

How to Engineer Tool-Chains for Automotive E/E Architectures?

Peter Waszecki
TUM CREATE
Center for Electromobility
Singapore
peter.waszecki@tum-
create.edu.sg

Martin Lukasiwycz
TUM CREATE
Center for Electromobility
Singapore
martin.lukasiwycz@tum-
create.edu.sg

Alejandro Masrur
Department of
Computer Science
TU Chemnitz, Germany
a.masrur@informatik.tu-
chemnitz.de

Samarjit Chakraborty
Institute for Real-Time
Computer Systems
TU Munich, Germany
samarjit@tum.de

ABSTRACT

The growing demand for both safety and comfort functionality in modern vehicles is rapidly increasing the complexity of automotive Electrical/Electronic (E/E) architectures. This makes it necessary to use specific design tools for modeling, implementing, and testing such systems. Although many tool vendors offer products covering different design and development phases, it is still cumbersome for car manufacturers to find the most appropriate tools that support their particular E/E architecture design process. And even with the proper set of tools, it remains an essential challenge to combine them into a consistent and flexible tool-chain, covering all design and development phases. This work is a first attempt to develop systematic approaches towards building such tool-chains. To this end, we provide an overview of the design process of E/E automotive architectures, its shortcomings and challenges, and study 22 possible tools currently available on the market. Based on this study, we quantify various usability and functionality aspects of the tools and use the outcome to evaluate their compatibility in terms of forming a tool-chain.

Keywords

Automotive, E/E architectures, Design tools, AUTOSAR.

1. DESIGNING AUTOMOTIVE E/E ARCHITECTURES

Nowadays, the major part of innovations in modern cars are introduced by Electrical/Electronic (E/E) architectures and the applications implemented upon them. Although, notable developments and new technologies in terms of batteries and engines are strongly pursued in the context of hybrid and electric vehicles, it is still expected that E/E architectures here become the main driver of innovation [1] [2]. Modern E/E architectures consist of more than 100 Electronic Control Units (ECUs) [3] and various bus systems like Controller Area Network (CAN), Local Interconnect Network (LIN), and FlexRay, operating in different domains and connected via one or more gateways [4]. This system complexity mainly stems from the increasing demand for driver assistance functions, providing both improved safety and more comfort for occupants. In this context, the Anti-lock Braking System (ABS), Electronic Stability Control (ESC) or Adaptive Cruise Control (ACC) represent

some of the most common functions. It is anticipated that implementation of novel drive-by-wire systems particularly in electric vehicles and enhanced Human Machine Interaction (HMI) applications will additionally increase the complexity of E/E architectures. Hence, the design of such architectures cannot be covered by a manual workflow anymore but rather requires a sophisticated tool-chain, for modeling, early-stage verification and validation, and testing of the system [5].

As a result, automotive companies and suppliers are showing an increasing interest in system-level modeling tools that cope with a growing complexity and harder requirements and, at the same time, simplify and improve the design process. In the past few years, a large number of design tools have emerged which are progressively being adopted in industry. However, with the growing requirements on E/E architectures on the one hand and the increasing number of possible design tools on the other hand, the selection of suitable tools is becoming a major challenge for car manufacturers (both, the Original Equipment Manufacturers (OEMs) and Tier 1 suppliers). It is not possible to choose one single tool for the entire design process since such a versatile tool simply does not exist. The complexity of E/E architectures rather requires the combined use of diverse model-based tools from various vendors, covering different development stages. However, while each tool might work well within its own scope of functionality, a tool integration into a flexible and consistent tool-chain is still a major challenge. Among others, problems arise from insufficient interfaces and error-prone manual, rather than automatic, input of essential design data [6].

This paper provides an overview of available tools and guidelines to help researchers, developers, and engineers in the automotive domain in the selection of a proper tool-chain. We describe how to narrow down from a high number of design tools, how to create a good summary of their most essential properties and, finally, how to choose the most appropriate ones in order to build a tool-chain. As a case study, we have surveyed 22 established and emerging tools that are commonly encountered in the automotive domain. Furthermore, based on several graphs and metrics, we quantitatively evaluate their utility at different design and development phases. However, since a particular composition of metrics strongly depends on companies' requirements a company-internal

evaluation of the tools is still necessary. The presented method allows significantly reducing the number of candidate tools and saving time and costs during the decision making process. To the best of our knowledge, this is the first systematic effort to evaluate such design tools. Although we specifically focus on the automotive domain, the basic principles proposed in this work are helpful in other domains as well (e.g., in the electronic design automation), where this is also a common problem.

1.1 State-of-the-Art in E/E Architecture Design

Usually, specific vehicle components like ECUs or bus systems are designed and provided by suppliers like Continental, Bosch, or Delphi rather than by OEMs themselves. These suppliers, on their part, rely on chip manufacturers, for example Freescale or Infineon. In this context, the main task of OEMs, like Volkswagen, BMW, GM, Ford or Toyota, is the specification and engineering of requirements. OEMs are also responsible for the integration of ECUs and bus systems into their specific car models and for performing extensive testing of the whole set-up. In a nutshell, the different roles of manufacturers and suppliers in the automotive industry are distributed among three groups. The first and closest to the end product are the OEMs whose tasks are the requirements specification and engineering, integration of systems and functions as well as validation of the whole E/E architecture [7]. The second group are the *Tier 1* suppliers, who are responsible for the development of ECUs and application software. Finally, the *Tier 2* suppliers deliver the necessary hardware (e.g., microcontrollers and memory chips) as well as basic software such as operating systems.

