CYBER-PHYSICAL SYSTEMS ENGINEERING: MODEL-BASED SOLUTIONS

Alfredo Garro
Vittorio Vaccaro

Department of Informatics, Modeling, Electronics
and Systems Engineering (DIMES)
University of Calabria
via P. Bucci 41C, 87036 Rende (CS), Italy
{alfredo.garro, vittorio.vaccaro}@unical.it

Stefan Dutré
Jef Stegen

Siemens Industry Software
Interleuvenlaan 68
B-3001 Leuven, Belgium

{stefan.dutre, jef.stegen}@siemens.com

ABSTRACT

As cyber-physical systems become more widespread, the scale and complexity of hardware and software is increasing dramatically. Requirement Engineering plays a main role in this field. It gives a better vision of the problem, facilitates the development of the components, makes the final product better and facilitates business choices. In this context, a lot of work is already being carried out to improve the functionalities offered to designers for the development of systems. This paper shows how to improve the efficiency of cyber-physical system development by redefining some phases of the underlying systems engineering processes. It also shows how the complexity that characterizes these processes can be reduced with the use of innovative techniques for the integration and visualization of system models.

Keywords: Model-based Systems Engineering, Requirements Management, Integration and Visualization, Safety Analysis.

1 INTRODUCTION

In the context of system development we are witnessing technological innovation. The most advanced systems involve aspects of an interdisciplinary nature, merging the theory of cybernetics, mechatronics, design and process science. Cyber-Physical Systems (CPS) are becoming increasingly important. A CPS is an integration of computation with physical processes whose behavior is defined by both cyber and physical parts of the system. In cyber-physical systems, physical and software components are deeply intertwined, each of which operates on different spatial and temporal scales, exhibiting multiple and distinct behavioral modes and interacting with one another in many ways that change with the context. The additional benefit offered by cyber-physical systems will lead to radical changes in many application fields (e.g. energy, mobility, health care) and will have an impact on our daily life and how it is affected by the software. Today, in a market where rapid innovation is assumed, engineers of all disciplines must be able to explore system projects collaboratively, assigning responsibility to software and physical elements and analyzing trade-offs between them. The design of a complex cyber-physical system, in particular one with heterogeneous subsystems distributed over networks, is a challenging task. The commonly used design techniques are sophisticated and include mathematical modeling of physical systems, formal computational models, simulation of heterogeneous systems, software synthesis, verification, validation and testing. In this context, the paper presents model-based solutions to improve the efficiency of cyber-physical system development by redefining some phases of the underlying systems engineering processes with particular focus on requirements management, models integrations and safety analysis. How the complexity that characterizes these processes can be reduced with the use of innovative techniques for the integra-

tion and visualization of system models, it is also showed. The rest of the paper is structured in five sections. Section 2 reports a brief introduction on MBSE and how it can be helpful in the design and implementation of cyber-physical systems. In Section 3 the key principles of requirements management are described along with proposals to facilitate the process by acting on their status. Section 4 discusses on Models Integration and specifically on how to reduce the complexity of system design through the visual representation of the models. Section 5 is centered on Model-Based Safety Analysis: the basic concepts of system safety analysis are introduced along with the exploitation of models exchange features between different software tools to ease its execution. Finally, conclusions are drawn and future work delineated.

2 MBSE FOR CYBER-PHYSICAL SYSTEMS ENGINEERING

The growing complexity of CPS directly affects their management. This complexity derives from the different nature of the components involved in the system which makes their integration challenging. To date, several techniques exist to manage this complexity (Bocciarelli et al. 2016), but important improvements must be made in this field in the near future. In this regard, the MBSE approach is a key enabler. MBSE is the practice of developing a series of related system models that help define, design and document a system under development (Micouin 2013). These models provide an effective way to explore, update and communicate aspects of the system to stakeholders, significantly reducing or eliminating dependence on traditional documents. This approach to modeling introduces further aspects of importance not less than those just mentioned, such as: traceability, competition support and distributed teams, incremental, iterative and parallel life cycle development. Data-centric specifications enable automation and optimization, allowing system engineers to focus on value-added activities and ensuring a balanced approach. By bringing together different but related models, first-time levels of system understanding can be achieved through integrated analysis. Below are the main advantages of the MBSE:

- Broad in scope, across multiple stages of system development and multiple physics.
- Allows for the development of virtual prototypes.
- Facilitates communication among disciplines in team-based development.
- Enables semi-formal and formal approaches to system assessment.
- Management of system complexity.

