Model-based Consistency Checks of Electric and Electronic Architectures against Requirements

Nico Adler[1], Philipp Graf[1], and Klaus D. Müller-Glaser[2]

[1] FZI Research Center for Information Technology, Karlsruhe, Germany
{adler,graf}@fzi.de
[2] Institute for Information Processing Technology, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
klaus.mueller-glaser@kit.edu

Abstract. The electric and electronic architecture (EEA), which is built up during the concept phase of automotive electronics development, has fundamental impact on the success of a vehicle under development. The complexity of upcoming architectures requires novel approaches to support system architects during the design phase.
This paper describes a model-based generic approach which allows verifying an EEA with regard to its requirements by using techniques of consistency checks during an early design phase. This includes handling of incomplete models. In this case it offers the possibility to automate consistency checks and in future work facilitate an automatism for optimization and design space exploration to check different realization alternatives of an initial EEA. Automatic report generation of results serves for documentation.

Keywords: electric and electronic architectures, model-based engineering, automotive, verification, requirements

1 Introduction

Electric and electronic architectures (EEA) in the automotive and avionic domain build a complex network of a multitude of embedded systems. In the automotive domain we already see an amount of up to 70 networked electronic control units (ECUs) in current upper class vehicles [1, 2]. Various innovations, e.g. driver assistance systems, and new technologies, e.g. FlexRay, make the EEA of vehicles more complex, including numerous technical and functional aspects. Driven by customer demands for more safety, comfort and infotainment, this bears growing challenges for upcoming development activities [3].

The Original Equipment Manufacturer (OEM) must find new ways to control and manage the rising complexity of an EEA. In addition different variants of the EEA, as a single product can have several equipment concepts, increase the difficulty to analyze, whether an EEA meets all requirements [4, 5]. Moreover new standards like ISO26262 [6] increase quality requirements and efforts to verify an EEA regarding functional safety.

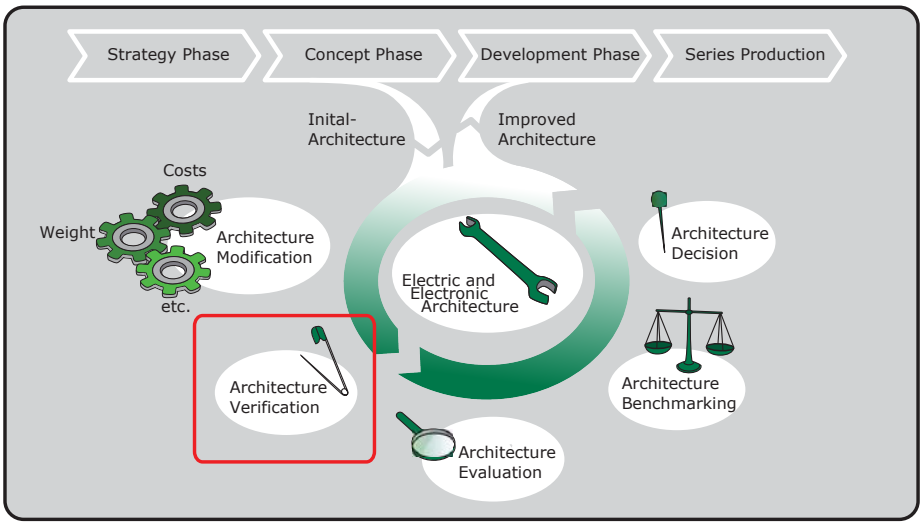

Fig. 1. Simplified product life cycle and EEA development cycle [7]

Within the product life cycle the *concept phase* deals with functionality, convenience, risk and profitability of the vehicle [8]. The result must be a concept that meets the main requirements of a vehicle. During the *development phase* the OEM implements the variants of the solution from the *concept phase* into prototypes. This deals with the technical feasibility of alternative solutions. Further phases follow and are out of the scope of this work.