In this supply chain, besides OEMs, particularly the Tier 1 suppliers are involved in the system-level design of E/E architectures. This process embraces different development stages or phases for which various tools exist. We first classify these tools into five categories with each category covering a different design phase. Table 1 gives an overview of these design phases and the corresponding tool categories. Note, that this comparison provides only a rough overview.

For ease of exposition, the design phases in Table 1 can be transferred to a V-model [8] which is commonly used in software and systems development to describe the consecutive steps in a development life cycle. Named after its V-shaped shape, the model depicts the chronological evolution of a system beginning with its general specification and modeling on the left-hand branch down to the implementation and up again to test, verification, and validation on the right-hand branch.

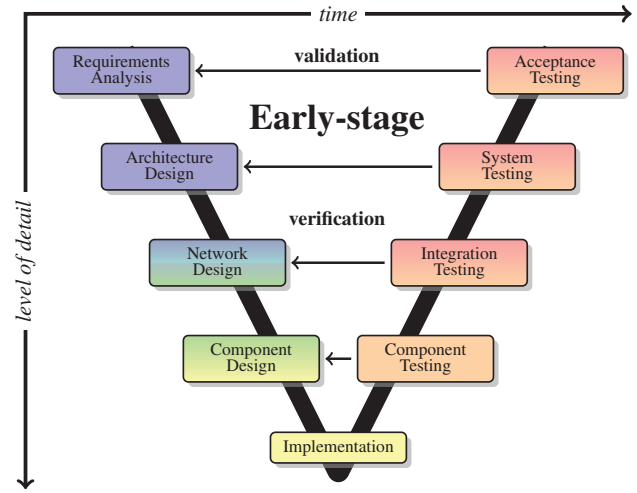


Figure 1: System development life cycle within the V-model. The five design phases are highlighted by their colors according to Table 1.

In this respect, the amount of system detail increases on both branches towards the bottom of the V-model. Figure 1 illustrates a V-model describing the development process of an automotive system in nine steps reflecting the design phases from Table 1 in different colors. For practical reasons, this allocation is fuzzy and, hence, the design phases might exceed the boundaries of single V-model steps.

For more complex projects, the V-models allows the insertion of horizontal connections between development steps at the same level (e.g., *system architecture design* and *system testing*). They represent an early-stage verification of the three lower V-model levels and an early-stage validation for the topmost level. Although, there exist more than a single possible representation of the V-model which may differ in the number and naming of used steps, all of them follow a very similar development cycle.

1.2 Design Phases and Tools

In the following list the main purpose of each design phase is explained together with the resulting challenges for the dedicated tools. Knowledge about these phases is necessary to understand and properly interpret the evaluation of tools in Section 2.

1) *High-level modeling* describes the E/E architecture development, beginning with the definition of system requirements, followed by the modeling of the entire system and partly also encompassing the ECU and network modeling. Consequently, tools operating in this phase should be versatile to allow defining the system requirements and providing an extensive modeling environment to sketch the system in a graphical manner. Besides, additional model analysis mechanisms for early-stage system verification and validation can improve the high-level design process.

2) *Network modeling* focuses on the design and configuration of the network architecture, its components, and the associated data communication. Here, the tools must first provide a comprehensive development environment for designing an overall architecture of the system’s network considering all buses and protocols being used. Second, they should offer component configuration function-

Table 1: Tool categories in different design phases

Stage	Design phase	Tool category
1 (● - blue)	High-level Modeling	High-Level Design
2 (● - green)	Network Modeling	Infrastructure and Configuration
3 (● - yellow)	Synthesis	Simulation and Code Generation
4 (● - orange)	Verification and Validation	Testing and Integration
5 (● - red)	Management	Test and Project Management

ality, for example, in order to specify the timing constraints for the buses and ECUs.

3) *Synthesis* represents the implementation of the actual system functionality and should support the application design by a model-based environment. In order to deploy the application code onto real hardware, efficient code generators supporting a broad variety of microcontroller architectures for different ECUs become necessary. However, since often ECUs may not be available during this development phase, it is important for synthesis tools to provide efficient simulation and emulation functionality to enable an early debugging and verification of the generated code.

4) *Verification and validation* ensures the error-free and requirements compliant operation for both the single components and the entire system architecture. Corresponding tools should provide mechanisms for signal measurement and diagnosis as well as bus and ECU calibration. Furthermore, efficient static and dynamic timing analysis and simulation functionality is necessary to calculate the worst case execution and response times (WCET, WCRT), respectively, for both tasks and messages. These are to ensure that the real-time behavior of the system is reliably verified.