All models and processes of systems engineering are based on the life cycle concept. The life cycle phases are defined by the international standard ISO\IEC 15288. The V model shown in Figure 1 is a common graphic representation of the life cycle of systems engineering. The left side of the V represents the development of the concept and the breakdown of the requirements into functional and physical entities that can be designed and developed. The right side of the V represents the integration of these entities (including appropriate tests to verify that they meet the requirements) and their definitive transition in the field, where they are managed and maintained.

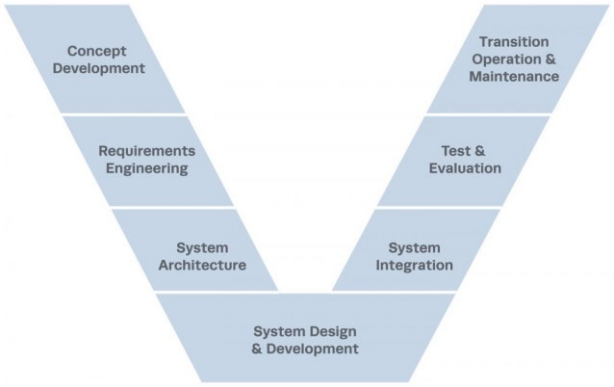

Figure 1: INCOSE V model.

For supporting system modeling throughout the above sketched process, the reference language is SysML (Systems Modeling Language). SysML is defined as an extension of a subset of the Unified and Modeling Language (UML) using the UML profile mechanism. SysML uses the XML Metadata Interchange (XMI) to exchange model data between the tools and is also intended to be compatible with the constantly evolving ISO 10303-233 engineering data interchange standard. Figure 2 shows the hierarchy of SysML diagrams.

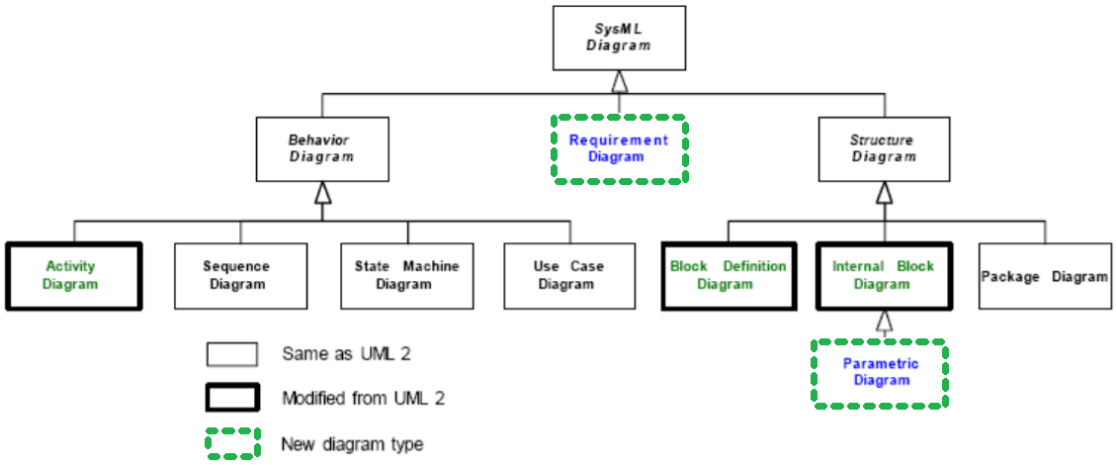

Figure 2: Hierarchy of OMG SysML Diagrams.

3 MODEL-BASED REQUIREMENTS MANAGEMENT

In this Section the key principles of requirements management are described along with proposals to ease the process by acting on their status. In particular, after a brief description of the basic concepts for requirements management, a new lifecycle implemented on the Polarion platform to increase the efficiency in requirements management is presented.

3.1 Requirements Management

The increasing complexity of cyber physical systems leads to an increased effort required of designers and engineers in terms of understanding and solving the problems encountered. Requirements management plays a fundamental role in the design and implementation of systems. Requirement Management is a systematic approach to research, documentation, organization and monitoring of the evolving require-

ments of a system. The collection of requirements may seem a rather simple task. However, in realistic projects difficulties will easily arise; the following characteristics of requirements lead to these difficulties:

- The requirements are not always obvious and can come from many sources.
- The requirements are not always easy to express clearly in words.
- There are many different types of requirements at different levels of detail.
- The number of requirements can become unmanageable if not controlled.
- The requirements are related to each other and also to other results of the software engineering process.
- The requirements have unique properties or characteristic values, for example, they are neither equally important nor equally easy to satisfy.
- There are many stakeholders, which means that the requirements must be managed by groups of cross-functional people.
- The requirements are constantly evolving.

