Autosar Compliant Model-Based Static Resource Analyzer

Berkay Saydam
TTTech Auto Turkey;R&D Centre; Izmir, Turkey;

Abstract: As the automotive industry is a highly dynamic and aggressive market, a short period of time can determine whether a new model will be successful or not. The main focus is frequent delivery of customer value and a quicker response to changing conditions. There are too much iterations between OEM and Tier 1 because of not having knowledge about hardware restrictions on OEM side in spite of AUTOSAR and Model-based design. It signifies us to AUTOSAR can not provide totally independency between OEM and Tier 1 while it provides abstraction between software and hardware.The motivation of this article is that there is no any compliant studies for automotive industry even though there is for embedded systems.Automotive industry has different dynamics, therefore our tool is thought beyond the box. Resource analyzer removes these time-consuming iterations via analysing current OEM design. Thus, OEM does not consume resource and time to have this knowledge from Tier 1. Besides, the current system can be analysed to make a decision future SWCs and detected the most consuming SWCs in the system.

Keywords: AUTOSAR Workflow, Model-based Design, Static Analyzer, Independency on Design.

I. INTRODUCTION

In 2012, the value of cost for the development of electronics in vehicles was expected 35% of the vehicle production costs in 2015 [1]. According to the study [2], the cost was almost tripled for the development of electronic systems in 2015. Modern cars have above 100 ECUs (Electronic Control Units) and 100M lines of code by 2015 [3]. Code complexity climbs exponentially and it is expected to over 650M lines of code by 2025 [4].

Abstraction of hardware from software is provided by AUTOSAR (AUTomotive Open System Architecture), however there is no completely independency between OEM (Original Equipment Manufacturer) and Tier1 (supplier) companies. Embedded systems have limitations, therefore designers of OEM should contact with developers of Tier1 to betested model according to limitations. Model-based design saved time with bringing easy modification on design via model approach. Despite all, there are many iterations on model because of not having knowledge about hardware resource limitation on OEM side. These iterations cause too much cost because they causes to give the product to market lately.

The overflow of storage and computationcan be seen in release phase, but this phase is too late to handle this situation. Because this situation requires to modelling againby OEM. Since the automotive industry is a highly dynamic and aggressive market, there is a short time. Storage and computation consumption of present model are produced via the tool. OEM designer can check their model in software component design phase with our tool.Thus, wasting iteration times is removed and there is independency between OEM and Tier1 in AUTOSAR workflow.

In this article, the reason of the requirement of standardized software architecture and AUTOSAR, which is a proper software architecture, are given in chapter 2. There is detailed information about model-based software development as complementary of AUTOSAR in chapter 3. Our tool is introduced in next chapter. The last section is conclusion which includes the benefits of tool.

II. REQUIREMENT OF STAND ARDIZED SOFTWARE ARCHITECTURE

The steady growth in the number of features and ECUs caused the complexity on E/E (Electrics/Electronics). Quantity of software have been rising in parallel with this [5]. Many kind of different hardware platforms are used by OEM. It limited the modularity of the software. Supplier have to support all OEM variants with their software. Reusability of the software was very low because there is no standardization between variants. This brought to mind the idea of a requirement of standardized software architecture.

2.1 AUTOSAR Partnership/Framework

AUTOSAR is a worldwide partnership which has "Cooperate on standards – compete on implementation" slogan [6]. The main goal of AUTOSAR is providing the reusability of pre-validated software components with reuse existing functions in different hardware. It provides abstraction of hardware from software to bring flexibility in development. It aims the improvement of software quality via changing process from implementation to configuration. It enhances the whole product life cycle with providing maintainability and the availability of upgrades over the entire life of a vehicle. In conclusion, competition is proceeded onto OEM-relevant features and resources are used for improving software quality instead of spending unnecessary time for implementation repeatedly.

