

Chapter 2

Model-Based Functional Safety Engineering

Dariusz Szymanski, Matthias Scharrer, Georg Macher,
Eric Armengaud and Holger Schmidt

Abstract This chapter presents some aspects of model-based way of working applied in the EU funding projects called iCOMPOSE [1] and INCOBAT [2]. The presented results from INCOBAT project focus on basic structure that serves to organize/handle the technical complexity of the system under development. This approach was supported by the use case of INCOBAT battery system. The presented result from iCOMPOSE project focus on SysML model-based approach applied during the safety engineering development process and software integration of Integrated Comprehensive Energy Management System (iCEM). Some extensions to SysML profile are proposed and presented in this chapter and some findings on the data processing capabilities are discussed. The software development and integration of iCEM's embedded software required also a dedicated toolchain which was established and deployed taking into account constraints typical for automotive domain like safety criticality, real-time applications, various communications interfaces, variety of tools involved in the development process.

Keywords Software development toolchain · Functional safety · SysML · Model-based · Automotive domain

D. Szymanski (✉)
Flanders Make, Lommel, Belgium
e-mail: dariusz.szymanski@flandersmake.be

M. Scharrer
VIRTUAL VEHICLE Research Center, Graz, Austria
e-mail: Matthias.Scharrer@v2c2.at

G. Macher · E. Armengaud
AVL, Graz, Austria

H. Schmidt
Infineon, Augsburg, Germany

2.1 Introduction

Modern automotive systems require high computational power due to complicated algorithms and are often of safety critical nature. This safety criticality imposes additional constraints on the embedded software itself as well as on its development process including the integration phase. While the requirements for safety-critical software development are well defined in part 6 of ISO–26262 standard [3], the efficient execution of the process can still be a challenge. SysML modelling promises to solve this problem of consistent and evolutionary description of the item including required traceability. However SysML profile is not capable of providing a full support for the safety engineering process. Some extensions to SysML profile towards functional safety engineering are possible and enhanced interfaces to supporting toolchains are required. The new concept of augmented Fault Tree Analyses (aFTA) performed with the help of SysML is presented in subchapter 3 of this paper, while the integration toolchain for software development supporting the SysML profile is presented in subchap. 5. The concept of Global System Structure providing support in correct structuring of complex systems is presented in subchap. 1.

2.2 The INCOBAT System Modelling Approach

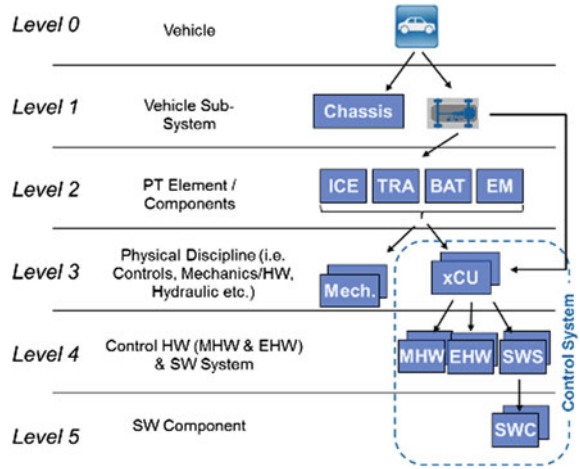
In context of the automotive domain it is very common to have some basic structure that serves to organize and handle the technical complexity of the system under development. Within AVL Global System Structure (GSS) has been defined and an aligned “Level Structure” for requirements, specifications, design, and integration steps & testing has been set up. This GSS level structure is depicted in Fig. 2.1 and also used in the INCOBAT project.

The purpose of the level approach is to have a classification framework to assign requirements/designs to the level they belong to. On the individual level, the requirements/designs are the basis to derive further requirements/designs to the next down-stream level. In other words: the requirements/designs are developed following a top-down cascading approach where on every level the requirements/designs will get more detailed. Nevertheless, if the project specifics and scope require, certain levels may be omitted.

Level 0—Vehicle—The vehicle structure element hierarchy and vehicle level solely represents a possible use-case for the battery management system and helps to serve as a common knowledge base for the project partner.

On this level specific environmental constraints can be govern and this most abstract layer helps to serve as a common knowledge base for the project partner to determine the sources and sinks for information of the BMS and the electronic and mechanical interaction partners. Due to the fact, that the INCOBAT project is focusing on a generic battery system applicable for manifold applications and

Fig. 2.1 AVL global system structure (GSS) and level structure (abbreviations are explained further in the text)



vehicles no assumptions, requirements, or design specifications are done on this level for the vehicle or vehicle's powertrain.