During the *concept phase* about 80 percent of lifecycle costs will be determined, although the *concept phase* itself is only about 6 percent of the total incurred costs [9, 10]. After finalization of the *concept phase* subsequent changes in the EEA are either associated with enormous costs or may not be feasible anymore. Inconsistencies, which cannot be eliminated, in the worst can put the project's success in risk. Therefore, it is a necessity to perform optimizations and verification already during the *concept phase*.

A simplified iterative process for developing an EEA can be described using a cycle with five steps, as shown in Fig.1. First of all, during *architecture modification*, system architects try to improve an existing initial EEA concerning different criteria. This can be done by optimization or by design space exploration. In the second step, the *architecture verification*, they must prove if the modified architecture meets all requirements. This is a very challenging and time-consuming task and can be accomplished using consistency checks. Subsequently the EEA has to be evaluated. This can be done using a cost breakdown structure to get the total system costs with regard to product lifecycle [11]. To find the most acceptable solution and to achieve an *architecture decision*, different realization alternatives are benchmarked.

The development process for vehicle architectures is usually spread over several departments. This results in difficulties for global design decisions and especially in the proof of overall consistency of an EEA. The objective is to verify and demonstrate in an early stage, which (sub-) areas present or could present inconsistencies against requirements, so that measure can be taken to solve them.

This paper focuses on a concept for automatically verifying an EEA during the concept phase regarding requirements and specifications and is organized as follows: The next section briefly defines verification and gives an overview of model-based domain specific languages for EEA. Section 3 briefly relates verification to model-based engineering. The following two sections present our methodology for verification of model-based EEA. A discussion of the adaption to a domain specific toolset for a first prototype is given in Section 6. The final section gives a conclusion and presents future work.

2 Related Work

Verification of an EEA is one of the significant points during development. Boehm described the basic objectives of verification and validation (V&V) early in the product life cycle with identification and solving of problems and high risk issues [12]. The V-Model-XT V1.3 describes: 'verification is to ensure that selected work results meet their requirements'. Therefore a definition of verification procedures and setting up the necessary verification environment must be done [13].

Electric and electronic architecture modeling supports system architects as models abstract complex problems. Different approaches and projects for the model-based description of especially automotive EEA exist, e.g. the project Titus [14], the language EAST-ADL [15], which emerged from the project EAST-EEA, and the EAST-ADL2 [16], which emerged from the follow-up project, and AUTOSAR [17].

Another model-based approach for the description of EEA is the 'Electric Electronic Architecture - Analysis Design Language' (EEA-ADL) [18, 19]. This data model also forms the basis of the architecture modeling and analysis tool PREEvision [4]. It combines the previously presented approaches within a tool and was used for the following described implementation. PREEvision Version 3.1 provides seven abstraction layers. *Requirements* and *Feature-Functionality-Network* constitute the first abstraction layer. Artifacts of this layer are text elements, which represent atomic features or requirements to the architecture. The underlying layers are: *logical architecture*, *function network*, *component architecture with network topology*, *electric circuit*, *wiring harness* and the *geometrical topology*. Cross layer links between model artifacts can be modeled using mappings. Apart from modeling EE relevant content, the EEA-ADL provides the opportunity to deposit attributes for costs, weight etc. This makes the model suitable to apply metrics and perform architecture evaluations. For analysis of an EEA an integrated, graphically notated metric framework can be used [20]. To perform consistency checks an integrated consistency rule model editor is given.

Rules for consistency checks can be modeled in a graphical way. Simple rules for checks can be set up easily, but for complex ones with lots of constraints, inputs and multiple involved abstraction layers, this approach is too limited. Another disadvantage of the provided rule modeling is that only the boolean logic operator AND can be used. Therefore, the validness of multiple solutions is hard to model. Consistency checks have to be generated within the consistency rule model and afterwards synchronized with the architecture model. This is very time-consuming, especially for building up and testing complex rules. The created consistency checks can be started only manually. Therefore it is not straightforward to use this methodology for a desired semi-automatism for optimizing an EEA as the checks must be started individually depending on the actual optimization state.