Thus, the following concepts need to be encompassed in the requirement management phase: problem analysis, understanding stakeholder needs, system definition, project goal management, refining system definition, requirements evolution management, requirements validation and verification (Aiello et al. 2017, Garro et al. 2016).

3.2 Polarion

One of the solutions to manage requirements can be found in Polarion. Thanks to its features it allows to manage different types of requirements, easily change the fields that compose them and the relationships between them within the project to which they belong. Polarion also provides mechanisms that allow to customize the different stages of requirements management in order to satisfy users' needs.

3.2.1 Requirements Management

Polarion ALM uses work items to manage different types of requirements. The work items are composed by a series of fields that contain data relevant to the corresponding requirement. Work items in Polarion can be customized, according to designer needs. Requirements are characterized by default fields (type, status, author, assignee, priority, description, etc.) and for the scope of the use case by custom fields (safety related, DAL level, validation compliance, etc.). The following fields have particular importance for the purposes of the paper:

- Status: it represents the current state of the requirement that can be: "Draft", "In Review", "Approved", "Rejected", "Deleted".
- Validation compliance: it represents the validation status of the requirement (i.e. that the requirement is properly constructed, traced and compiled); it can take false, true and partially values.
- Linked work item: it represents the list of requirements related to the current requirement, defining parent/child relations.

The requirements in Polarion are subject to continuous changes dictated by the needs of the project. Users can modify the fields if they have the necessary permits to make the changes.

3.2.2 Requirements Life Cycle

This paper shows how the requirements management has been improved in Polarion by introducing a clear structure supporting the life cycle of requirements (see Figure 3). This made it possible to automate some operations performed on it, which would otherwise take time and a lot of manual work by the user. It also introduces control over who carries out the actions and also ensure that the requirement set remains more consistent by following the rules. It is possible to add conditions on the state of a requirement which have to be met before the status can be modified. Conditions refer to the current attributes of the requirement. If the conditions for change of status are satisfied, the status can be changed and also functions can be triggered that alter fields from the respective requirement. In general, a function is a list of actions that can change requirement attributes.

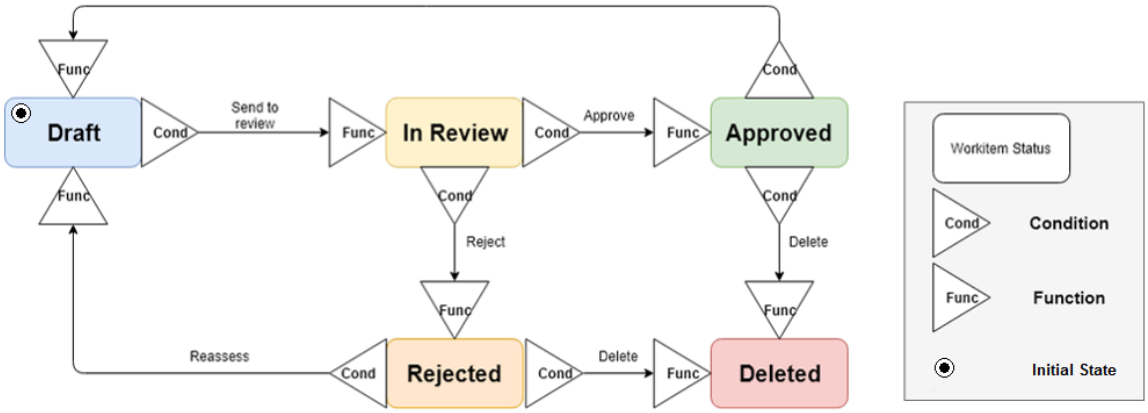

Figure 3: Work items statuses of requirement life cycle.