Level 1—Vehicle Sub-system—On this level again a generic electric powertrain structure is assumed. This structure mainly consists of a combustion engine (ICE), transmission (TRA), energy storage (BAT), and an electro-motor (EM). Nevertheless, some assumption and requirements for the powertrain and electrical drive have to be done on this layer.

As the INCOBAT battery system is developed according to the safety element out of context (SEooC) approach of part 10 of ISO-26262 [4], some fundamental assumptions are stated for the SEooC development. Assumption examples for the INCOBAT battery are e.g.: 'A_04 recuperative braking is not the only brake system within the car' and 'A_14 e-motor is freewheeling in case of powertrain failure or battery disconnection'.

Level 2—Powertrain Element (PT)/Component—On this layer the item definition for the INCOBAT battery system is done. According to clause 5.4.2 of part 3 of ISO-26262 [5] 'the boundary of the item, its interfaces, and the assumptions concerning its interaction with other items and elements, shall be defined ...'. Therefore, the INCOBAT battery item and its interfaces to the environment are defined and a Hazard Analysis and Risk Assessment (HARA) has been performed on this level.

The objective of the HARA is to identify and categorize the hazards that malfunctions in the item can trigger. Furthermore, to formulate the safety goals related to the prevention or mitigation of the hazardous events, in order to avoid unreasonable risk.

Safety goals (SG) and their assigned ASIL are determined by a systematic evaluation of hazardous events. This analysis is based on the item's functional behavior and determines the ASIL for the system by considering the estimation of the impact factors severity, probability of exposure, and controllability of the hazardous events.

‘SG5: Prevent from electric shock’ with an assigned ASIL D is here an example. On this level the functional safety concept (FSC) is composed by functional safety requirements (FSR), such as: ‘FSR_07: The safe state which shall be reached is disconnection of HV contactors.’ And ‘FSR_13: BMS shall control the energization/de-energization of the HV boardnet by control of the contact relays’.

This FSC is related to the INCOBAT battery system and its sub-components. Hereby, also the mechanical system must be taken into considerations, thus also a separation of mechanical and electronic control system is done on this level.

Level 3—Physical Domain/Control System—This layer constitutes the most concrete layer of system development and solely focuses on electrical and electronic components (HW and SW). Therefore the structure elements of the main and satellite modules compose of hardware and software elements.

On this abstraction level the FSC is evolved to a technical safety concept (TSC) and functional requirements are engineered, such as: ‘Req 346102: BMS shall control HV+ relay for proper energization and de-energization of HV boardnet when requested by vehicle control unit’ and ‘Req 346197: BMS shall control HV precharge relay for proper energization (ramp-up) of HV boardnet when requested by vehicle control unit’.

Level 4—HW/SW System—This abstraction level is not included in classical system development approaches but defines the start of parallel software and hardware architecture design. This abstraction level is therefore mostly only present in special purpose SW development tools or HW design tools rather than SysML based system development tools. With the INCOBAT modeling approach this tool breach and semantic gap can be bridged and SW architecture definition can be represented by the model-based development tool.

Already at this level (and further on for Level 5) the limitation of using SysML approach has been experienced. Hence, the high level of details and the specific vocabulary makes the use of Domain Specific Languages (DSL) more appropriate. The challenge is thus to transfer and synchronize the information between a general purpose language (such as SysML) to different DSL. To support this the elements on this level are represented using the INCOBAT model approach and meta-model.

At this level the mapping of HW to SW interfaces (HSI) [6] can be modeled. This supports the explicit definition and configurations required to support correct implementation of, e.g., HV relay signals.

Furthermore, SW architectures and specific HW blocks can be designed.

2.3 iCOMPOSE Model-Based Functional Safety Engineering

The starting point of functional safety engineering process according to ISO-26262 is Concept Phase (see Fig. 2.2).