5) In the *management* phase, an acceptance validation of the entire system is performed. In contrast to the verification and validation steps before, here the emphasis is more on legal and safety issues such as compliance with the automotive function safety standards ISO 26262 and IEC 61508. To the best of our knowledge, there is no software used merely for this purpose. Rather, this task is taken over by suitable requirement, test, and project management tools. With respect to the classification within the V-model, these can cover a big part of the right branch and even the requirement analysis stage.

The development of an entire E/E architecture is a highly complex task with many interdependencies. Hence, in order to ensure an efficient and faultless development flow, it is important for the design tools to exchange information between each other, i.e., relevant model-data from one design phase must be passed on to the next one. This data exchange requires suitable interfaces and exchange standards and ideally enables the tools to be combined to a consistent tool-chain. However, as for each design phase already multiple tools exist which, in addition, often lack appropriate interfaces, the selection of proper tools is difficult. To overcome this problem, in this paper we describe a systematic approach for tool selection.

1.3 Challenges in the Near Future

Without influencing the general purpose of the tools as introduced, future E/E architectures, particularly in the area of hybrid and electric vehicles, will demand some essential paradigm shifts concerning the functionality of design tools. Current E/E architectures are mainly event-driven and rely on event-triggered communication and operating systems like CAN buses and OSEK/VDX¹, respectively. The inherent non-determinism of event-triggered architectures leads to time-consuming simulation and testing during the design process. Although important simulation techniques, like software in the loop (SiL) and hardware in the loop (HiL), allow us to emulate the input and output of a system, its testing remains very time and cost intensive. To prevent expensive re-testing after changing system design parameters, an incremental design flow is

¹ Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen (English: Open Systems and their Interfaces for the Electronics in Motor Vehicles) is a specification for an embedded operating system.

required where tools support formal methods, automatic test case generation, and verification of single design steps.

However, as the complexity of information and communication and the number of safety-critical functions in next-generation cars will increase, time-triggered control and communication buses will become inevitable. A prominent example for this paradigm change is FlexRay [9] which, besides a time-triggered communication, provides a high data rate and fault-tolerance characteristics to meet the latest requirements on safety and performance particularly in the context of drive-by-wire systems [10]. Furthermore, current automotive software already exceeds 100 million lines of code and it is expectable that this number will double or triple in the near future [11]. Thus, the growing complexity in terms of software, hardware, and the associated wiring harness, component weight, and manufacturing costs will necessarily lead to a general re-design of today's E/E architectures. This change can be achieved either by an extension of existing architectures through Domain Control Units (DCUs) that provide the main functional software for each specific vehicle domain [12] or by the more fundamental consolidation of ECUs. The latter will combine the functionality of multiple single-core ECUs on one multi-core ECU and, in turn, results in a reduction of the entire E/E architecture complexity in terms of hardware and wiring [13] [14]. This approach is often referred to as a transition from *federated* to *integrated* automotive architectures [15].

To support this development, two basic requirements must be fulfilled. First, system-level tools must enable abstracting functionality into models from which code is automatically generated and provide powerful connectivity to other tools for simulation and testing. Such a model-based design offers the possibility to partly perform the verification and validation at an early design stage [16] as illustrated in Figure 1. Second, to enable design optimization for systems from different suppliers, the generated code must be based on non-proprietary, open software design standards which allow a hardware-independent development and distribution of functions. In this context, the Automotive Open System Architecture (AUTOSAR) plays a major role in providing an abstraction layer separating open software standards for the integration of subsystems from proprietary applications provided by Tier 1 suppliers and OEMs [17].

2. EVALUATING AND SELECTING ARCHITECTURE DESIGN TOOLS

So far, we presented the design process for E/E architectures and its challenges, highlighting the importance of appropriate design tools and tool-chains. This section begins with the description of an overall approach which should lead to an appropriate choice of the design tools and the corresponding tool-chain. To illustrate the proposed procedure, we have chosen 22 available design tools in order to qualitatively investigate their properties and suitability, see Section 2.2. This investigation serves as a basis for a detailed quantitative evaluation that is described in Sections 2.3 to 2.5. So as not to impede the reading flow, all referenced figures and tables are located at the end of the paper.

2.1 An Approach For a Proper Design Tools Selection

An investigation of the model-based design and optimization of E/E architectures by applying several *keyfigures* (i.e. metrics) to them was presented in [6]. This was done in an abstract manner, i.e., without considering any available tools, and supported only by

one case study. In our opinion, such an analysis is much more useful for the automotive industry, if a number of existing architecture design tools are taken into account. In fact, OEMs are mainly interested in having a selection of available software products combined into a consistent tool-chain, accompanying the entire E/E architecture design life-cycle.

We believe that the selection process of design and development tools must be carried out in a systematic manner and should begin with obtaining a detailed overview of all possible tools. That includes an investigation of the required tool coverage for given design phases according to the V-model as well as their support for specific automotive standards. This step may already rule out tools with insufficient coverage of initial requirements. Furthermore, a detailed metric-based evaluation of the remaining tools will leave a manageable selection of products which ideally should be followed by a real-world testing by architecture designers and engineers. Here, *metrics* define the measures of selected tool properties in terms of functionality, quality or availability and, hence, facilitate a quantitative assessment. The latter step serves both gaining practical experience with some products and analyzing the necessary interfaces for connecting different tools from different vendors. Finally, an additional chart visualizing the possible tool interconnectivity will help selecting the most appropriate products from the OEM's perspective. As the metrics can be chosen according to the particular application domain, the presented selection strategy can be adopted for the general E/E architecture development as well as for specialized E/E design tasks.