Some of the presented domain specific languages offer an import and export of requirements from requirements management tools like IBM Rational DOORS[3]. The combination of these tools offers to map requirements within the model-based domain specific language to the corresponding artifacts and consistency checks.

3 Checking EEA-Consistency in the Context of Model-Based Development

The quality of verification for EEA models is only as good as requirements are described, the according consistency checks are derived from these and realized as executable rules. To do this, it is mandatory that a consistent model exists, which ensures that all EE relevant data is available and up-to-date, including requirements. During the concept phase with consideration of model-based engineering, models are not complete and different sub-parts exist at different detail levels. Therefore model-based verification of constraints has as an additional requirement to secure that verification rules also work with incomplete models. Also, incomplete requirements and specifications either from OEM or supplier have to be taken into account. However absolute maximum ratings of datasheets or specifications can be precalculated or estimated from previous series to perform first consistency checks.

Inconsistency at any point shall not abort verifying as any information about existing consistencies and inconsistencies are beneficial. With automated reporting the results must be captured so that during the development process different versions can be traced and reproduced. Guidelines or regulations of standards must be used as a basis for documentation templates. This is, for example, requested by ISO26262.

It has to be ensured that consistency checks do not apply changes to the EEA data model. The deposited rules are only allowed to retrieve data out of an immutable model.

[3] http://www.ibm.com/software/awdtools/doors, 2011.

4 Overview of the Constraint Verification Approach

Starting point for model-based verification is a EEA data model, which is filled with any available and relevant information, as shown in Fig.2 on the left hand side. This can be done using a model-based domain specific language such as the examples presented in Section 2.

Within this model the EEA is specified and requirements must be created or imported from other tools. The requirements linked to the EEA data model are the corner stones for the verification. For the requirements layer textual described and hierarchical constructed requirements are suggested. To structure requirements, requirements packets should be used. Layer internal mappings are used for building up a network between requirements and also requirements packets. In further steps requirements can be mapped to the corresponding consistency checks.

For automatic verification, five different functional blocks, as shown in Fig.2 on the right side, are provided and are described in the following sub-sections: *consistency check blocks* with the checking rules for verification, *model query blocks* concerning data acquisition, *control unit*, *requirement block* and *report generation block* for documentation.

4.1 Consistency Check Block

From each requirement or requirements packet at least one consistency check must be derived and implemented as an executable rule. As a decision criterion, the complexity to check the requirement or requirements packet can be used. Therefore the architect has to analyze all requirements separately.

Verification can relate to different abstraction layers in an EEA model. For consistency checks, the corresponding artifacts, including their attributes and their mappings between different abstraction layers, are required. This input data for the consistency check can be provided by outsourced data acquisition using model queries and is described in the following subsection.

The multitude of incoming model data from the model queries has to be preprocessed. This includes sorting and filtering data, furthermore structuring of model data which belongs together. Another task is to capture the required corresponding attributes of model artifacts.

The crucial point to verify the EEA is the execution of the consistency check. This is composed of a sequence and examination rule. In a first step the sequence is partitioned into *consistency checks that cannot be performed* and *consistency checks that can be performed*. It is advisable to inspect if all relevant data for performing the check are available. If any relevant information is not available, the consistency check cannot be performed.

At this point results of not possible checks must be collected and stored for postprocessing and preparation for reporting. If the consistency check can be performed, the examination rule describes what shall be checked and in fact represents the derived requirement or requirements packet. Instructions for checking