2.2 AUTOSAR Workflow

In the automotive industry, companies make development cycles get shorter with continuous integration and delivery.Agile software management methods are also used fast release-cycles to give response the market needs.AUTOSAR arising from component-based software engineering where a system is divided into a number of software components each of which encapsulates a set of related functions which are called runnable entities. The interfaces of the software-components are designed and the software components are mapped to ECUs. The result is a system description which includes the interface of such components and is given to the supplier who implements the behavior of the software components, performs the mapping of runnables to OS tasks, and configure the basic software modules like the communication stack.Afterwards, the software is compiled, built, and flashed to an ECU.

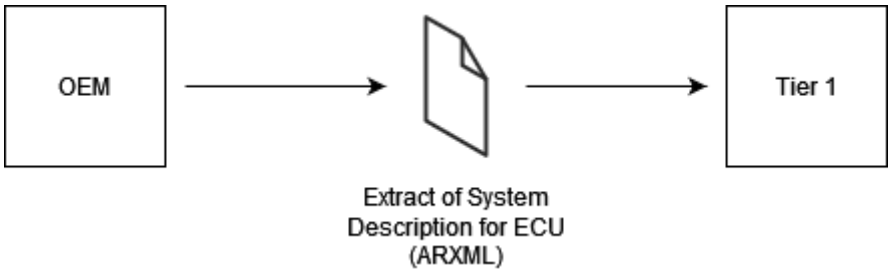

Figure 1: The relation between OEM and Tier-1

In the first step of AUTOSAR workflow, a software component is modeled in ARXML (AutosaR XML) format on VFB (Virtual Functional Bus) level [7]. It is called Software Component Description which includes operations and data elements provided and required, requirements regarding the infrastructure, resources needed and also information regarding the specific implementation.The communication between Software Components on System Level is modelled with the help of the concept of the VFB.

System description is obtained with mapping SWCs (Software Components) to ECUs in the second step. An extract is created in ARXML format for each ECU.This is the output of OEM to give Tier1. Tier1 takes this description file as input and integrating and configuring one ECU's software are done according to the specifications. The steps of AUTOSAR workflow is given in Figure 2.