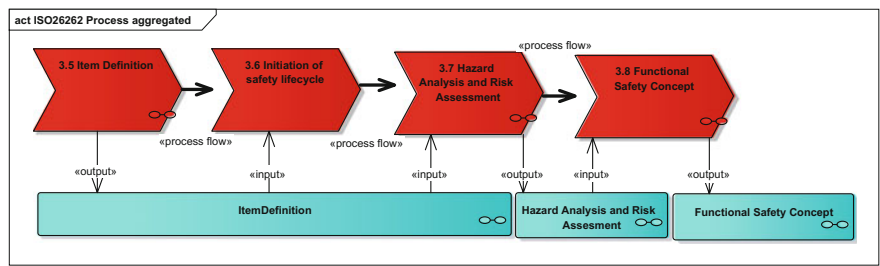


Fig. 2.2 Process model of ISO 26262 concept phase of safety engineering lifecycle in business process modelling notation (BPMN)

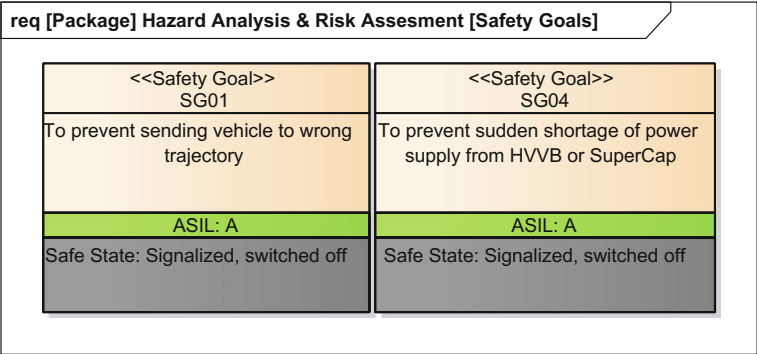


Fig. 2.3 Safety goals of iCEM in case of road usage

During this phase functional, operational and environmental requirements were collected, the boundaries of Integrated Comprehensive Energy Management System (iCEM) together with its preliminary architecture were identified. iCEM’s architecture was modelled with the help of block definition diagrams. Safety Goals (SG) were identified with the help of HARA and they were captured in the SysML model with the help of dedicated extension [7] (see Fig. 2.3).

According to clause 8.4.2.3 of part 3 of ISO–26262 [5] Functional Safety Requirements (FSR) derivation can be supported with Hazard and Operability Study, Failure Mode and Effect Analysis, Fault Tree Analysis (FTA). In the presented approach FTA constitutes the basis for FSR derivation and it is included in SysML model and augmented in this sense that it also shows relations of undesired event to previously identified shortfall states, SGs, software units and basic events (UE). The concept of augmented FTA (aFTA) requires mixed elements coming from various SysML diagrams to be present as a specific view in a single diagram (see Fig. 2.4).

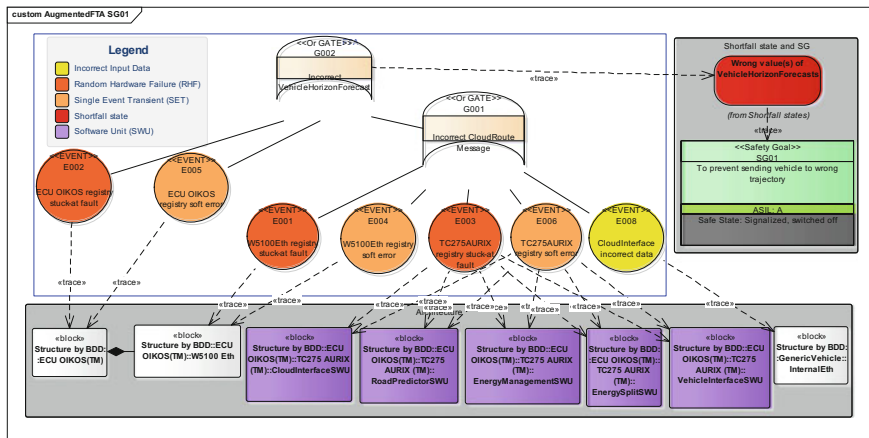


Fig. 2.4 Augmented fault tree for SG01

Activities which describe the functionality of basic safety mechanism preventing from violation of SGs were designed after identification of UEs with the help of top-down approach using System Use Cases (SUCs). The specific feature of these SUCs is that SGs are actors and all SUCs have unique identifiers (see Fig. 2.5). Finally aFTA together with system use cases constitute demanded rationale for FSRs.

During the System Design phase of ISO–26262, SUCs defined with the help of aFTA are further detailed. This enabled an identification of activities which describe the response of the iCEM and other elements to stimuli that can lead to violation of SGs as depicted in Fig. 2.6.

