ISO 26262: Better be safe with modelling and simulation on system-level

Joachim Hößler, ikv++ technologies ag, Berlin, Germany (*hoessler@ikv.de*)

Sven Johr, TWT GmbH Science & Innovation, Stuttgart, Germany (*sven.johr@twt-gmbh.de*)

Thang Nguyen, Infineon Technologies Austria AG, Villach, Austria (*thang.nguyen@infineon.com*)

Stephan Schulz, Fraunhofer Gesellschaft IIS/EAS, Dresden, Germany (*stephan.schulz@eas.iis.fraunhofer.de*)

Gert-Jan Tromp, Dizain-Sync B.V., Borne, Netherlands (*g.j.tromp@dizain-sync.com*)

***Abstract*— In this article, we propose an ISO 26262 compliant modelling and simulation workflow in order to ensure an item's functional safety. With the proposed workflow, we close the gap between different heterogeneous modelling languages that are used in the distributed development chain in the automotive domain. We propose to use SysML, SystemC and SystemC-AMS. The system design will be done with SysML and subsequently, FTA and FMEA will be used by a team of experts for safety analyses of that SysML model. We propose to use IP-XACT's hierarchical components in order to map results of the FTA, i. e. potential weak points, to architectural interfaces in order to conduct fault injection. These architectural interfaces and elements are modelled and simulated within the tool COSIDE® that uses SystemC and SystemC-AMS as modelling and simulation languages. We include a holistic strategy on information management that uses standard formats as, e. g., IP-XACT's vendorTag or ReqIF, for exchanging, tracing and allocation of safety requirements. Our modelling and simulation workflow has been manually demonstrated on a real-life example of an airbag system application.**

Keywords—functional safety; ISO 26262; Modelling; Simulation; Workflow; System development; Safety analysis

I. Motivation

Since the introduction of the ISO 26262 standard, today's development behavior of automotive industry is strongly affected to a safety oriented working culture. Furthermore, the number of functionalities and complexity with high integration is rapidly increasing in modern automotive electronics making mixed-signal System on a Chip (SoC) design become trend in automotive nowadays. On top of that, high pressures to produce reduced time-to-market, first-time-right design are additional key elements. These facts pose great challenges for designers at different design level from system to hardware components, embedded software. An effective utilization of modelling and simulation facilitates the design process so that these challenges can be met. As such, this work focuses on developing an ISO 26262 compliant modeling and simulation workflow supporting system level development of safety-related items, including mixed-signal hardware component and custom embedded software at micro-controller side.

II. Paper contribution

This paper's major contribution is a modelling and simulation workflow that is ISO 26262 compliant and bridges the gaps between different modelling techniques and development steps by the following activities:

1. *functional safety analysis* including Hazard Analysis and Risk Assessment (HARA) in order to derive the safety requirements;
2. *functional modelling and allocation* of functions to system components;
3. *identification of malfunctions.*

