, product lines, and online-reconfiguration. In Computer Safety, Reliability, and Security, 25th International Conference, SAFECOMP 2006, Gdansk, Poland, September 27- 29, 2006, Proceedings, volume 4166 of Lecture Notes in Computer Science, pages 156–169. Springer, 2006.

[12] C. Priesterjahn, D. Steenken, and M. Tichy. "Timed hazard analysis of self-healing systems." Assurances for Self-Adaptive Systems. Springer Berlin Heidelberg, 2013. 112-151.

[13] D. Hearnden, M. Lawley, and K. Raymond. "Incremental model transformation for the evolution of model-driven systems." Model Driven Engineering Languages and Systems. Springer Berlin Heidelberg, 2006. 321-335.

[14] A. Avizienis, et al. "Basic concepts and taxonomy of dependable and secure computing." *Dependable and Secure Computing, IEEE Transactions on* 1.1 (2004): 11-33.

[15] J. Carvalho. ”Validation Criteria For Outcomes Of Design Research”, The International Workshop on IT Artefact Design & Workpractice Intervention, 10 June, 2012, Barcelona.

[16] P. Runeson and M.Höst. "Guidelines for conducting and reporting case study research in software engineering." *Empirical software engineering* 14.2 (2009): 131-164.

[17] N. Medvidovic and Richard N. Taylor. "A classification and comparison framework for software architecture description languages." *Software Engineering, IEEE Transactions on* 26.1 (2000): 70-93.

[18] Y. Papadopoulos, J. McDermid, R. Sasse and R. Heiner,“Analysis and Synthesis of the Behaviour of Complex Programmable Electronic Systems in Condition of Failure“ *Reliability Engineering and System Safety*, Volume 71, pp. 229-247.

[19] S. Getir, L. Grunske, C.Bernaska, V. Käfer. T. Samwald, and M. Tichy. „CoWolf – A Generic Frameowork for Multi-View Co-Evolution and Evaluation of Models.“ 8th International Conference on Model Transformations (ICMT 2015), L’Aquila, Italy, July 20-21, 2015.

Acknowledgements: This work was supported by the European Commission, the Artemis-JU, and by VINNOVA under Grants 2012-04304 Crystal, 2013-2196 Synligare, and 2014-01271 Amalthea4public.