Finally the activities were decomposed into the Technical Safety Requirements (TSR) and rationale was provided for each derived TSR. The presented structure of TSRs (see Fig. 2.7) reflects the scope of clause 6.4.2.2 of part 4 of ISO–26262 [6]. Furthermore, the numbering scheme applied to TSRs also matches the labeling applied to lists in the clause 6.4.2.2.

The specific coloring schemes applied in the diagrams provide not only better readability of the diagrams but also they cover some important pieces of information which can be seen in the legend included in Fig. 2.7. Also the frames were used as the way to cluster important information. The frame with grey background shows elements already present in the model which are re-used in given diagram. In this way the evolution of safety design can be captured. Finally, the Preliminary Architecture was further detailed resulting in System Design architecture. This architecture takes into account measures to control random hardware failures provided by native safety mechanism of AURIX controller [8, 9].

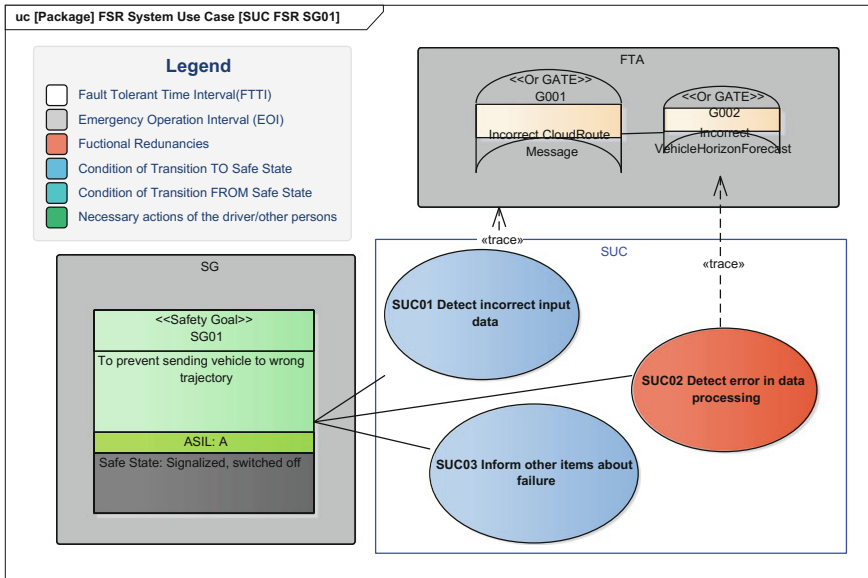


Fig. 2.5 FSR derivation with the help of system use cases

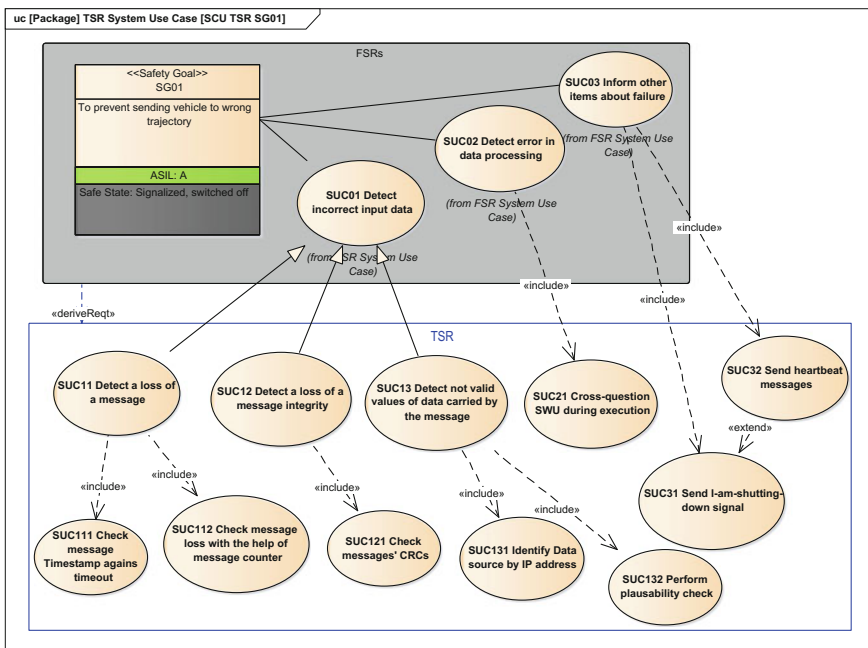


Fig. 2.6 TSR derivation with the help of system use cases

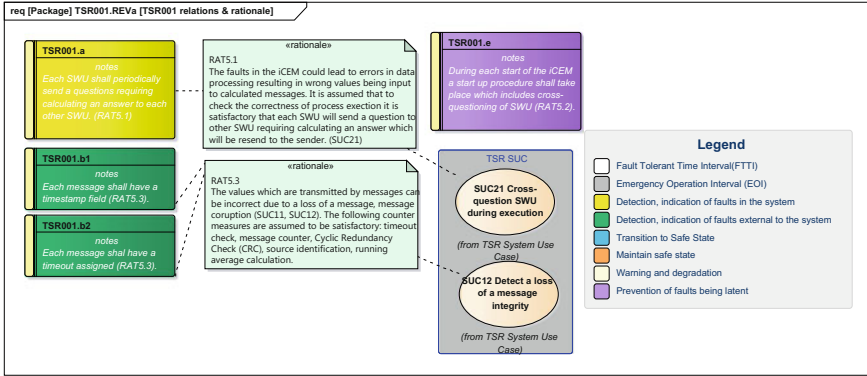


Fig. 2.7 Technical safety requirements representation

2.4 Aspects of Traceability