In the following, all the states (which are also shown in Figure 3) are considered. After creating a new requirement, the state is set to "Draft" by default. When a requirement is in this state, the user can modify the requirement by enriching it with information and modifying its representation. Once this is done the requirement can go to the state "In Review". From here the requirement can be rejected or approved. The requirement can be rejected if for example it is not complete, it can be replaced or it is badly structured. From the "Rejected" state, the designer can think about removing the requirement because it is no longer useful for the project or sent back in draft to make changes that will then have to be approved. The transition from "In Review" to the "Approved" state makes use of a checklist on the requirement quality as specified in the ARP4754 standard. ARP4754A (Guidelines For Development Of Civil Aircraft and Systems), is a guideline from SAE International that discusses the development of aircraft systems taking into account the overall aircraft operating environment and functions. This includes validation of requirements and verification of the design implementation for certification and product assurance. It provides practices for showing compliance with the regulations and serves to assist a company in developing and meeting its own internal standards by considering the guidelines herein. To proceed to the "Approved" state the work item in question must satisfy a checklist, as is described in the ARP4754A standard, concerning the verification of the correctness of the requirements. The list of correctness checks for a requirement in Polarion is shown in Figure 4.

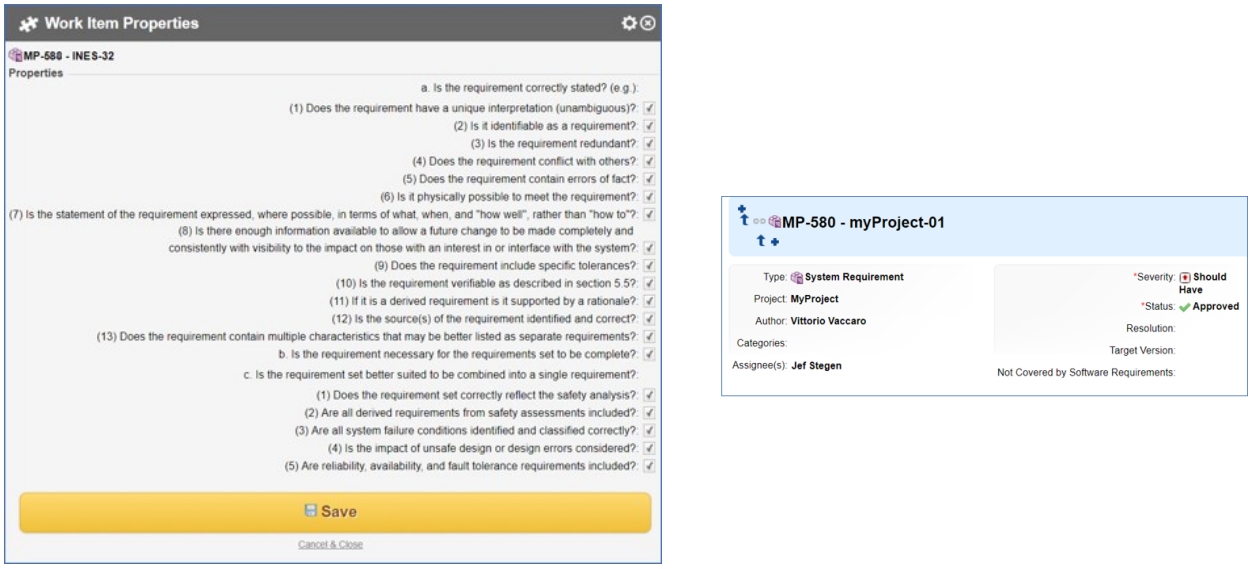

Figure 4: Correctness Check List of a requirement in Polarion (on the left) and the related work item with validation field true after the "Approved" state change (on the right).

If all answers to the questions are affirmative, the conditions for making the change are met. The functions that are performed are intended to automatically change the validation compliance field of the requirement from false to true and the status of the approved requirement, as shown in Figure 5. Of equal importance is the change of status in "Deleted". To move to this state the conditions must satisfy that the work item has no relationship to other work items (linked work items as children). The function in this phase automatically eliminate other relationships of the work item (linked work items as parents) and proceeds to change the status to "Deleted". In conclusion, it can be stated that the introduction of the life cycle of the requirements in Polarion has brought several advantages. It has made it possible to increase the consistency of the data. It allowed to follow the ARP4754A standard for the design of aircraft through the correctness check. It has allowed to automate some redundant actions during the requirements management phases within the project.

4 MODELS INTEGRATION

This Section deals with Models Integration and specifically on how to reduce the complexity of system design through the visual representation of models. In particular, after the presentation of the Sirius tool a meta-model based approach to support models integration is presented.

