Automotive Software Architecture Views and Why we need a new one – Safety view
Miroslaw Staron

To cite this version:
Miroslaw Staron. Automotive Software Architecture Views and Why we need a new one – Safety view. Workshop CARS 2016 - Critical Automotive applications: Robustness & Safety, Sep 2016, Göteborg, Sweden. hal-01372336

HAL Id: hal-01372336
https://hal.archives-ouvertes.fr/hal-01372336
Submitted on 27 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Automotive Software Architecture Views and Why we need a new one – Safety view

Miroslaw Staron*
*University of Gothenburg, Sweden
miroslaw.staron@gu.se

Abstract—The growth of the size and complexity of automotive software has been driven by the increased number of software-dependent innovations in modern cars. The growth of the size and complexity drives the usage of multiple abstraction levels in automotive software designs. One of these levels is the architecture level where the high level design of software is documented using multiple views such as logical, function, deployment.

In this paper we review these architectural views and argue that they require one more view – safety view – which is compliant with the ISO/IEC 26262 and links the design of software systems (both the functional and system views) with the safety goals. We advocate the need for this view that would help the designers to understand and trace the design of software safety throughout the software lifecycle.

I. INTRODUCTION

As the amount of software in modern cars grows we observe the need to use more advanced software engineering methods and tools to handle the complexity, size and criticality of the software [1], [2]. We increase the level of automation and increase the speed of delivery of software components. We also constantly evolve software systems and their designs in order to be able to keep up with the speed of the changes in requirements in automotive software projects.

However, we also need to handle the complexity is an efficient way, meaning that we need to adjust the engineering methods depending on the criticality of the software, according to the prescriptions of the ISO/IEC 26262 standard [3] – both during the design and verification/validation of software systems. These analyses are currently poorly supported in the general software engineering tooling and methodologies. Therefore in this paper we set off to address the following research question:

How to provide explicit support for safety-related design elements in the architecture work?

By the safety-related design we mean the support for safety related questions which are important during the design of the system, such as:

- How to identify which software elements should be validated using which methods based on their ASIL levels?
- How to find out which elements are linked to a specific safety case?
- How to identify which elements should be re-validated if the safety goals/cases change?

Even though these questions seem rather straightforward we need an architectural view which will allow us to see the ASIL levels assigned to the system elements, their verification requirements, relevant safety-related metrics (e.g. control path complexity), etc. We also need to be able to see which software elements have none of the safety elements assigned to them, such as which software components have no ASIL level assigned or which ASIL D software components have a control path complexity way above verifiable limits.

II. RELATED WORK

One of the most prominent works in the area of visual safety argumentation has been done by Kelly and Weaver [4] who presented a notation for modelling safety goals. The main goal of the notation is to link the safety goals to the safety evidence using the safety argumentation. Similar approach has also been done in the later work of Hawkins et al. [5]. In our proposal we use the same symbols for the safety arguments and goals, and complete with the other modeling elements of interest.

Xiaoping et al. [6] describes the safety editor based on the ISO/IEC 26262 standard and shows that this view requires linking to the architectural elements of the system construction, something that we also support and present a proposal in this paper.

Hause and Thom [7] present a method for extending SysML to show safety concerns integrated into SysML models. Our proposal is based on very similar principles, although adding a separate view, which is not present in SysML.

Another part of the safety view is the visualization of metrics in order to reason about the sufficiency of the safety argumentation. Termeer et al. [8] presented a tool which augmented UML diagrams with metric information, a view which we intend to integrate with the safety view of the architecture – or at least the ideas presented in their work.

III. ARCHITECTURAL VIEWS

Architecting is a continuous process, which starts at the first requirement and ends with the last defect-fix in the car. Although one could see the process of architecting as a prescriptive design, the process continuous as the design evolves. Certain aspects of design decisions influence the architecture and are impossible to be known a priori – increased processing power required to fulfill late function requirements or safety criticality of the designed system. If not managed correctly the architecture has a tendency to evolve into a descriptive documentation that needs to be kept consistent with the software itself [9], [10], [11].
In order to put the process of architecting in a context and describe the current architectural views in automotive software architectures let us first discuss the V-model as shown in Figure 1.

The first level is the functional development level where we encounter two types of the architectural views – the functional view and the logical system view. The functional view, often abbreviated to the functional architecture is the view where the focus is on the functions of the vehicle and their dependency on one another [12]. An example of such a view is shown in Figure 2.

Another view is the system view on the architecture, usually portrayed as a view of the entire electrical system at the top level with accompanying lower level diagrams (e.g. class diagrams in UML). Such an overview level is presented in Figure 3.

The focus of the view is on the topology of the system. This view is often accompanied by the logical component architecture as presented in Figure 4.

For the logical view the architects often use a variety of diagrams (e.g. communication diagrams, class diagrams, component diagrams) to show various levels of abstraction of the software of the car – usually in its context. For the detailed design these architectural models are complemented with low level executable models such as Matlab/Simulink defining the behaviour of the software [13].

IV. SAFETY VIEW

The previously presented view are dedicated for the design disciplines in software engineering, but in the automotive software engineering the safety aspects are usually treated as a separate concern and therefore we postulate that they need a specific architectural view. Having such a view is important as it allows to easily trace the safety argumentation, goals and elements to the system design elements. And since the number of system elements in modern cars is constantly growing, having such a linkage allows to assure that we do not miss an important design step or validation step/method required for achieving a specific safety level.

In this view we need to include the concepts which are important for safety engineering:

- Safety goals – safety goals as specified in ISO/IEC 26262
- Safety arguments – argumentation for the fulfillment of the safety cases (documents), usually in form of the safety cases
- Safety verification requirements – verification require-
The research results presented in this paper contain the meta-model of the new view which shows how the combination of these two domains (safety and software construction/design) can be combined by a simplistic extension of the UML/SysML meta-model. This extension can be done both as a fully-fledged extension or as a stereotype extension [14], [15]. The stage of this research is rather preliminary but it shows the potential impact on the design of software for software critical systems.

Our future directions include the development of the UML profiles for Papyrus UML tool where we can include the stereotypes for modelling safety cases and linking them to the software design elements as proposed in this paper.

ACKNOWLEDGMENT

The research presented here is done under the VISEE project which is funded by Vinnova and Volvo Cars jointly under the FFI program (VISEE, Project No: 2011-04438)

REFERENCES

Fig. 5. The meta-model of the safety view.

Fig. 6. Example of a safety view combining two notations.