The applied approach assures achieving traceability between all important artefacts of the development process, however it requires creating various relations between different types of SysML modelling elements: requirements to requirements, use cases to uses cases, blocks to use cases, SGs to use cases, SGs to states, OR as well as AND gates of aFTA to states, aFTA gates to aFTA events, aFTA events to blocks, aFTA gates to use cases, requirements to blocks, requirements to signals, requirements to rationale, rationale to SUCs, blocks to blocks, SGs to blocks. Where, SGs, OR as well as AND gates and events of FTA are non-standard extensions of SysML profile.

It was noticed that due to complicated structure of relations the best results in performing traceability can be achieved by direct querying SQL database running in the background of the SysML modelling tool, which in this case was Enterprise Architect.

2.5 Software Development and Integration Toolchain

The main idea of the software development scenario within the iCOMPOSE project was to set up a consistent toolchain from requirements to deployed code with code generation at the earliest stage possible. The elaborated toolchain is based on models of different levels of abstractions and it is offering consistency from Functional Safety Concept up to software development and software testing. Figure 2.8 shows the proposed and implemented workflow.

Functional software development started already in the concept phase described in Sect. 2.3. The preliminary architecture was identified along with drafting state machines that described the iCEM's behaviour. This information was detailed

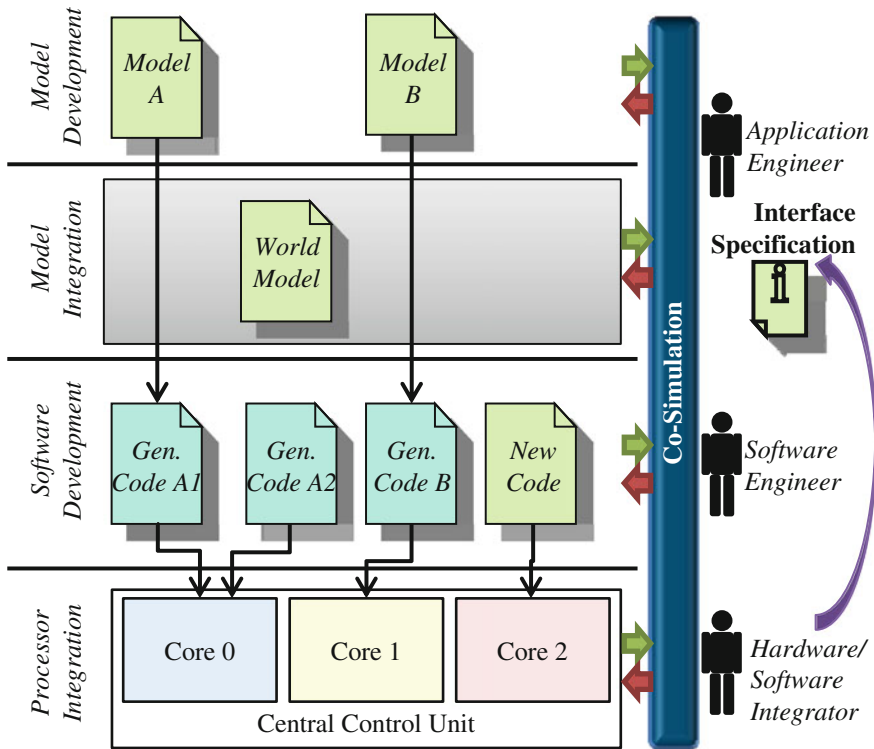


Fig. 2.8 Proposed software development and integration workflow

further during System Design Phase by drafting class diagrams and improving the state machine behaviour.