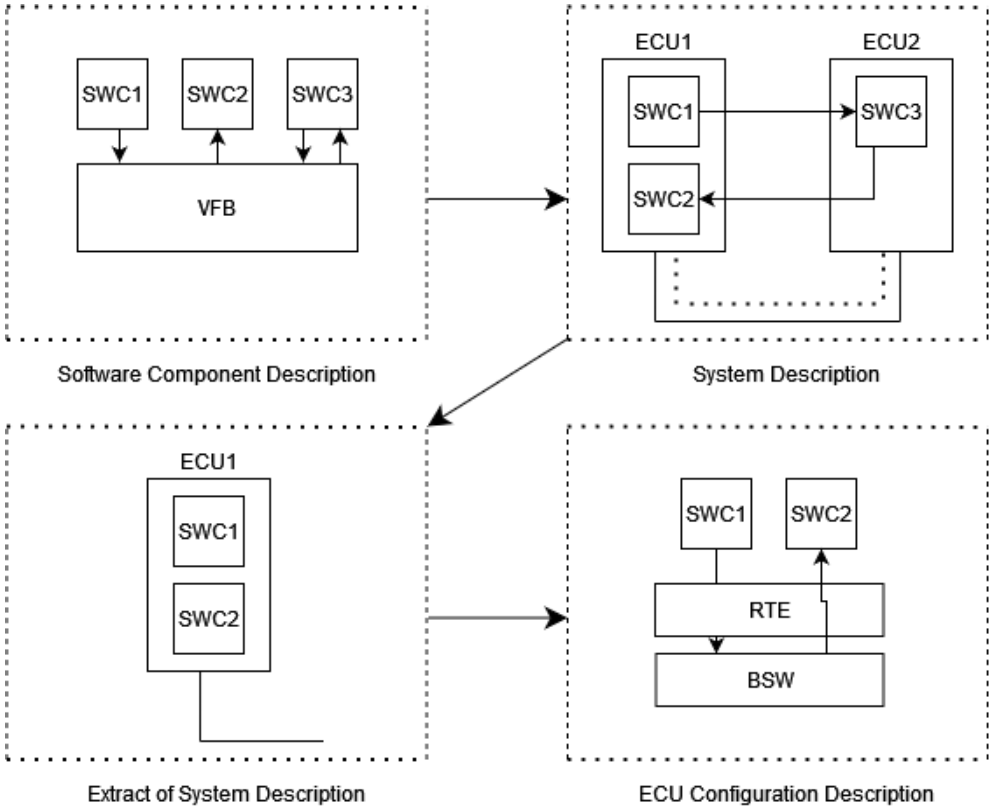

Figure 2: AUTOSAR Workflow (Methodology)

III. MODEL-BASED SOFTWARE DEVELOPMENT

Model-Based Design is a paradigm to solve complexity on a functional level while AUTOSAR is focusing on the software and communication aspects between AUTOSAR software components and the decoupling of the application and the Basic Software layer. Model-Based Design and AUTOSAR are not only compatible, but they are complementary. The combination of both is an excellent way to improve the cooperation between system engineers and design engineers as well as the collaboration between OEMs and suppliers.

Traditional embedded software development involves paper designs and hand coding followed by verification activities. These activities involve manual interaction because of lacking tool automation. Therefore, they are error prone and time consuming. Lack of tool chain causes errors to be found in the software which are often detected late and at high costs to the development process.

Model-Based Design aims error prevention and early error detection. It gives opportunity to detect error earlier in a project. The software is automatically generated from the models during production codegeneration. If the continuous verification and validation process discovers errors, they will be fixed within the model and the software can be regenerated very quickly and efficiently. Model-Based Design enables teams to complete important tasks much earlier than in a traditional approach. At the same time it enhances the quality of the complete design process, because it avoids time and cost intensive iterations if errors are found at the very end.

In brief, Model-Based Design provides these benefits:

- Continuous test and verification to detects errors earlier
- Automatic code generation to minimize manual error prone coding
- Designing with simulation to create prototype
- Executable specifications to facilitate communication between OEM and supplier

IV. IMPLEMENTATION

The tool takes ARXML files for each host as inputs. The outputs are xlsx format for each host to being readable by human.The tool system can be summarized like Figure 3.

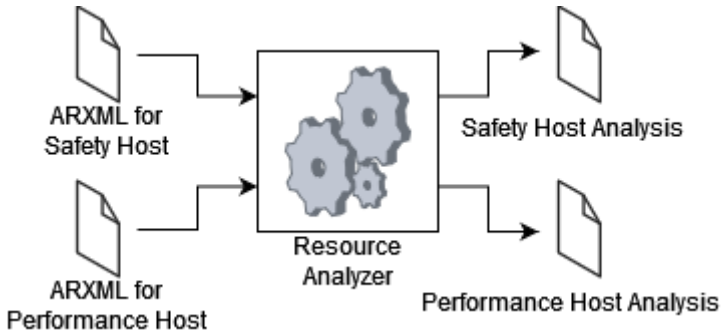

Figure 3: System block diagram

Our tool takes SWCs, preferred type of communication between SWCs, data type which is used in these communications and so on from these ARXML files. Resource Analyzer estimates RAM (Random Access Memory) and ROM (Read-Only Memory) allocations which will be done by RTE (Run-Time Environment) according to generic calculations. It gives readable table which has statistics as Figure 4.