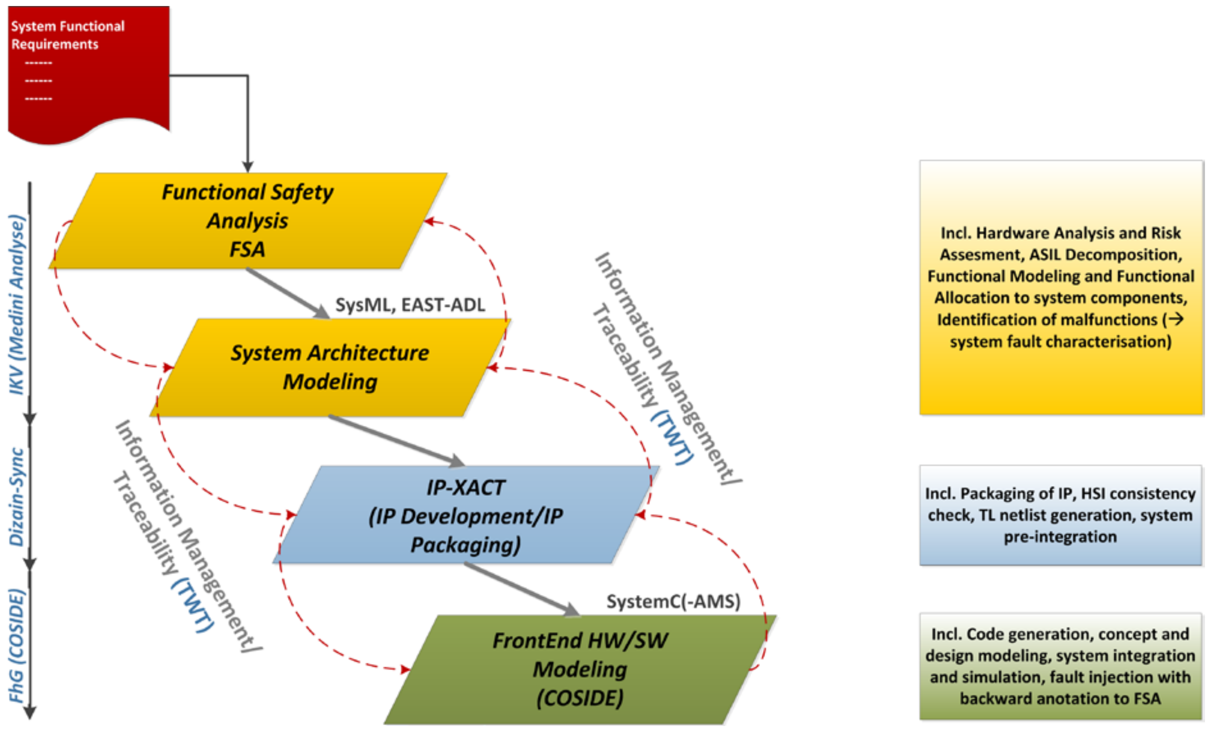
For instance, the identified malfunctions are used to derive input for fault injection activities (required by the ISO 26262 standard) and, thus, to simulate the system under faulty conditions caused by defects or electromagnetic interference. Based on the simulation results appropriate safety mechanisms are defined and developed. Subsequently, the previously defined fault injection runs are used in order to validate the developed safety mechanisms.

Moreover, the proposed modelling workflow should enable traceability of requirement changes and management from system level down to hardware/software component level and back. This is not only one of the most important requirements of the ISO 26262 standard but also helps to strengthen the consistency of the information flow from multidisciplinary design teams.

III. Modelling and simulation workflow-Overview

Depicted in Figure 1 is the overview of the proposed modelling and simulation workflow. One of the challenges for a (generic) workflow is the heterogeneity in terms of languages and tools used across the various design and development stages. For example, the functional design is often done informally in functional dependency diagrams (e.g. SADT in Visio, PowerPoint, or Excel), while the system architecture uses a (semi-) formal modelling language (e.g. SysML, EAST-ADL). Our workflow utilises SysML, SystemC, and SystemC AMS as modelling languages. Each language is supported by a specific tool to develop the system.

medini analyze® from IKV is used to conduct the functional safety analysis, from the hazard analysis to the functional safety concept and to the safety architecture, including functional allocation and ASIL decomposing at the later step. The system architect is then modelled using e.g.: SysML as shown in Figure 3. The tool includes as well the capability of mapping requirements into subsystem/system architect models. SystemC and SystemC-AMS will be supported by COSIDE® (Complex System Integrated Development Environment) for subsystem components development to system integration, its simulation, and verification activity. The workflow is highly automated using packaging of IP with the IEEE standard IP-XACT allowing automatic generation of Hardware Software Interface (HSI) interconnects, test bench and verification environment and verification software.


System Functional Requirements

Functional Safety Analysis FSA
SysML, EAST-ADL
System Architecture Modeling
IP-XACT (IP Development/IP Packaging)
SystemC(-AMS)
FrontEnd HW/SW Modeling (COSIDE)
Information Management/ Traceability (TWT)
Information Management/ Traceability (TWT)
IKV (Medini Analyse)
Dizain-Sync
FhG (COSIDE)
Incl. Hardware Analysis and Risk Assesment, ASIL Decomposition, Functional Modeling and Functional Allocation to system components, Identification of malfunctions (→ system fault characterisation)
Incl. Packaging of IP, HSI consistency check, TL netlist generation, system pre-integration
Incl. Code generation, concept and design modeling, system integration and simulation, fault injection with backward anotation to FSA

Figure 1: Overview of the proposed modeling and simulation workflow

For the workflow proof-of-concept and demonstration purpose, an airbag system application is taken.