2.2 Qualitative Survey of E/E Architecture Design Tools

As mentioned above, we investigated 22 design tools for all five design stages, considering different evaluation criteria. Table 2 lists these tools together with their corresponding vendor name (in brackets) and gives a brief description of the main functionality of each product. The survey was partly based on publicly available information (e.g., data that are freely accessible on the Internet) and partly on our own experience with several of these tools. Due to the, in some cases, only incomplete information provided by tool vendors and the continuous and fast changes in the automotive domain, this paper does not claim to cover all features and properties of the presented products. Nonetheless, this is not a drawback for our selection approach as we do not simply provide a comparison and rating of a number of selected products. The main goal of this work is rather to present a reliable strategy for choosing the most suitable tools in order to create an E/E architecture design tool-chain.

2.3 Tool Coverage and Compatibility

Following the steps described in Section 2.1, before performing the actual metric-based analysis of the tools, it is necessary to determine the following two aspects:

- coverage of the V-model steps, and
- compatibility with standards and protocols.

The first aspect is concerned with the tools' scope according to the nine design steps of the V-model, whereas the second aspect refers to their compliance with important automotive standards.

For the coverage aspect, we have extracted the information about the application area of the tools and visualized their position within the V-model. The result is depicted in the *Coverage Graph* in Figure 2 where each column represents one design step. By covering some of these columns, the horizontal bars illustrate the tools' belonging to different design steps. More precisely, a dark color shade

signifies that the particular design step is fully supported and a light color shade signifies a partial or a not fully determinable support. Each design step needs specific functionality (e.g., requirements definition in step one, code generation in step five or bus calibration in step seven), hence, the degree of coverage of a particular design step has been extracted from both the tools' specification or manual and our own experience with some tools. When analyzing the graph in Figure 2 we can recognize that the four high-level tools and three network tools have a good coverage within the left branch, which renders them eligible for the first two design phases. The same applies to the synthesis tools in the third design phase located at the bottom of the V-model. However, looking at the fourth and fifth design phases, the gaps in the particular columns as well as the light shaded bars give us a hint on insufficient verification and validation functionality within the investigated tools. Note that *RPLAN e3* is represented by a full-length white bar to emphasize its exceptional role as, a more or less, pure project management tool providing secondary functionality for all design steps. Based on this analysis, a possible tool-chain could contain *PREEvision*, *Network Architect*, *SIMTOOLS*, *CANx* and *DOORS*. However, as we are considering the coverage of design phases only and the last two design steps are not fully supported these example selection is not ideal.

As listed above, the second aspect which has to be determined is the tools' ability to deal with specific automotive software standards, bus protocols, and data exchange formats. This knowledge helps us to identify and optimize the degree of compatibility between design tools. In this context, the *Compliance Information* in Table 3 lists the tools' support of essential automotive standards. Here, for example, we can see that a broad use of AUTOSAR is mitigated by the lack of bus protocol and exchange format implementations particularly for the high-level and synthesis tools.






2.4 Tool Metrics

The next step towards the selection of design tools is a thorough evaluation, enabling a significant quantitative comparison of the tools' essential functionality and their general properties. Therefore, we have defined a number of criteria and assigned weighted multi-level metrics to them. The development of these metrics is described below. First, let us note the outcome of the quantitative analysis of our specific tool investigation which is depicted as a *Feature Metrics Matrix* in Figure 3. On the top of the chart we can see four categories that represent the general metrics of automotive design tools:

- *Functionality*, comprising the most essential functional properties,
- *Compatibility*, comprising the main interconnection and compliance features,
- *Usability*, comprising the user experience,
- *Distribution and Actuality*, comprising the market penetration and product release information.

The importance of the proper *Functionality* of a design tool is self-evident whereas *Compatibility* was already discussed in Section 2.3. By adding *Usability* as well as *Distribution and Actuality* to the evaluation, we can characterize the most essential aspects of the design tools. These four categories are refined into different criteria, which are listed on the horizontal axis below. We have decided to choose six features for the *Functionality* segment, as it

has the highest impact on the tool selection process. The remaining three categories have been split into three criteria each. Within a row, the support of a particular tool for a specific metric is visualized by a small pie-chart. As the degree of support might be an insufficient measure for some criteria (e.g., market penetration, latest release, etc.), in addition to the support level, each pie-chart is given an alternative interpretation, as listed below.

-  - The support for a given feature is very strong or it is one of the tool's main features. Alternatively, a feature indicates a good penetration or a short time range.
-  - The support for a given feature is sufficient but may contain gaps and weaknesses. Alternatively, a feature indicates a moderate penetration or a moderate time range.
-  - The support for a given feature is only rudimentary and its implementation will most probably not fulfill the user's needs. Alternatively, a feature indicates a poor penetration or a long time range.
-  - The feature is not provided by the tool or does not fall within the tool's scope.
-  - The support for a given feature could not be determined.