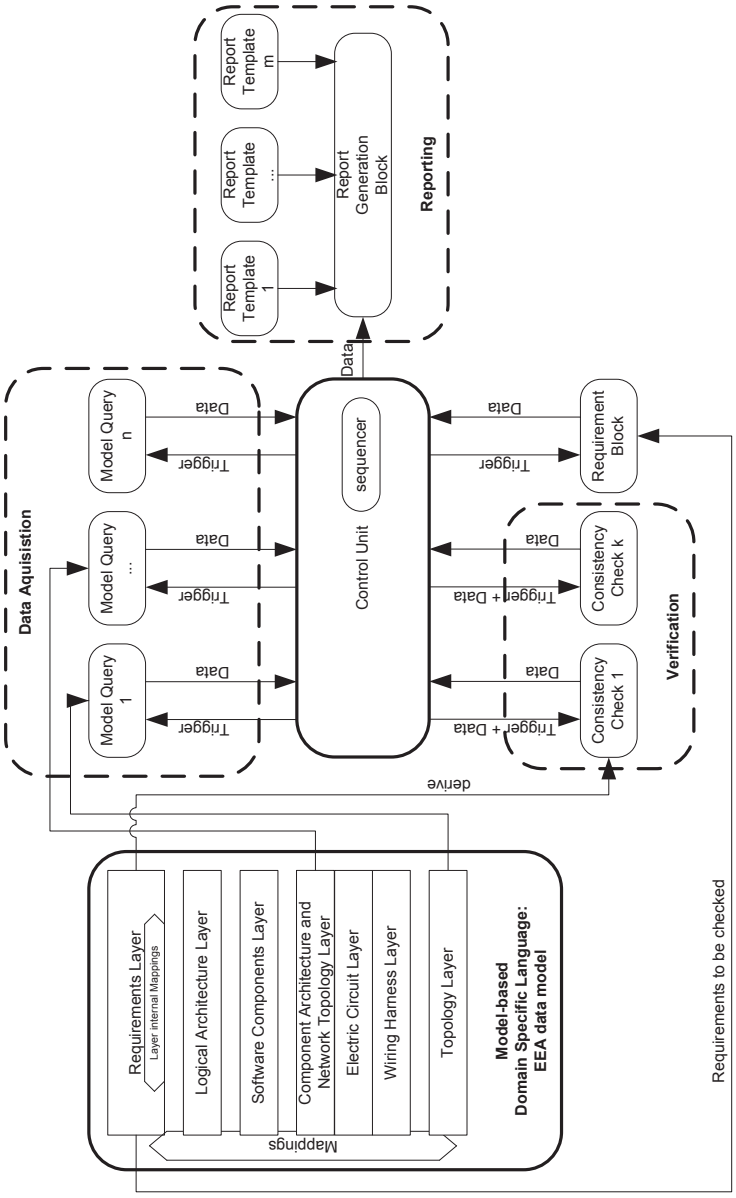

Fig. 2. Approach for verification of EEA against requirements

on inconsistency or consistency can be implemented using a specific algorithm. The algorithm examines the available data and executes the rule.

Results of the consistency check are lists or tables. Additionally the degree of a possible violation of a rule can be estimated. Subsequent post-processing is used mainly for preparing the data in a structured way for reporting.

4.2 Model Queries Block

Expert knowledge demonstrates that large parts of verification consists of data acquisition and structuring. Model queries can be used for data acquisition and should be based on the corresponding EEA meta-model. This ensures that model queries are correct.

Different consistency checks may require the same model artifacts. The reuse of existing model queries can significantly reduce the overhead of setting up new checks. Hereby model queries can be used for different consistency checks which need equal model artifacts, or the artifacts are part of the model query result and must be extracted by the consistency check block. Therefore, an approach has to be found to avoid redundancy so that model queries are implemented only once. To provide results of model queries to different consistency checks, the model queries must be connected to a *control unit*. The *control unit* forwards the model query results to the corresponding consistency checks.

4.3 Control Unit

The *control unit* is the central block. It acts as a sequencer and is connected to all other blocks by input and/or output ports. The ports are used to transmit different kind of data. Two modes are differentiated: *trigger mode* is used for data request and *data mode* is used for transmission of data consisting of header and payload. For both port modes, it is possible to directly connect an input port to one or more output ports. Also incoming triggers or data can be split and/or combined.

Data preprocessing for paths coming from model queries to consistency checks is not provided, because preprocessing can distort results with possible data losses. To avoid this, preprocessing of model query results has to be implemented directly in consistency checks. However data preprocessing is applied for the path coming from consistency check output ports to report generation. Therefore the *control unit* is further connected with the report engine. After final preparation of verification results, the *control unit* transfers data together with report filename to the report generation block.