IV. Workflow Details

A. *Concept phase*

1) Item Definition

The Item Definition sets the overall scope of modelling as well as boundaries (interfaces) to other systems (see ISO 26262-3, section 5). This marks the outermost boundary for simulations of the system and clearly states context dependencies that need to be modelled. This should already includes a rough system architecture and a description of the main functions of the system. Additionally, all relevant information on the item are collected, such as design study documents, predecessor specifications, applicable standards, product specific best practices, regulations and laws, applicable statistics and so on.

2) HARA

Hazard Analysis and Risk Assessment (HARA) provides the top level hazards for investigations (see ISO 26262-3, section 7). In the same manner as the item definition sets the scope for modelling and simulation, hazardous events will drive the search for potential malfunctioning behaviour/failures of the system causing system hazards.

During the HARA, system malfunctions (derived e.g. by a HAZard and OPerability study (HAZOP)) are analyzed by considering them in relevant driving situations. The relevant driving situations are selected from a situation catalog (typically setup at an OEM).

The next step is to assess the previously identified hazardous events by means of severity and controllability (exposure is taken from the catalog). For each hazardous event specify the hazard or worst case effect. Severity is an estimation of the harm caused by a hazard. Controllability estimates the countermeasures the driver (or any other traffic participant involved in the hazardous event) can take to prevent the hazard from occurring. Note that both factors have to be considered independently and any mitigation actions that influence the controllability shall not be considered when estimating the severity.

3) Functional Safety Concept

The Functional Safety Concept defines the safety functions that will be implemented by the system as well as the warning and degradation concept (see ISO 26262-3, section 8).

For all hazardous events that have an ASIL different than QM safety goals have to be defined and assigned. A safety goal is mainly characterized by being the inversion of one or multiple hazards. Typically, one safety goal will be shared among a number of hazardous events and describes prevention and/or mitigation aspects. For our Airbag example, the most important safety goal is "Prevent unintended deployment" which is usually ASIL D in real systems. Safety goals are the top-level safety requirements and subsequent requirement break-down starts with the same in order to develop the Functional Safety Concept (FSC). The FSC defines the overall strategy how the safety of an item is achieved on a functional level. This means mainly the derivation of functional safety requirements from safety goals and their allocation to elements in the system architecture. Since safety goals have a descriptive nature (the what), the functional safety requirements specify the realization of the goals (the how) without prescribing a specific technological approach. The latter will be done in the Technical Safety Concept (TSC). The core of information captured by the functional safety requirements address different aspects of safety including transitioning to safe states, fault detection and failure mitigation mechanisms, fault tolerance and fault tolerant time intervals, warning and degradation concept (here: sensors and processing).

B. *System Level Development*

1) Technical Safety Concept

The TSC defines how the FSC is realized and makes the technological choices. Thereby, the implementation approach is defined, including the separation to HW and SW, taking into account the functional safety requirements and safety goals. This is mainly expressed by technical safety requirements that are allocated to the system design. Figure 2a shows some examples for technical safety requirements of the Airbag and also a SW safety requirement.

The system design itself is modelled in SysML. Figure 3 shows the technical architecture consisting of the sensor arrays (upfront/side impact, on-board g-sensors), the combined SoC part with SPI interface to the main micro, voltage regulator, watchdog, and actuators.

For the modelling and simulation workflow, the interesting question for safety and system engineers is whether certain failures are effectively handled by safety mechanisms that have been considered. For this purpose, safety analyses are conducted usually as a refinement process of the system design. This process is structured and should be driven by techniques such as Failure Modes and Effects Analysis (FMEA) or Fault Tree Analysis (FTA), and it involves experts from OEM and supplier. For our airbag example, a fault tree (see Figure 2b) has been developed starting with the top-level event Unintended deployment (hazard imposed by the safety goal under consideration).