The last column in each category depicts the weighted average of its preceding values and the same applies to the overall average in the rightmost column. In contrast to the specific metrics with four different values, the more fine-grained average results are additionally shown in their numerical representation between 0 and 1. It should be mentioned, that in case of data which cannot be interpreted unambiguously (represented by gray circles) the worst case is considered which equals a value of 0. Additionally, weight factors assigned to each metric allow a more specific distribution of their relative significance and are reflected in the average values. To highlight more important functions, in the *Functionality* category we have chosen the metric weights for system modeling (30%), bus support (20%), and simulation (20%) with higher weight factors than for model analysis (10%), code generation (10%), and timing analysis (10%). The criteria for *Compatibility* and *Functionality* are approximately equally weighted, whereas the *Distribution* category puts a higher emphasis on market penetration (50%) than on the last release (20%) and the release interval (30%).

Considering the average values for each category as well as the overall average, we can gain a rough estimate of which tools are more applicable compared to others within the same design phase. For instance, for high-level modeling, *SystemDesk* has the best overall average value, whereas *Vehicle System Architect* seems to outperform it when considering functionality only. Obviously, it is important to take a more accurate look at the single evaluated criteria which are weighted according to their importance given by designers. Thus, we can filter out tools with a strong average rating but lacking properties or functionality which are paramount for a particular development process defined by the OEM.

As some features are more tailored to some design phases than to others, comparing average values from different design phases might be misleading. This is particularly the case when looking at the overall average values that combine data from all four categories. To overcome this problem, it is recommendable to compare tools within particular tool categories, as depicted in the *Feature Metrics Matrix* (Figure 3). However, as some tools can cover more

than one category and, hence, contain functionality from other design phases, one should also take the *Coverage Graph* (Figure 2) into account when comparing tools by means of their overall average values.

When analyzing the results presented Figure 3, the lowest scattering of average values can be seen in the High-Level Modeling phase. This indicates a general similarity between the corresponding tools in terms of their features. For the other phases the difference in the average value seems to be more significant, thus, simplifying the tool selection process. According to the *Feature Metrics Matrix* the five most suitable tools are *SystemDesk*, *EB tresos Inspector*, *MATLAB/Simulink*, *CANx* and *DOORS*.

It is important to bear in mind that metrics strongly depend on the specific needs and requirements of the OEM. Thus, our selection of metrics as well as the choice of weight factors are a special case and the *Feature Metrics Matrix* shall support a previous qualitative tool analysis. Additionally, in this work it also shall give an idea of how to apply and visualize customized metrics during the selection and evaluation of design tools. In the end, car manufacturers can hardly avoid testing some of the tools to verify their functionality and the actual usability. Nevertheless, applying the quantitative metric-based evaluation can reduce the initial number of tools to be regarded in a significant manner.

2.5 Tool Interfacing

Finally, knowing the tools' coverage of the design phases and their functionality, features and shortcomings according to the different metrics, we can investigate the possibility of building a consistent tool-chain. As it may be the case, designers might possess practical experience with some of the selected tools which can be used for the interface investigation. The information about the connectivity between single tools can be visualized by links inside a *Interface Graph*, as depicted in Figure 4.

The results show an overview of possible interfaces between tools without describing their technical details, orientation, and quality, which means that the links might constitute natively supported interfaces, open exchange formats as well as in-house or third-party plug-ins. Besides showing the bare interfaces, this graph can also be used to visualize additional quality information by graphically emphasizing tools and links that have more suitable interfaces and a higher overall rating from the metric-based evaluation.

Regarding the results in Figure 4, the tight net of connections suggests an overall good interoperability. However, by taking a closer look, we can recognize that apart from the high-level design domains, many tools are not linked at all. And even when disregarding some less important interconnection groups (for example, the management tools), obtaining a flexible and consistent tool-chain seems to be a difficult task at the moment.

The results of the *Interface Graph* are mainly based on the communication and data exchange standards as described in Section 2.3 and listed in Table 3. Beyond the evaluation of provided tool interfaces, a thorough analysis of the general compatibility between tools is necessary. This analysis is important for tools that belong to different design phases, since this might reveal additional valuable information for the *Interface Graph* and help both obtaining a proper tool-chain and performing detailed pairwise tests to evaluate mutual compatibility between the tools. However, such an analysis is very time-consuming and goes beyond the scope of this work.

Eventually, it shall be mentioned, that the methodological aspects of our work do not depend on any specific selection of tools. In this paper we applied it to a set of 22 tools only for the sake of illustration; of course it can be applied to a different set of tools, which will lead to a different outcome.

3. CONCLUDING REMARKS AND OUTLOOK

Regardless of the functionality a single tool can provide, it will not be able to unfold its full potential for the E/E architecture design when lacking appropriate interfaces to tools from other design phases. Moreover, generally, the tools do not allow an intuitive learning and thus make it necessary to attend time-consuming and expensive seminars which, together with the high license costs, can already become a criterion for exclusion. Along with this, the documentation and tutorials are often not sufficient enough to fulfill the needs of customers.