The subsequent execution of the verification is to be started by the *control unit*. Thus the *control unit* needs information about which requirements should be verified. For this, a *requirements block* as a further block is incorporated.

4.4 Requirements Block

The *requirements block* is used to insert a collection of the demanded requirements which have to be verified. Therefore the corresponding requirements from

the *requirements layer* are mapped to the block. As data inputs, the *control unit* retrieves them and triggers the derived consistency checks. No mapping means that all consistency checks must be executed.

4.5 Report Generation Block

Reporting for documentation is an important tasks to reproduce and capture results. Reporting can be differentiated into documentation for internal or external use. Therefore, an individual configurable template based approach must be provided, as in different departments they have to fill in different forms or bring a different proof respective to standards or recommended practice.

For internal use, there also can be some kind of documentation, but the key factor is to design an EEA which meets all the requirements as soon as possible. Therefore additional identified information can be documented.

The *control unit* must decide at runtime, which results have to be sent to the *report generation block* to fill the corresponding placeholders in the templates. Result export to other tools must also be performable. For this purpose e.g. a XML Schema Definition (XSD) can be implemented using the templates. This offers a wide range to interface other tools for further processing of results. Therefore the generation of documents shall be delivered through a suitable report engine, which can access the prepared templates.

5 Execution of Verification

The typical sequence for a model-based verification is shown in Fig.3. In the vertical 'swimming lanes' (columns) the five different blocks are presented. Starting point is the *control unit*. The *control unit* requests the *requirement block* for information which requirements should be checked by using a trigger. The bundled requirements are sent to the control unit which selects the model queries to be executed for the corresponding consistency checks. The results in the form of lists or tables are sent back to the *control unit*. The model queries are executed only once, because during verification the EEA data model is not modified. At this point a loop starts. For every requested verification, the *control unit block* bundles the demanded model queries results and sends them to the corresponding *consistency check block*. The verification sequence starts and therefore the examination. Relevant results of the consistency check are sent to the *control unit*, which forwards them after potential preparation for reporting to the *report generation block*.

6 Prototype Implementation in PREEvision

Integrated consistency checks require an integrated model as described in Section 2. PREEvision is a software tool that allows persistent modeling and evaluation from requirements down to topology and was used for the first prototyping the concept described before.