As can be seen from the fault tree, the SPI communication was identified as a potential weak point that needs to be further investigated (undeveloped event E15). Hence, the decision was to do a fault injection simulation in COSIDE® as further explained in section 4.4. For this purpose the system and failure information needs to be exchanged between the tools.

C. System architecture exchange with IP-XACT

IP-XACT (IEEE standard 1685-2009) is an XML format to de ne and describe electronic components, designs and flows. The use of IP-XACT in our flow has a purpose both to document and to automate. The IP-XACT standard describes (amongst others) the documentation of components:

- bus interfaces, to describe the mapping of component signals to standard bus interface
- signals models, to describe different levels of abstraction
- files, to support automated flows for simulation and synthesis, but also fault injection

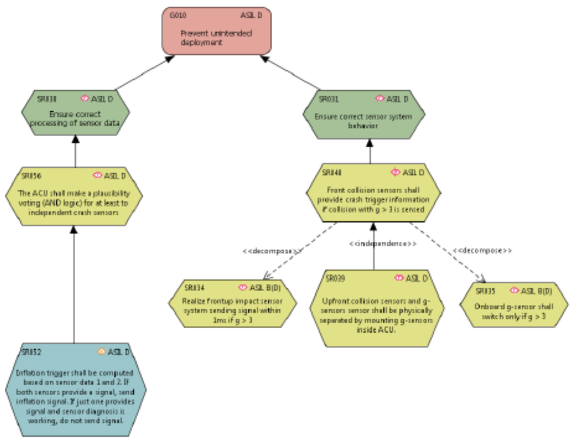
The IP-XACT standard does not cover safety or fault injection, but uses the "vendorExtension" tag that is allowed on specific location within the IP-XACT XML and provides a container for inserting the required information. The use of IP-XACT for the integration of SoC is well known (see for example [4]). The advantages are a reduction of the integration time as well as increase of the verification quality as automation helps there. For an automotive chip design workflow traceability of requirements and safety goals is critical throughout the design, verification and validation process. The "V"-model is commonly used in the automotive industry (see Figure 4a). As shown in Figure 4a, IP-XACT is used to determine the vertical coverage, when the hierarchical decomposition of the design is represented by IP-XACT hierarchical components and designs:

- e.g. requirements that are decomposed and implemented in components of the system (both the ReqIF ID and UUID can be used for this)

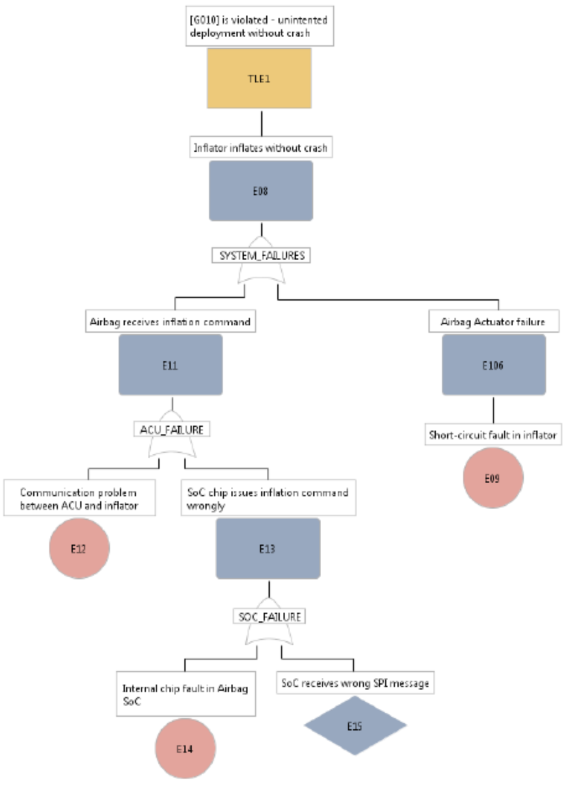
and also for horizontal coverage, when the requirements are added to the IP-XACT xml in the manner that will be explained in the next paragraph:

- e.g. to document for each test case, which requirements are tested

In our workflow, we have used the verdorExtensions mechanism of IP-XACT to map in a top-down manner the undeveloped event E15 from the FTA (see Figure 2b) to the SPI slave interface of the Airbag SoC using the faultsimulation tag (see Figure 4b).