The major contribution to the toolchain achieved in Enterprise Architect is the automatic generation of MATLAB¹ script code implementing and thus making further use of the state machine behaviour and class structure as designed earlier. A new *Enterprise Architect Code Template* was implemented for MATLAB consisting of:

- classes, including
 - attributes,
 - methods,
 - actions, i.e., branches, loops, calls, and assignments, and
 - behaviours, as well as

¹MATLAB 2016b, <https://www.mathworks.com/products/matlab/>.

- state machine fragments of type “legacy”, i.e., previous to release 11.0,
 - states
 - transitions, and
 - triggers.

The generated state machines serve as a code stub to the iCEM’s framework for subsequent software development, reducing the effort as well as sources for errors with respect to re-implementing the designed software in the target programming language.

Since all functions to be integrated into the iCEM’s software are based on MATLAB and Simulink, Mathworks Embedded Coder together with MATLAB Coder and Simulink Coder were employed to generate C-code from Simulink diagrams targeting at embedded devices. This generated C-code is particularly well suited for use in real-time applications. Figure 2.8 depicts the proposed software development and integration workflow. Development stages are separated by horizontal black lines and labelled at the left. Items under development are depicted schematically on every stage and linked to corresponding items on the next development stage. Pictograms of humans and labels show the roles in the development process at the right. All stages are connected vertically via a common “co-simulation” that may interact with a combination of models at any stage. The arrow on the right indicates a stronger interaction between the Hardware/Software Integrator up to the Application Engineer level by providing the interface specification to the higher level.

For many real-time projects it is essential to communicate with target hardware via hardware interfaces. Hardware interfaces from Software in Simulink are typically source or sink blocks. In this project, device drivers for two interfaces were developed targeting the iCEM application:

- Controller Area Network (CAN) and
- Universal Asynchronous Receiver Transmitter (UART), better known as RS232 or serial port.

The device drivers were developed using the Legacy Code Tool in MATLAB. This tool automatically generates a block for custom code.

Figure 2.9 shows a Simulink diagram with a CAN receive and two CAN transmit devices already linked to a subsystem block that represents a generic model. Both block types handle common CAN message settings, such as offset, factor, start bit and stop bit of each input signal. This configuration is similar to the dbc format of VECTOR tools.²

In contrast to the bus-based CAN communication, data transmission through UART is designed to be a very flexible and generic point-to-point connection, but in this case, the UART device drivers were implemented to encapsulate data in a

²CANdb++, http://vector.com/vi_candb_en.html. The dbc format provides a quasi-standard for specifying CAN messages and included signals in terms of frequency and data format.

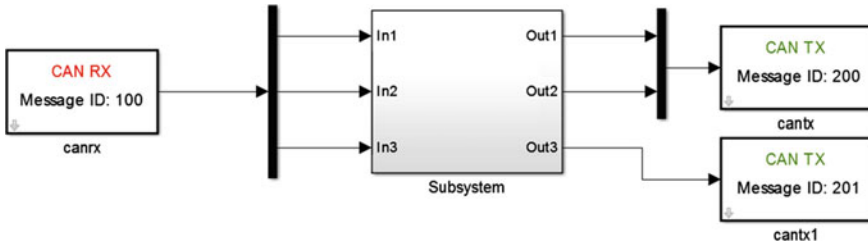


Fig. 2.9 CAN device drivers in a Simulink diagram

similar fashion to the CAN-protocol, as both implementations were designed to communicate with a vehicle controller.

Since these functions are hardware specific, it is necessary to ensure that the functionality of the devices is only used in combination with the target hardware. During simulation on the development platform, i.e., the development desktop computer, the devices blocks (sinks and sources) have no functionality, yielding constant input. However, on the target platform the blocks result in code fragments that implement the desired functionality.