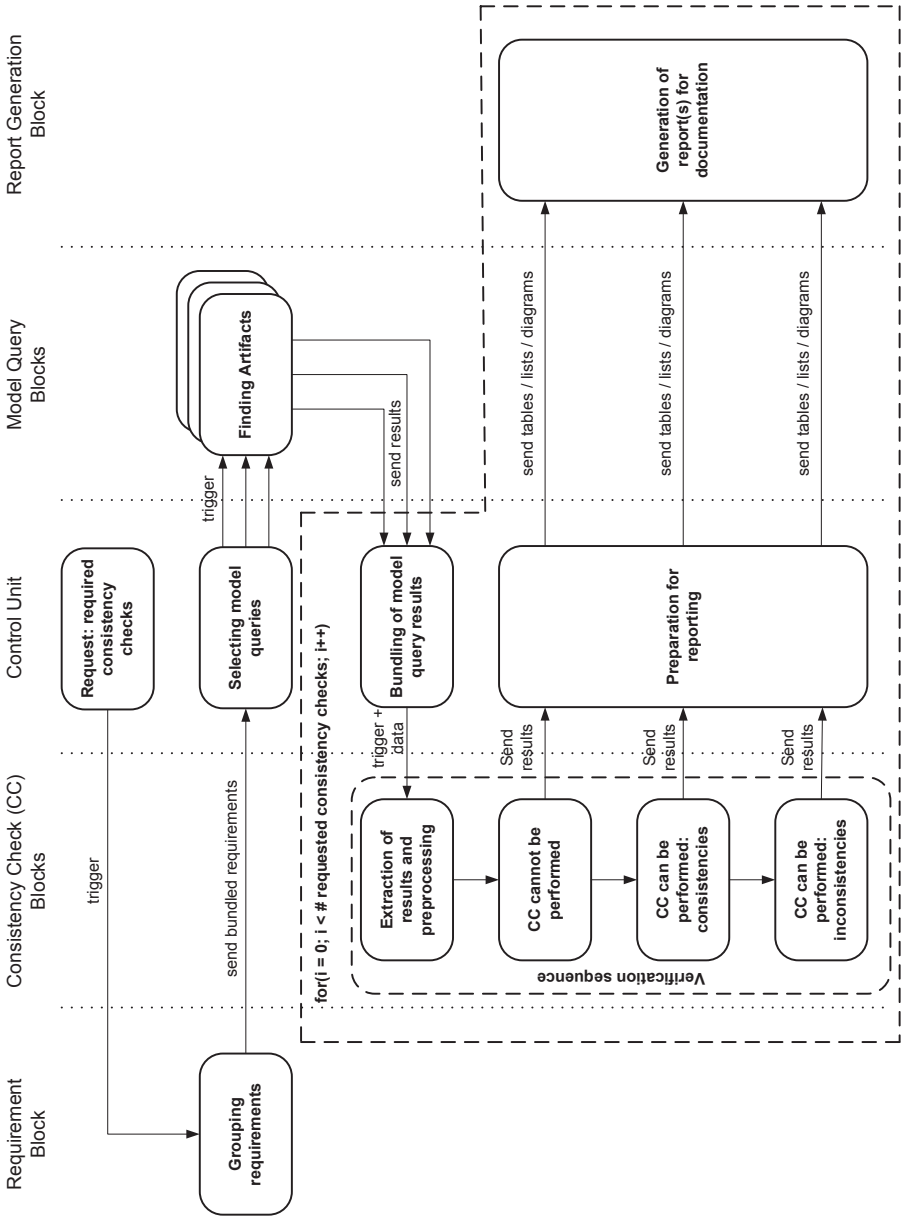

Fig. 3. Execution of verification

As a simple example we demonstrate the approach with a consistency check derived from the requirement 'All ECUs with the attribute *isPartOfActiveVariant = true* and *availability in crash* is set to *low, medium* or *high* must be allocated to corresponding *installation spaces*!'

For preparation we modeled an exemplary EEA within PREEvision. This consisted of *component layer* with *ECU* artifacts, the *topology layer* with *InstallationLocation* artifacts and added *HardwareToTopology-Mappings* between the two abstraction layers. The requirement was inserted in the *requirements layer* of the EEA data model.

6.1 Model Query Blocks

Afterwards we identified which artifacts from different abstraction layers are relevant for performing a consistency check. In this case we needed the involved *ECU* artifacts from the *component layer* including the attribute *availability in crash*. This formed the first model query. Further we needed all *InstallationLocations*. This formed the second model query. The last model query was to find all existing ECU mappings from the component into the *topology layer*.

For the model queries, the integrated rule model in PREEvision was used which is based on the corresponding EEA data model. It allows to model rules in a graphical way. Complex patterns to match can be defined, using not only the *source-object* and its properties, but also objects and *LinkPairs* between the objects, as shown in Fig.4 on the left hand side for the rule diagram *ECUtoInstallationLocationMappings*. A further restriction was added to the example by using one of the *attributes* of the *ECU*, thus the *isPartOfActiveVariant* was set to boolean value *true*.

We generated the rules for the three required model queries, that can be used within the metric framework. The results after execution of the model queries are tables or lists shown on the right hand side in Fig.4. These can be further processed by the consistency checks.

6.2 Consistency Check Blocks

For the *consistency check block* a Java-based calculation block within the metric editor was used. With this approach we access EEA model artifacts and their attributes using Java as a programming language instead of the graphical rule modeling as for the model queries. Being more flexible, it allows to construct simple to very complex consistency checks. The results of the *model query blocks* are used as their input.