G010 ASIL D
Prevent unintended deployment
SR038 ASIL D
Ensure correct processing of sensor data
SR031 ASIL D
Ensure correct sensor system behavior
SR056 ASIL D
The ACU shall make a plausibility voting (AND logic) for at least to independent crash sensors
SR048 ASIL D
Front collision sensors shall provide crash trigger information if collision with g > 3 is sensed
<<decompose>>
<<independence>>
<<decompose>>
SR034 ASIL B(D)
Realize frontup impact sensor system sending signal within 1ms if g > 3
SR039 ASIL D
Upfront collision sensors and g-sensors sensor shall be physically separated by mounting g-sensors inside ACU.
SR035 ASIL B(D)
Onboard g-sensor shall switch only if g > 3
SR052 ASIL D
Inflation trigger shall be computed based on sensor data 1 and 2. If both sensors provide a signal, send inflation signal. If just one provides signal and sensor diagnosis is working, do not send signal.

(a) Main safety goal "Prevent unintended deployment" with excerpt of functional and technical safety requirements


[G010] is violated - unintented deployment without crash
TLE1
Inflator inflates without crash
E08
SYSTEM_FAILURES
Airbag receives inflation command
E11
Airbag Actuator failure
E106
ACU_FAILURE
Short-circuit fault in inflator
E09
Communication problem between ACU and inflator
E12
SoC chip issues inflation command wrongly
E13
SOC_FAILURE
Internal chip fault in Airbag SoC
E14
SoC receives wrong SPI message
E15

(b) Initial fault tree analysis: undeveloped event "SoC receives wrong SPI message" [E15] needs to be further investigated by simulation

Figure 2: Safety analyis with *medini analyze*®

D. FrontEnd HW/SW Co-Modelling with COSIDE®

In this workflow COSIDE® is used to create hierarchical multi-domain mixed-signal designs by using SystemC and SystemC AMS. The designer can implement the functionality, simulate it, and then check against the requirements by means of verification.

1) Modelling

After continuing from the previous analysis step the designer gets an prefilled SystemC module (see example in Fig. 5) which he can use as a starting point for implementing the Item's functionality. Additional information provided by the IP-XACT input are used to further generate additional modules and netlists if already defined to present a seamless transition from definition to implementation. The multi-domain ability enables the designer to combine different MoCs and domains for a complete representation of an Item instead of creating multiple versions for each single domain. In this airbag example the multi-domain modelling can be used to simulate the car's movement related to another object for starting the airbag system.

2) Simulation

Simulation is achieved by using the Accellera SystemC implementation and the proof-of-concept implementation of the Accellera SystemC AMS standard of Fraunhofer IIS-EAS. This guarantees interoperability with existing third-party IP and enables the current design to be a standard compliant third-party IP itself e.g. for use as a Security Element out of Context (SEooC). Simulation speed is dependent on the detail level but SystemC provides a broad range from TLM down to RTL level for modelling so that unnecessary details can be avoided to gain simulation speed.