4.1 Models Integration

While designing and implementing systems, the system structure complexity keeps on increasing. In order to cope with this complexity, the designer needs to have available mechanisms or tools that simplify the overall view of the system (Falcone et al. 2017). Some preliminary effort has already been performed for development of a tooling prototype which features both system models, requirements and safety artefacts (Ratiu, Zeller, and Killian 2015, Burger et al. 2016). This paper proposes such mechanisms by making use of Sirius, an Open Source technology developed by Obeo and Thales to facilitate the creation of cutting-edge design solutions (Obeo 2019). Thanks to its functionalities, it provides a workbench for the creation of custom diagrams fitted to the user's need.

4.2 Obeo Designer

Obeo Designer is a tool that allows to easily create a graphic modeling workbench for specific domains, for example, industrial systems, software applications or the organization of large companies. The modeling workbenches created by Obeo Designer are composed of a set of editors (diagrams, tables and trees) that allow users to create and modify models in a shared repository. It uses Eclipse Sirius, an Open Source technology developed by Obeo and Thales to facilitate the creation of cutting-edge design solutions. It also makes use of the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF) for creating tools based on structured data models.

4.2.1 Metamodel

Often during the system design phase, many tools are used that contain different data sets within the same project (e.g. Matlab, Amesim, Polarion, Capella). Data from these different tools is related to each other so there is a need to integrate these models with each other. Sirius is used in this paper as a first step into integrating all this data. This is done by creating a meta model of the data mentioned above which allows to integrate the model themselves. Meta Models help in obtaining and maintaining a consistent data set (Meier and Winter 2018). Making use of meta models for system design has been leading to a 5X speedup (de Weck 2015). Thanks to Sirius native environment, it is possible to create a metamodel diagram similar to UML diagrams, in which the different relationships between the components are defined (such as composition, referencing, etc.) and store them in an *ecore* file. The metamodel created in Sirius is shown in Figure 5; the metamodel is represented with data from Polarion, Matlab, Amesim and Capella. Having a joined meta model of all different data from the various tools offers the advantage of being able to analyze the internal structure of each model and also looking for points in common between them.

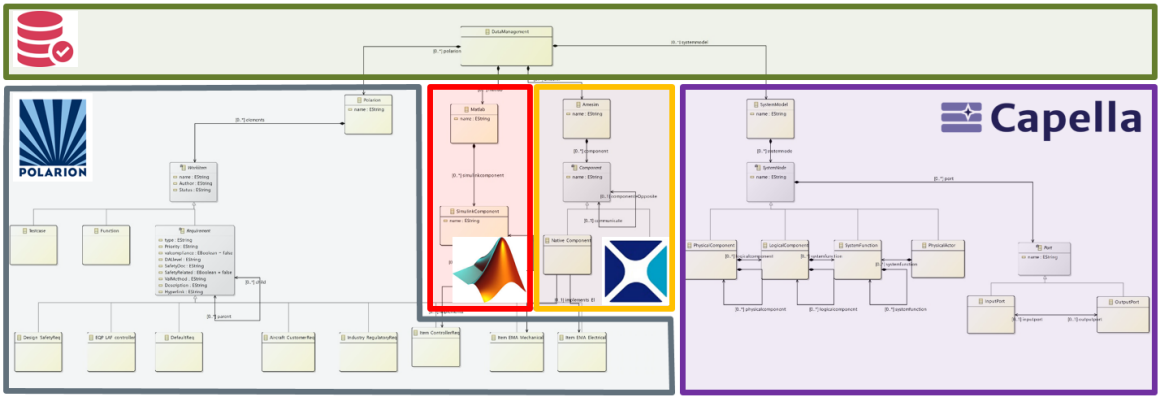

Figure 5: Metamodel in Sirius.

4.3 Sirius Visualization

A Sirius visual representation of a DX model is shown in Figure 6. DX is a software tool under development for the exploration of different architectures, owned by Siemens. DX models are stored in text format and then displayed in the tool in blocks (components) and arches (relations). To make changes to the system structure, the DX user can edit the text file associated with the system. Once the existing meta model was extended with the DX concepts, a series of configurations are carried out within Sirius that link concepts from the meta model with visual representations that will be shown to the end user. During these phases it is possible to set the behavior of the buttons, create import functions for the model, assign colors and shapes to entities and so on. Figure 6 shows the Sirius runtime environment as the end user experiences with its custom diagram for DX models. In the right part of the working environment it is pos-

sible to choose between different components and relations to be added to the model, in the left part the list of the components present is illustrated, while at the bottom there is a table that contains the data relating to the component selected within the model. The components within the model are represented as blocks of different colors based on their nature, for example green electricity, orange vehicles and so on. In conclusion it can be stated that the use of Sirius technology has brought some advantages in systems management. It has made it possible to maintain the consistency of system-related data used by different tools with different representations. It has made it possible to increase the integration between multiple modeling tools. In the case examined (DX tool) it has made it possible to introduce the graphic editing functions, not yet present in DX.