It is also possible to construct hierarchical consistency checks using several calculation blocks. For example this can be used to subdivide a requirement or requirements packet into several consistency checks. In this case trigger and data paths have to be looped through the parent *consistency check blocks*.

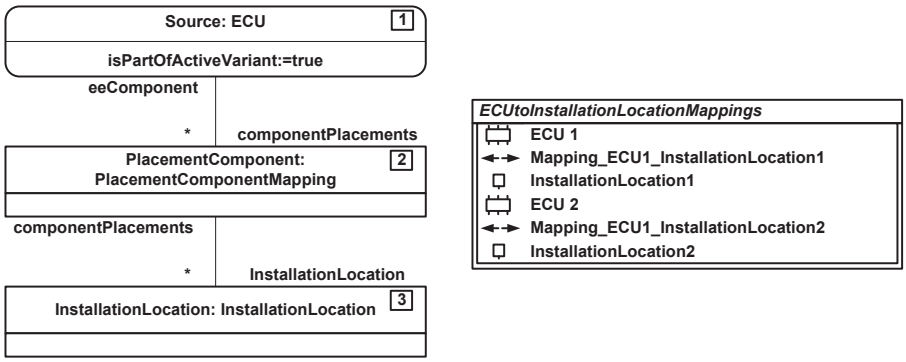

Fig. 4. Rule diagram for the model query (left) and results (right)

6.3 Control Unit

The *control unit* was implemented using a calculation block and was connected with all other blocks using data flows. It contains the allocation table of model queries to the corresponding consistency checks. The execution sequence was implemented and internal trigger and data paths were connected. It is possible to individually improve existing consistency checks or to add new ones. Extension of any kind can be performed easily, as existing model queries, etc. can be reused and only the allocation table in the control unit has to be updated.

6.4 Report Generation Block

For the *report generation block*, we used the open source templating *Apache Velocity Engine*[4], which is integrated in the PREEvision metric framework. Velocity permits to use a simple template language to reference objects defined in Java code. As the output format for the first prototype we chose HTML for the generated files. This allows graphical layout of results and ensures traceability using hyperlinks. Using velocity templates we formed the basic structure of the graphical appearance including a navigation bar. Placeholders in the velocity templates were filled with the data coming from *consistency check blocks*, looped through the *control unit*. To obtain better overview of the identified inconsistencies we export the corresponding diagrams to PNG-file format automatically and include them in the reports.

7 Conclusion and Future Work

In this paper we have presented an approach that makes it possible to automate verification for electric and electronic architectures already during concept phase

[4] http://velocity.apache.org, July 2011

using consistency checks in a model-based way. The methodology is generic and even incomplete models can be checked. This is a significant step to support the system architect concerning reduction of development time and ensures EEA being consistent against requirements. Integrated reporting serves for documentation.

The developed methodology and its implementation in PREEvision has shown to work in our first prototype. An analysis and application of the approach in EAST-ADL is being considered. Also, adapting to an existing standard for expressing constraints, the Object Constraint Language (OCL) which is included in the Unified Modeling Language (UML), will be analysed.

Future work will mainly focus on expanding the approach to architecture evaluation using metrics for calculating quality of the EEA. This ability can be used for benchmarking different EEA realization alternatives. In further steps, the approach can be extended to (semi-) automatic optimization and design space exploration. For generating new EEA realization alternatives a strategy for design space exploration must be found. For this purpose, the automatic verification can deliver useful information about the degree of compliance to requirements for a new generated EEA realization alternative.

Application and evaluation of the approach with a real-world EEA model is planed for a case study, but will require the cooperation with an Original Equipment Manufacturer (OEM) to bring in the real-word application as its intellectual property.

Acknowledgements. This document is based on the SAFE project in the framework of the ITEA2, EUREKA cluster programme 3674. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS11019. The responsibility for the content rests with the authors.