Taking all our findings into account, we can conclude that due to strong competition in the automotive domain, there is a lack of appropriate interfaces between tools from different vendors. In spite of a few, more or less, proficient exceptions, for example, provided by tools from *dSpace* or *Mathworks*, there is almost no native tool connectivity between the high-level design and the synthesis or verification and validation tools. And although AUTOSAR finds its way into more and more automotive software products, there is still a need for strong interconnection mechanisms beyond the established data exchange formats, like Data Base Container (DBC), Field Bus Exchange (FIBEX), Kabelbaumliste (KBL-Harness Description List), or AUTOSAR XML.

Nevertheless, it must be mentioned, that there are also good reasons to assume that tool vendors are interested in enhancing the functionality of their products towards a better tool interfacing. This development is supported by recent vendor partnerships, like the cooperation between *INCHRON* and *Vector* or *IBM*, respectively, as well as the announcement of interface plug-ins such as the one between *PREEvision* and *ChronSIM/ChronVAL*. Furthermore, it can be expected that some of the surveyed tools in this paper will establish themselves as a standard in the development of E/E architectures. Especially, products from *IBM* and *dSpace* seem to be well accepted by both OEMs and Tier 1 suppliers. In addition, *MATLAB/Simulink* is becoming a de-facto standard for code generation which extends to tools directly based on it, such as *TargetLink* or *SIMTOOLS*. With the study at hand as groundwork, we are furthermore interested in the research on universal information exchange mechanisms for E/E architecture design tools. Together with the analysis of the mutual compatibility between the tools, as mentioned in Section 2.5, this is part of a future work.

Acknowledgment

The authors would like to thank Wanli Chang, Reinhard Schneider, Licong Zhang and Matthias Kauer for their valuable contributions to this work.

4. REFERENCES

- [1] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *IEEE Computer*, vol. 35, no. 1, pp. 88–93, 2002.
- [2] U. Abelein, H. Lochner, D. Hahn, and S. Straube, "Complexity, quality and robustness—the challenges of tomorrow's automotive electronics," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 870–871, 2012.
- [3] G. Pitcher. (2012) Cutting Control Complexity. [Online]. Available: <http://www.newelectronics.co.uk>
- [4] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *IEEE Computer*, vol. 40, no. 10, pp. 42–51, 2007.
- [5] R. Zurawski, "Embedded systems handbook: Networked embedded systems," *CRC Press*, vol. 2, 2009.
- [6] O. Larses and J. El-khoury, "Multidisciplinary modeling and tool support for E/E architecture design," *World Automotive Congress, FISITA*, 2004.
- [7] P. Wallin and J. Axelsson, "A case study of issues related to automotive E/E system architecture development," *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pp. 87–95, 2008.
- [8] IABG - Industrieanlagen-Betriebsgesellschaft mbH. (2012) Das V-Modell XT. [Online]. Available: <http://v-modell.iabg.de>
- [9] FlexRay Consortium. (2012) FlexRay Communications Systems - Protocol Specification. [Online]. Available: <http://www.flexray.com>
- [10] C. Song, W. Chen, C. Feng, and D. Liaw, "Study of a vehicular drive-by-wire system based on flexray protocol," *Proceedings of SICE Annual Conference*, pp. 3618–3624, 2010.
- [11] R. Charette, "This car runs on code," *IEEE Spectrum*, vol. 46, no. 3, p. 3, 2009.
- [12] W. Stolz, R. Kornhaas, R. Krause, and T. Sommer, "Domain control units—the solution for future E/E architectures?" *SAE International*, 2010.
- [13] R. Obermaisser, C. El Salloum, B. Huber, and H. Kopetz, "From a federated to an integrated automotive architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 956–965, 2009.
- [14] A. Monot, N. Navet, B. Bavoux, F. Simonot-Lion *et al.*, "Multicore scheduling in automotive ECUs," *Embedded Real Time Software and Systems (ERTSS)*, 2010.
- [15] M. Di Natale and A. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.
- [16] W. Oh, J. Lee, H. Kwon, and H. Yoon, "Model-based development of automotive embedded systems: a case of continuously variable transmission (CVT)," *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 201–204, 2005.
- [17] AUTOSAR Consortium. (2012) Automotive Open System Architecture. [Online]. Available: <http://www.autosar.org>