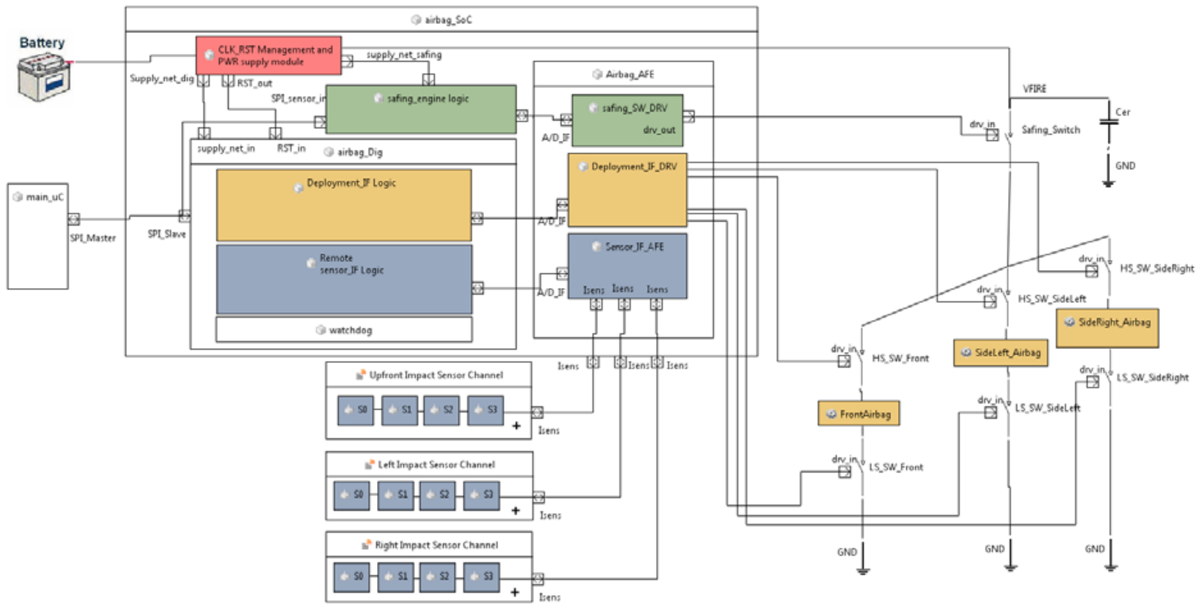


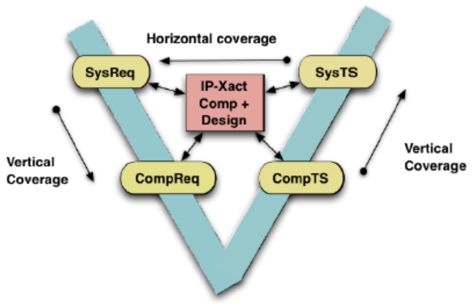
Figure 3: Technical system architecture modelled with SysML in *medini analyze*®

3) Verification

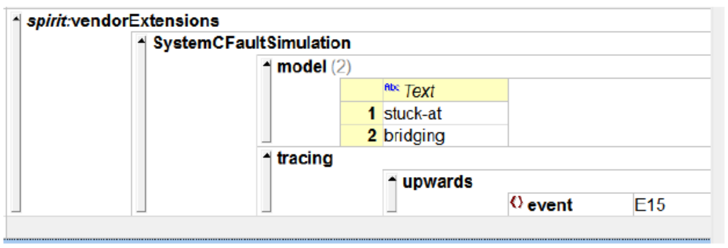
To verify the simulation against the requirements the designer has to implement regression tests which map to the requirements and issues found during the analysis step. ISO 26262 requires fault injection depending on the Automotive Safety Integrity Level (ASIL) level which has to be achieved. COSIDE® provides a fault injection library to cover this part of the standard which enables the designer to check if the behaviour of the implementation is robust or safe enough to match a certain ASIL level. The fault-insertion facility comes with predefined faults but is extensible to provide design-tailored high- level faults, e.g. intermittent value changes at higher abstraction levels than RTL netlists. In the future an automated test case generation sourcing from the requirements to create a complete regression test is planned.

E. Information management considerations

In this section, we highlight the challenges for a holistic information management strategy w.r.t. safety information, i. e., Safety Goals (SGs), Functional Safety Requirements (FSRs), Technical Safety Requierments (TSRs). As highlighted in Section 4.1, the SGs, FSRs/FSCs and TSRs/TSCs are directly related as, for example, the FSRs are derived from the SGs. Here, the two main challenges are: (A) traceability of safety information, i. e., bi-directional traceability from SGs to FSRs, and from FSRs to TSRs; (B) allocation of safety information, i. e., linkage of TSRs to development artefacts of the safety mechanisms. Both traceability and allocation can be seen from two di_erent perspectives. First, from a technical perspective where the main challenge is the technical linkage of, e. g., SGs and FSRs. In theory, the technical issue of linking artefacts in a bi-directional way is solved very well, see also [1], and [2]. Different possibilities using Uniform Resource Identifiers (URIs) or even database-approaches exist.