References

[1] Larses, O.: Architecting and Modeling Automotive Embedded Systems. doctoral dissertation. Stockholm (2005)

[2] Reichart, G., Haneberg, M.: Key Drivers for a Future System Architecture in Vehicles. SAE Convergence 2004, Vehicle Electronics to Digital Mobility. Detroit (2004)

[3] Hillenbrand, M., Heinz, M., Adler, N., Müller-Glaser, K.D., Matheis, J., Reichmann, C.: ISO/DIS 26262 in the Context of Electric and Electronic Architecture Modeling. Architecting Critical Systems ISARCS (2010)

[4] aquintos GmbH: PREEvision Version 3.1 Manual, www.aquintos.com. Karlsruhe, Germany (2010)

[5] Burgdorf, F.: Eine kunden- und lebenszyklusorientierte Produktfamilienabsicherung für die Automobilindustrie. doctoral dissertation. Karlsruhe Institute of Technology, Karlsruhe (2010)

[6] International Organization for Standardization: ISO/DIS 26262 Roadvehicles-Functional Safety. Part 1 - 10, www.iso.org, Tech. Rep. (2010)

[7] Adler, N., Gebauer, D., Reichmann, C., Müller-Glaser, K.D.: Modellbasierte Erfassung von Optimierungsaktivitäten als Grundlage zur Systemoptimierung von Elektrik-/Elektronik-Architekturen. 14. Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV) 2011, OFFIS - Institut für Informatik, Oldenburg (2011)

[8] Kuster, J., Huber, E., Lippmann, R., Schmid, A., Schneider, E., Witschi, U., Wüst, R.: Handbuch Projektmanagement. Springer (2008)

[9] Voigt, K.I.: Industrielles Management - Industriebetriebslehre aus prozessorientierter Sicht. Springer-Verlag Berlin Heidelberg (2008)

[10] Bürgel, H.D., Zeller, A.: Controlling kritischer Erfolgsfaktoren in Forschung und Entwicklung. Controlling, Vol. 4 (1997), Nr. 9, S. 218-225

[11] Blanchard, B. S., Fabrycky, W. J.: Systems Engineering and Analysis. Pearson Prentice Hall, New Jersey (2006)

[12] Boehm, B.: Verifying and Validating Software Requirements and Design Specifications. IEEE Softw., Los Alamitos, CA, USA (1984)

[13] V-Modell-XT Version 1.3, Part 7: V-Modell Reference Mapping to Standards. (2009)

[14] Eisenmann, J., Köhn, M., Lanches, P., Müller, A.: Entwurf und Implementierung von Fahrzeugsteuerungsfunktionen auf Basis der TITUS Client/Serverarchitektur. VDI Berichte, Nr. 1374, VDI-Gesellschaft Fahrzeug- und Verkehrssicherheit, Systemengineering in der Kfz-Entwicklung, (1997)

[15] The East-EEA Project: Definition of language for automotive embedded electronic architecture approach. Technical Report, ITEA, Deliverable D3.6, (2004)

[16] The ATESST Consortium: EAST ADL 2.0 specification. Technischer Bericht, ITEA, (2007), http://www.atesst.org/

[17] AUTOSAR, Automotive Open System Architecture. http://www.autosar.org, (2010)

[18] Matheis, J.: Abstraktionsebenenübergreifende Darstellung von Elektrik/Elektronik-Architekturen in Kraftfahrzeugen zur Ableitung von Sicherheitszielen nach ISO 26262. doctoral dissertation. Karlsruhe Institute of Technology, Karlsruhe (2010), ISBN: 978-3-8322-8968-3

[19] Belschner, R., Freess, J., Mroko, M.: Gesamtheitlicher Entwicklungsansatz für Entwurf, Dokumentation und Bewertung von E/E Architekturen. VDI Bericht, Nr. 1907, S. 511-521, VDI-Verlag, Düsseldorf (2005).

[20] Gebauer, D., Matheis, J., Kühl, M., Müller-Glaser, K.D,: Integrierter, graphisch notierter Ansatz zur Bewertung von Elektrik/Elektronik- Architekturen im Fahrzeug. HDT (Haus der Technik), (2009)