Table 2: Investigated architecture design tools and their functionality

	Design Tool	Functionality
High-level	PREEvision (Vector)	Multi-layer architecture development supporting a model-based specification, design and evaluation of components; Provides signal routing and model consistency checks;
	Rhapsody Designer (IBM)	Model design based on the Unified Modeling Language (UML); Offers requirements analysis, simulation framework, traceability and code generation mechanisms;
	Volc. Vehicle System Architect (Mentor Graphics)	Design and management of automotive software and hardware systems; Contains mapping functionality to connect software components with ECUs and system signals;
	System Desk (dSpace)	Modeling of software architectures and functional networks; Verification and off-line simulation for software functions and component diagrams;
Network	Network Designer (Vector)	Building of bus architectures, schedules and communication matrices for <i>CAN</i> , <i>LIN</i> , and <i>FlexRay</i> ; Offers bus-specific consistency checks and visualization of gateway relationships;
	Volc. Network Architect (Mentor Graphics)	Design of communication systems based on imported vehicle functions and electrical architectures; Enables the definition of communication matrices and schedules as well as their verification;
	EB tresos Designer (Elektrobit)	Planning complex signal communication among distributed ECUs; Contains an extended support for <i>FlexRay</i> through simplified configuration functions and wizards;
Synthesis	Matlab/SIMULINK (MathWorks)	High-level language and numerical computing environment; Provides a platform for model-based design and simulation of dynamic and embedded systems supporting a number of tool boxes;
	SIMTOOLS/SIMTARGET (SIMTOOLS)	Matlab/SIMULINK toolbox for the development of distributed systems based on <i>CAN</i> and <i>FlexRay</i> . Allows a user-friendly configuration of timing and bus properties;
	ASCET (ETAS)	Model-based software development targeting the automotive domain; Allows the specification of executable real-time functions, graphical design mechanisms and automated code generation;
	TargetLink (dSpace)	Development of distributed systems based on SIMULINK models; Provides built-in simulation and testing mechanisms and enables an incremental code generation and code optimization;
	Artop/AUTOSAR Builder (Geensoft)	Series of tools for the design, configuration and simulation of AUTOSAR based systems; Supports the development and validation of software components and system descriptions on application level;
Verification and Validation	ChronSIM/ChronVAL (INCHRON)	Graphical validation and simulation of the systems's timing requirements; Analyzes and validates the dynamic behavior of a distributed architecture and provides real-time simulation and optimization;
	SymTA/S (Symtavision)	Timing and scheduling analysis of distributed embedded architectures; Allows to plan and optimize the system and its integration concepts and determine its reliability and safety;
	ControlDesk NG (dSace)	Flexible and modular development platform for automotive systems; Offers modules for ECU calibration, measurement and validation as well as bus configuration for <i>FlexRay</i> , <i>CAN</i> and <i>LIN</i> ;
	INCA (ETAS)	Base product for measurement and calibration of automotive systems; Provides extensions for diagnosis, monitoring and data acquisition and visualization for <i>FlexRay</i> and <i>LIN</i> buses;
	CANx (Vector)	Measurement, calibration and diagnosis of ECUs and ECU-based networks; Consists of several interconnected stand-alone tools: <i>CANape</i> , <i>CANalyzer</i> , <i>CANoe</i> , <i>CANdela</i> ;
	EB tresos Inspector (Elektrobit)	Monitoring and analyzing traffic on automotive buses; Enables precise measurements on <i>FlexRay</i> , <i>CAN</i> and <i>LIN</i> bus systems with enhanced data visualization features;
Management	DOORS (IBM)	Capturing, analyzing and managing system requirements; Offers optimization, traceability and verification mechanisms and allows the creation of automated and manual test cases;
	Quality Center (Hewlett-Packard)	Definition and management of system requirements; Provides version control, report generation and defect management to support different strategies for automated and manual test scenarios;
	ECU-Test (TraceTronics)	Test automation for ECUs-based systems; Supports the testing process by graphical test scenario definition, multi-staged test case execution and automated log-file generation;
	RPLAN e3 (Actano)	Collaborative project management; Enables managing and outsourcing parallel processes as well as workflow synchronization across project and company borders;