(a) V-model using IP-XACT


spirit:vendorExtensions
SystemCFaultSimulation
model (2)

	Text
1	stuck-at
2	bridging

tracing
upwards

event	E15

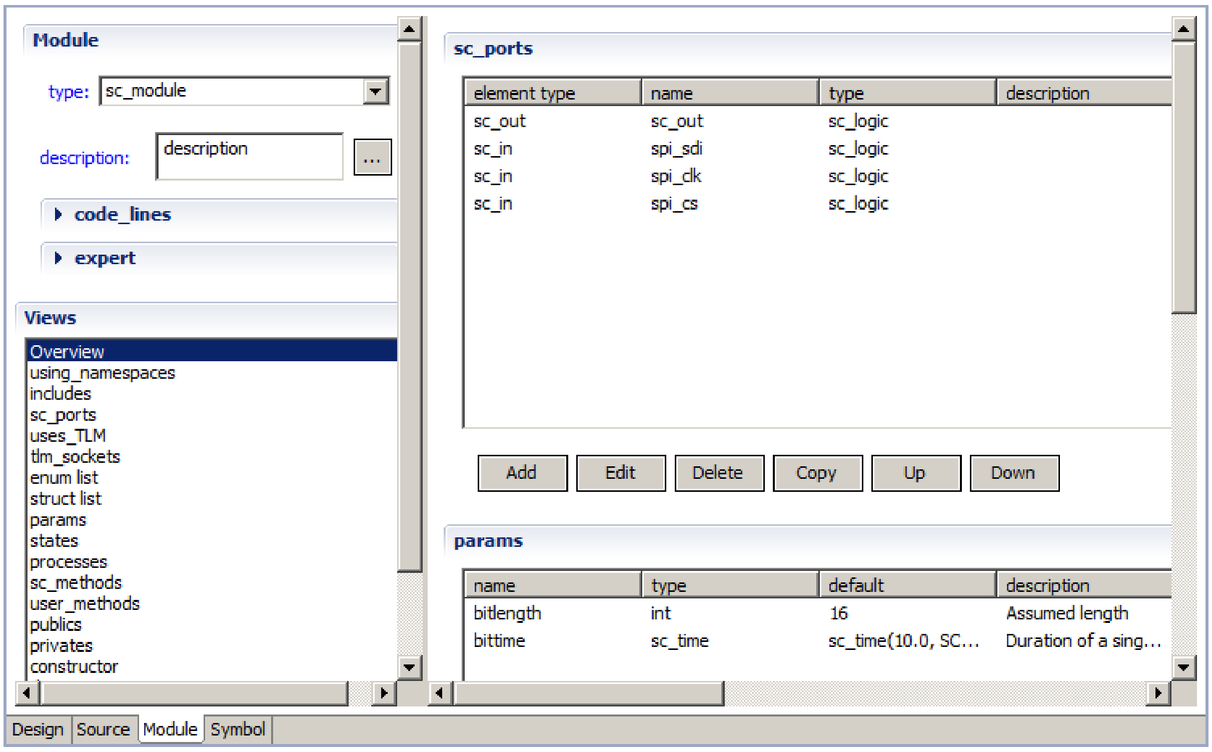
(b) vendorExtensions in SPI_SLAVE busInterface in component

Figure 4: V-model using IP-XACT for safety goals and requirements

However, in practice and especially in a complex development environment as given in the automotive domain, the simple linkage of artefacts remains a challenge. Due to space limitations, we do not further elaborate on this topic. However, we consider it of utmost importance to use an open and standardised format as, e. g., Requirements Interchange Format (ReqIF), for exchanging safety goals and safety requirements in a simple *export - send - receive - import* workflow. The introduction of ReqIF as an interchange format for safety requirements raises by itself different challenges. Tool vendors of requirements management tools have to implement, say, interfaces from their own proprietary used format to ReqIF and, even more important, no additions to the standard shall be made by a single tool vendor: that would jeopardise a smooth exchange of information with a simple *export - send - receive - import* workflow. In the following, we consider a solution for allocation of safety information by using unique identifiers for, e. g., SGs. Thus, it is a precondition, that unique identifiers for that kind of safety information have already been generated by, e. g., appropriate tooling. We include the safety information in its ReqIF format in IP-XACT by using the unique identifier. This reference point is also directly included in the prefilled SystemC modules. Thus, establishing the allocation of safety information down-to architectural and implementation elements of the item's model is, under the precondition of unique identifiers, straight forward.