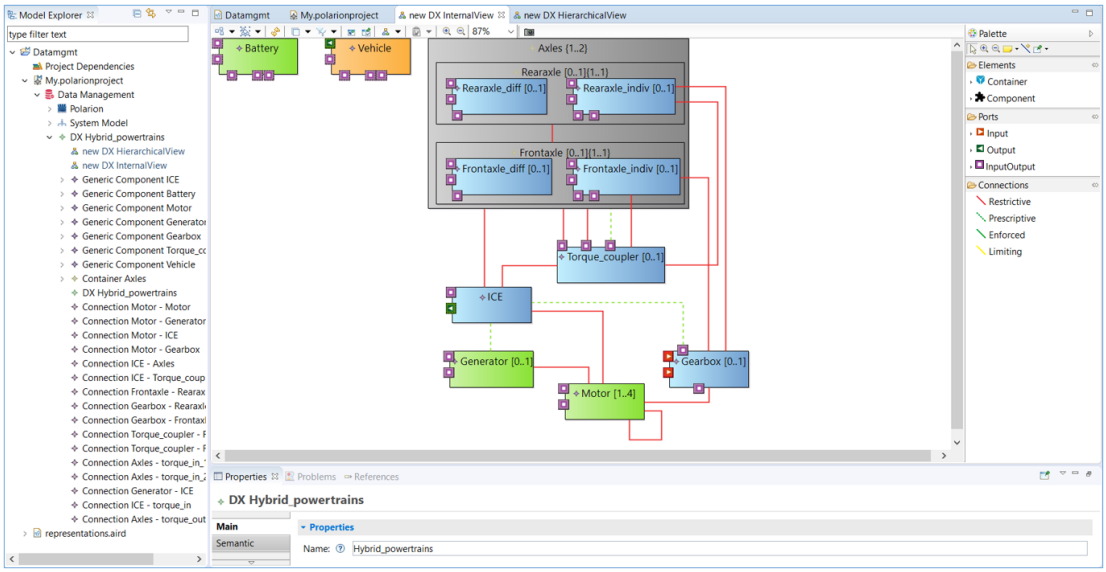

Figure 6: Sirius workbench for DX models.

5 MODEL-BASED SAFETY ANALYSIS

This Section is centered on Model-Based Safety Analysis. In particular, after introducing the basic concepts of system safety analysis, a solution based on model exchange features between different software tools is exploited to ease its execution.

5.1 Safety Analysis

Software plays an increasingly important role in the functioning of the system and in the control of hazards as well as in safety-critical functions. Software failures can cause serious damage to the equipment or properties of the system to which they are exposed and in some cases can even threaten people's lives as is been happened in the Boeing 737 Max accident in 2018/2019. Safety analysis is an important means to recognize these risks and eliminate them, especially in the requirements management phase. It provides and ensures: that all the risks and hazards associated with the functional failures of the system are identified (Garro et al. 2014, Garro and Tundis, 2015).

5.2 Capella

Capella is a system modelling tool provided by Thales. It is an Eclipse application that implements the ARCADIA method (Jean-luc Voirin, Stephane Bonnet, Daniel Exertier, 2015) providing both a Domain Specific Modeling Language (DSML) and a dedicated set of tools. Capella has been specified, designed and developed to provide a high-value engineering environment to the System Engineering and Software

Architecting teams, promoting innovative approaches in engineering practices. It has been successfully implemented in a wide variety of industrial contexts and has proven to have an acceptable maturity for implementation in terms of reliability, robustness, performance, efficiency and adaptability.

5.2.1 Capella System Model

This section illustrates a valid solution to carry out the safety analysis of a model. During the early design phases, a Capella system model is created. In order to make the right design choices, some first safety analyses should be performed. Since Capella does not support this function, the model is exported to other software that supports safety analysis. There are a number of tools on the market that allow this. Figure 7 shows a simple Capella model of a fire detector, in which smoke, sprinklers and alarms are actors of the system. The fire detector is a physical component composed of a sensor and a processor. The sensor detects the presence of smoke while the processor processes the smoke data and sends the execution signals to the sprinkler and the alarm.