On top of the device drivers described above, another major contribution to the toolchain achieved in MATLAB is the automatic generation of a Simulink model stub. Since the description of the communication via the CAN bus is described commonly in dbc format, a tool was developed that translates a dbc file into a Simulink model featuring inbound messages (emitted by other bus subscribers) as sources and outbound messages (to be emitted by the device implementing the model) as sinks. Thus there exists the possibility to include a communication specification—as shown in Fig. 2.8—provided by the Hardware/Software-Integrator directly into the models by the Application or Software Engineer in a less error-prone fashion.

All models in the iCOMPOSE project have been developed in MATLAB and Simulink, yet it makes sense to use co-simulation for disconnecting components from another. It is thus possible to establish a clear interface, reduce model complexity and enable the easy replacement of individual parts by newer versions or variants. In co-simulation, an overall system is divided into several component models that can be modeled in their original, (possibly) different simulation tools or versions (“multi-tool”). During computation each component is provided its respective solver (“multi-method”). The solver used may also be customized with different time-steps (“multi-rate”).

The gradual integration of real components into the co-simulation environment at hardware/software integration level enables the testing of the iCEM, as well as several of its controlled components, in a virtual environment before they are used in the entire system and interacting with the products. For example, every modification to one function of the iCEM’s software is tested within a restricted environment designed to test this particular function on the target hardware by replacing its stimuli (sensor data) by results of a simulation and return the control commands in the simulation.

The correct co-simulation on model, model integration and software level of the digital prototype ultimately enables analyzes that may be difficult or impossible to carry out on physical prototypes, such as parameter sweeps and testing close to or out of valid bounds.

2.6 Conclusion

Systems engineering is a key aspect to coordinate the different development and generate a level of common understanding between engineering experts. The challenges are to gather and to manage the information efficiently and to generate an added value related to the generation and maintenance of this knowledge.

In the context of the INCOBAT project a method for the integration of system and SW models was developed. This approach consists on the refinement of a modeling profile and on the implementation of tool bridges toward SW development frameworks.

The SysML model based approach presented in the context of iCOMPOSE project, showed to be useful. However using standard SysML profile looks to be not fully sufficient for the purpose of functional safety engineering, and what's more, requirements on traceability between artefacts created during the development process are highly demanding. The fulfillment of the demanding requirements imposed on the traceability can be tool dependent.

Furthermore, based on the SysML model a toolchain was developed to generate code for MATLAB from class diagrams and state machines. To enable communication between models in generated code on the target platform CAN and UART interface blocks were devised for Simulink. These interface blocks may be automatically generated from a VECTOR-dbc file based CAN-specification, thus easing the specification exchange between Hardware/Software-Integrators and Application and Software engineers. The integration of developed toolchain with higher level conceptual models was achieved.

Finally, we provided a brief outline of the testing environment using co-simulation as a fundamental tool for testing at every development stage.

Acknowledgements The research work of the authors has been funded by the European Commission within the projects Integrated Control of Multiple-Motor and Multiple-Storage Fully Electric Vehicles (iCOMPOSE) and INCOBAT under the European Union's Seventh Framework Programme (FP7/2007–2013) under grant agreement №. 608897 and №. 608898.

VIRTUAL VEHICLE Research Center is funded within the COMET—Competence Centers for Excellent Technologies—programme by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET programme is administrated by FFG.

References

1. <http://www.i-compose.eu/iCompose/>
2. <http://incobat-project.eu/>
3. ISO-26262 Road vehicles—functional safety, Part 6: product development at the software level
4. ISO-26262 Road vehicles—functional safety, Part 10: guideline on ISO 26262
5. ISO-26262 Road vehicles—functional safety, Part 3: concept phase
6. ISO-26262 Road vehicles—functional safety, Part 4: product development at the system level
7. Szymanski D, Dexters B, Descas Y, Van Vlimmeren M (2014) Model based and scalable functional safety engineering methodology for on- and off-highway vehicles, FISITA 2014—Maastricht (NL)
8. TriCore AURIX family 32-Bit AURIX safety manual, AP32224, Infineon
9. AURIX TC27x 32-Bit single-chip microcontroller, user's manual

Comprehensive Energy Management - Safe Adaptation,
Predictive Control and Thermal Management

Watzenig, D.; Brandstätter, B. (Eds.)

2018, X, 112 p. 80 illus., 60 illus. in color., Softcover

ISBN: 978-3-319-57444-8