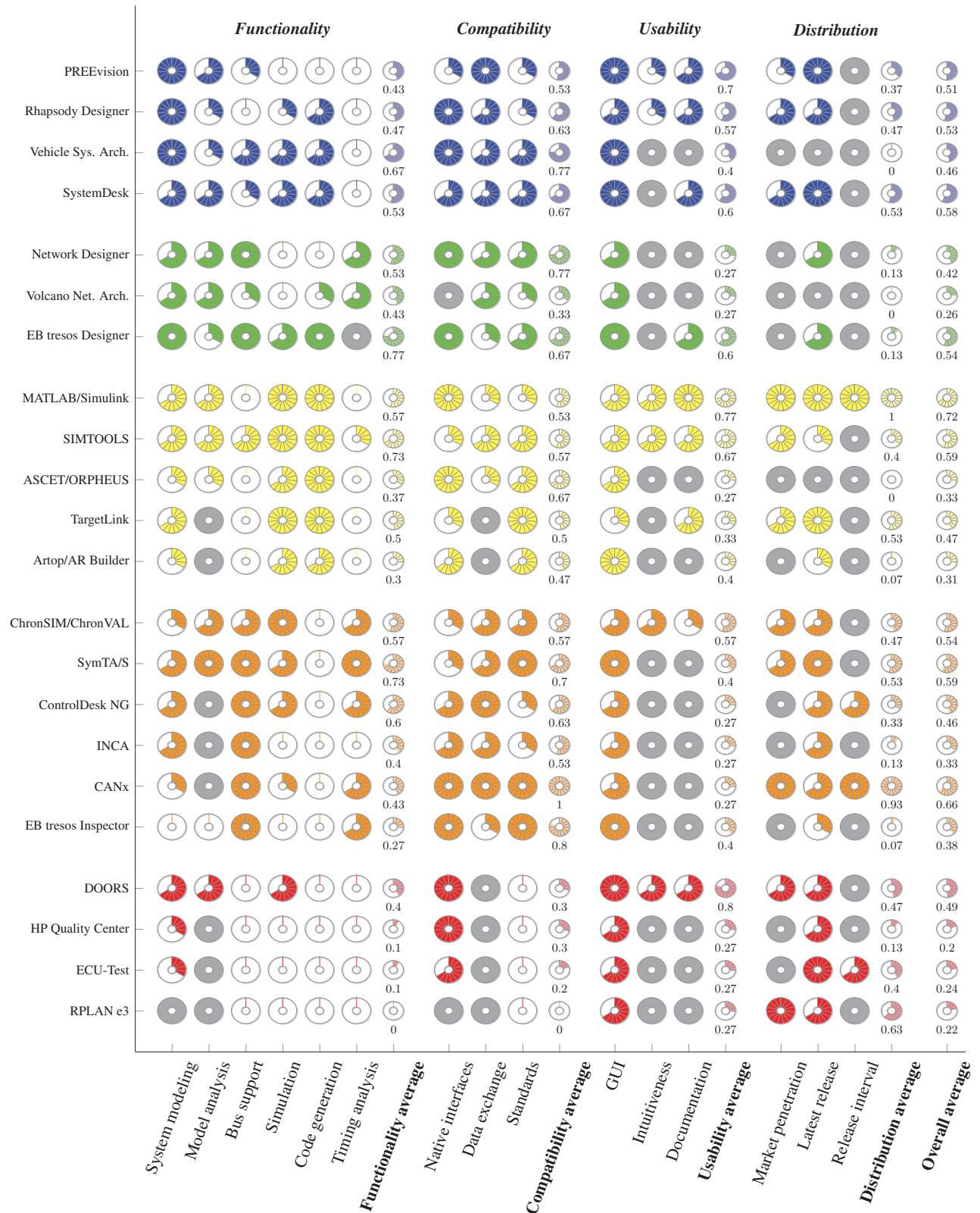


Figure 3: The Feature Metrics Matrix depicts the quantitative evaluation of E/E architecture design tools according to 15 criteria from four categories. The pie charts denote the support for each criterion by a particular tool (empty: no or unknown, 1/3: weak, 2/3: sufficient, 3/3: strong) and provide the corresponding average values.

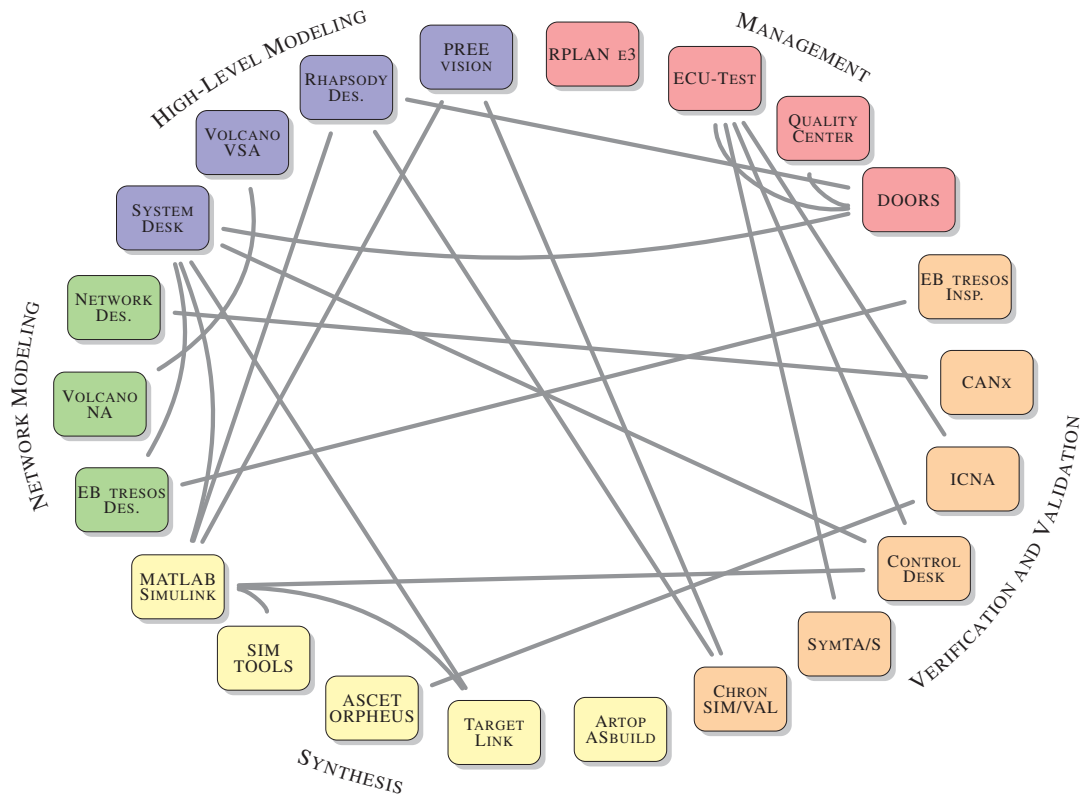


Figure 4: The Interface Graph illustrates interfaces between particular tools. The tools are colored according to their design phase and connections between them are represented by gray lines. The illustration does not show any information about the quality or direction of the interfaces.