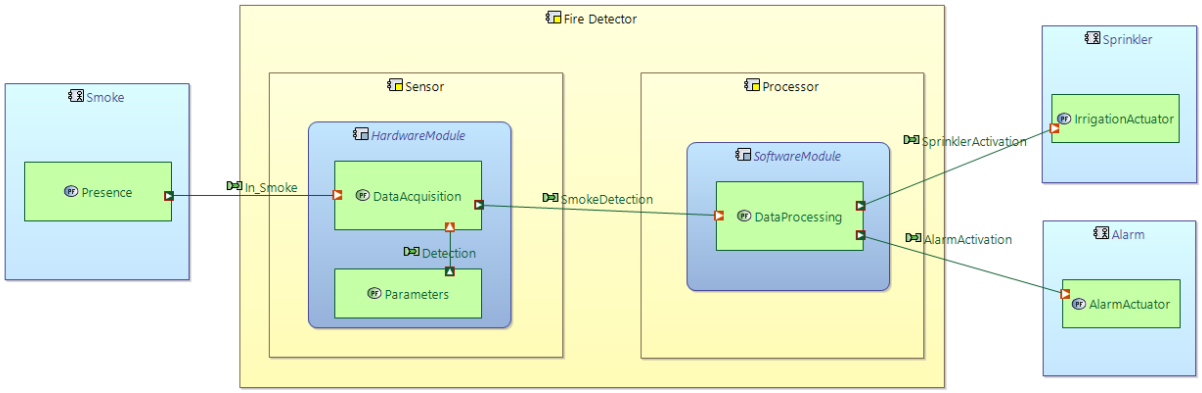

Figure 7: Capella model of a fire detector.

5.3 Perform Safety Analysis of a System

Once the model has been exported to the new tool it is possible to obtain more specific safety information in the form of graphs, tables or fault trees (Garro and Tundis 2012). The general steps to perform a safety analysis are defined below (note that some phases may vary depending on the tool used). The first step is to identify the parts of the system on which we want to analyze the possible failures. For this reason it is necessary to identify within the model a point in which it is relevant to carry out this analysis. For instance, it is meaningless to analyze the causes of failure of a system if a single component is taken into account. The purpose of the safety analysis is to carry out an overall analysis and to see how a failure of one of them affects the system globally. The second step is to set the system failures, for example by configuring the rate with which the failures of each single component taken into account can occur. Here, probabilistic analysis can come into play which allows you to estimate the overall failure rate of the system. The third phase consists in the generation of results, which can occur in various forms: graphs, tables and so on. Within this project, fault trees have been generated. They represent the flow of information between different components in the system. They also make it possible to identify the points in which the system is weakest and on which it is possible to act, perhaps trying to lower the probability of failure of a component or changing the topology of the system. A fault tree generated by the tool is shown in Figure 8. All the components of the architecture are reported in this tree where the different relations between components and the failure that can affect them are highlighted. It also shows how these can lead the structure to a system failure. In this system the following failures can be identified: inside the processor and inside the sensor (green blocks) and how their occurrence contributes to system failure (water com-

mand is not enabled, gray block). Fault trees also show that one or more internal failures alone are capable of causing the system to fail. For example, in the right branch of the tree there are two internal failures, one in the processor and one in the sensor. They converge towards an OR operator which indicates that it is sufficient for one failure to propagate towards the root, which can then lead to the failure of the whole system. In conclusion, it can be stated that the generation of fault trees helps to identify risks during system design. Nevertheless, the tool taken into account for the execution of the safety analysis does not allow the quantitative analysis to be performed. This is why other tools are still needed. These findings are similar to the ones of a similar approach based on the *AADL* language (Peter Feiler and Julien Delange 2016).