Besides establishing the allocation of safety information, it has to be monitored whether: (1) SGs and FSRs/TSRs are completely covered, e. g., each SG is traced down-to at least one TSR and that TSR is allocated to at least one implementation component; (2) information is changing.

For (1), different proprietary solutions as, e. g., Reqtify [5], asureSign [6], RequisitePro [3], are available. To the best of our knowledge, it remains to include these solutions into the concrete development process at, say, the developers' side in order to use this tooling in the background. Point (2) is, say, simply solved by using an appropriate version control system as, e. g., cvs [7], svn [8], git [9], MKS integrity [10]. A concrete solution is completely dependent on the development process of the manufacturer. In order to establish a meaningful change management solution, the used version control system has to be neatly integrated by, e. g., customised solutions that use the version control as a background activity. However, these solutions currently lack the possibility to make changes directly available in all of the development artefacts. For that, the research project Crystal [11] is deemed to produce valuable results.



type: sc_module

description: description

Overview
using_namespaces
includes
sc_ports
uses_TLM
tlm_sockets
enum list
struct list
params
states
processes
sc_methods
user_methods
publics
privates
constructor

sc_ports

element type	name	type	description
sc_out	sc_out	sc_logic	
sc_in	spi_sdi	sc_logic	
sc_in	spi_clk	sc_logic	
sc_in	spi_cs	sc_logic	

params

name	type	default	description
bitlength	int	16	Assumed length
bittime	sc_time	sc_time(10.0, SC...	Duration of a sing...

Figure 5: Example of an imported IP-XACT packaged item

V. Conclusion

In this article, we have introduced our ISO 26262 compliant modelling and simulation workflow. The workflow follows the methodology prescribed by the ISO standard and uses tools and languages as, for instance, *medini analyze*®, IP-XACT, COSIDE®, ReqIF and Reqtify.

In summary, HARA is considered on a SysML model present in the tool *medini analyze*® and the results of the analysis are used with IP-XACT in order to handle over those results to the simulation tool COSIDE®. For simulation within COSIDE®, prefilled SystemC modules based on the SysML model are generated. Moreover, modules and netlists are generated from the IP-XACT description such that a seamless transition from definition to implementation is given. In our workflow, the safety information ranging from HARA, see Section 4.1.2, down to the system design in SysML, see Section 4.2.1, is exported in the ReqIF format and will be used within IP-XACT as reference point within the “vendorExtension". For the airbag SoC the undeveloped event E15 from the FTA, see Figure 2b, the related safety information, i. e., TSRs that are allocated to the system's safety architecture, is exported in ReqIF and then used in the IP-XACT description as a reference point within the XML description. We use the “vendorExtension" with its get and set commands as given in the Intellectual Property-XACT (IP-XACT) standard for that.

Acknowledgement

The research leading to these results has received funding from the ARTEMIS Joint Undertaking VeTeSS under grant agreement no 295311.

References

[1] D. Cuddeback, A. Dekhtyar, and J. H. Hayes. Automated Requirements Traceability: The Study of Human Analysts. In Requirements Engineering Conference (RE), 2010 18th IEEE International, pages 231 - 240, September 2010.

[2] O. C Z Gotel and A. C W Finkelstein. An analysis of the requirements traceability problem. In Requirements Engineering, 1994., Proceedings of the First International Conference on, pages 94 - 101, April 1994.

[3] IBM. IBM - Rational RequisitePro. http://www-03.ibm.com/software/products/de/reqpro, 2014.

[4] W. Kruijtzer et al. Industrial IP Integration Flows based on IP-XACT standards. In Design, Automation and Test in Europe, 2008. DATE '08, pages 32 - 37, March 2008.

[5] Dassault Systèmes. CATIA Systems Engineering - Reqtify. http://www.3ds.com/de/produkte-und-services/catia/funktionsumfang/catia-systementwicklung/requirements-engineering/reqtify/, 2014.

[6] TVS. Requirements | TVS. http://testandverification.com/solutions/requirements/, 2014.

[7] http://www.nongnu.org/cvs/

[8] http://subversion.apache.org/

[9] http://git-scm.com/

[10] http://www.mks.com

[11] http://www.crystal-artemis.eu/