	Port Name	Data	Rte	Size (As Byte	Producer ROM (As Byte	Producer RAM (As Byt	Consumer ROM (As Byte	Consumer RAM (As Byte
539	PpPdSWC0_DeAdasGeometries	DeAdasGeometries	MBU	4988	4994	14973	12	12
540	PpPdSWC0_DeAdasObjectsParameter	DeAdasObjectsParameter	MBU	6228	6234	18693	12	12
541	PpPdSWC0_DeRoadmodelParameter	DeRoadmodelParameter	MBU	12156	12162	24319	6	6
542	PpPdSWC1_DeParameterCameraFront	DeParametersCameraFront	MBU	268	274	1083	12	12
543	PpPdSWC1_DeParameterCameraLeft	DeParametersCameraLeft	MBU	268	274	1083	12	12
544	PpPdSWC1_DeParameterCameraRear	DeParametersCameraRear	MBU	268	274	1083	12	12
545	PpPdSWC1_DeParameterCameraRight	DeParametersCameraRight	MBU	268	274	1083	12	12
546	PpPdSWC2_DeSuspension	DeSuspension	MBU	24	30	107	12	12
547	PpPdSWC2_DeSuspensionBack	DeSuspensionBack	MBU	24	30	107	12	12
548	PpPdSWC2_DeSuspensionFront	DeSuspensionFront	MBU	24	30	107	12	12
549	PpPdSWC2_DeSuspensionLeft	DeSuspensionLeft	MBU	24	30	107	12	12
550	PpPdSWC2_DeSuspensionRight	DeSuspensionRight	MBU	24	30	107	12	12
551	PpPdSWC2_DeViewParameter	DeViewParameter	MBU	524	530	2107	12	12
552	PpPdSWC3_DeRoadgraphPersistent	DeRoadgraphPersistent	MBU	12	8	14	8	2
553	PpDsSWC3_DeRoadgraphDataSet	DeRoadgraphDataSet	MBU	1036	8	1038	4	1
554	Overall data element memory consumption:	ROM : 215264 B	RAM : 586535 B					

Memory Consumption Analyze | Runnable Analyze | SWC Analyze

Figure 4: The report of Resource Analyzer

The report of our tool can be examined some different perspectives such as memory consumption, runnable and SWC. The SWCs, which consume too much memory or CPU (Central Processing Unit), can be detected with our tool. This process can be applied for runnables, which are found in SWCs, via SWC Analyze sheet in the report.

V. CONCLUSION

With our tool, OEM can design its model without having any dependency. It can check the model runnable by runnable. Some runnables or SWCs, which are consumes too much resource, can be removed or optimized after checking the report of resource analyzer. The decisions can be made for the future functionalities in research and development phase with checking current model consumptions. The overflow of resource can be detected in the earliest phase.

The visualisation of present values can be future study for this tool. Besides, some artificial intelligent algorithms can be added to give recommendation to designer for having the best model which has the lowest storage and CPU usage. This also reduces time spentand designer effort and provides opportunity to have the best model.

REFERENCES

[1] M. Brou, S. Kirstan, H. Krcmar and B. Schartz, "What is the benefit of a model-based design of embedded system in the car industry?" Emerging Technologies for the Evolutin and Maintenance of Software Models, pp.343-369, 2012.

[2] Franco, F.R., et al: Workflow and Toolchain for Developing the Automotive Software According AUTOSAR Standard at a Virtual-ECU. IEEE 25th International Symposium on Industrial Electronics (ISIE), pp. 869-875, 2016.

[3] Open standards enable continuous software development in the automotive industry

[4] R. McOuat, Cars are made of code, 2020 (accessed August27, 2021).[Online]. Available: https://blog.nxp.com/automotive/cars-aremade-of-code.

[5] Mullangi Divyakanth, Why AUTOSAR, 2019 (accessed July 23, 2021).[Online]. Available: https://assets.vector.com/cms/content/events/2019/VH/VIC2019/Track_4_1_Why_AUTOSAR.pdf

[6] https://www.autosar.org/. [Online]. Available:https://www.autosar.org/about/. [Accessed: 02- Jun- 2021].

[7] Leoš Mikulka, "Low-level software for automotive electronic control units references", May 2013.