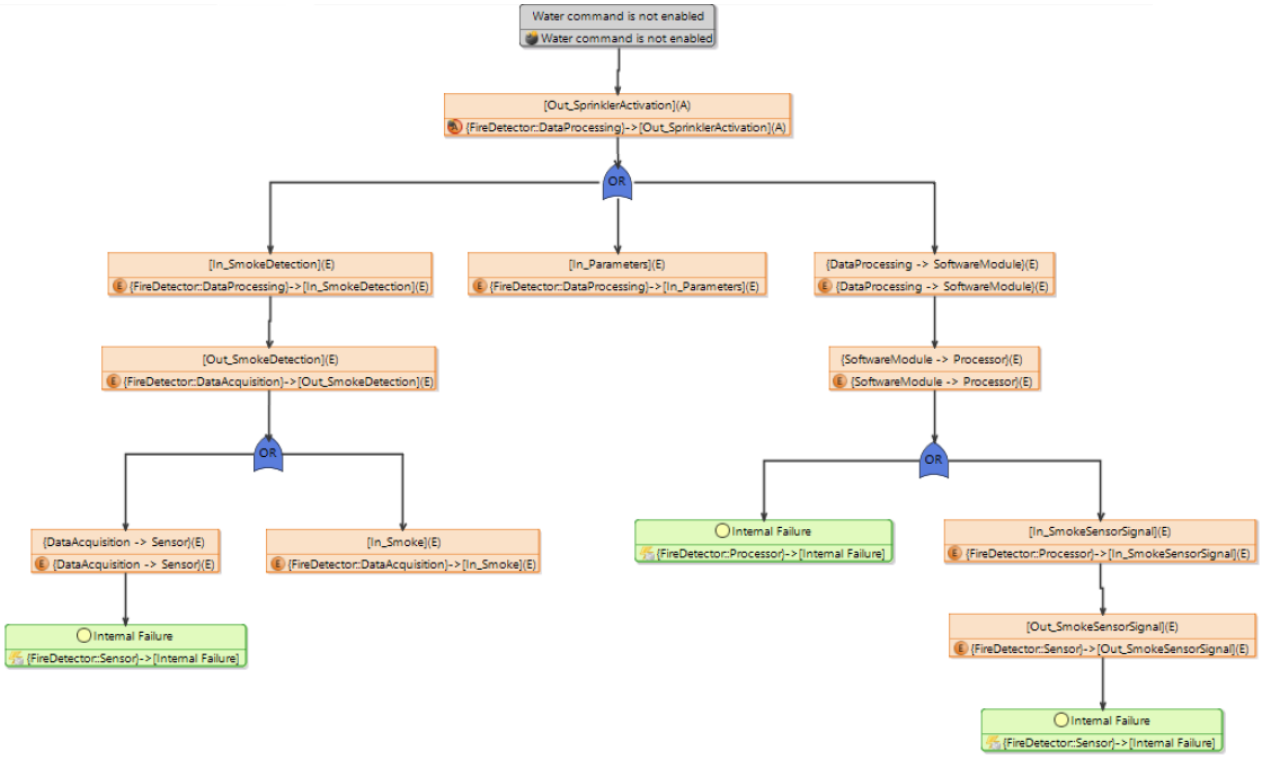

Figure 8: Fault Tree Diagram of fire detector model.

6 CONCLUSIONS

Companies need to make their products more and more attractive on the market, not only by adding new features but also enhancing existing ones. This trend is leading to a more widespread of systems involving tightly coupled hardware and software components (i.e. Cyber-Physical Systems). Requirement Engineering covers a fundamental role in developing CPS, providing a better vision of the problem, facilitating company choices, facilitating the development of the system components and improving the overall quality of the final product. In this context, the paper has proposed some specific extensions to system design tools in compliance with the best practices of MBSE. In particular, in the context of Requirements Management, some improvements have been introduced to the Polarion software tool. The increase in complexity of the models has increased the work flow required by the user to manage the requirements of a project. In this regard, a series of operations have been automated, which can be repeated several times, within a project, introducing a rigorous outline of the requirements life cycle. The constant increase in the complexity of the systems affects also the system representation; in the paper some mechanisms, based on metamodeling, have been introduced to support models integration and simplify the visual manipulation

of system models. In particular, efforts have been invested in the development and implementation of a visual modeling environment in Sirius. Benefits, that derive from this implementation are not limited only to the increased integration between different tools and the related views of the system, but they have also allowed to add the functionalities of interactive editing not present in the original modeling tool, as was for the DX case. Moreover, how to integrate safety analysis in the system design phase by using models exchange features has been also investigated by considering a tool chain based on Capella and Safety Architect. In the context of safety analysis it might be useful to evaluate the effectiveness of the proposed mechanism for the creation and integration of safety assurance cases (Retouniotis et al. 2017), taking full advantage of its potential. Ongoing activities are devoted to modify the metamodel created in Sirius by compacting it and improving the interaction between different system models. This does not appear to be a simple task, but by doing so, it would be possible to verify how modifications made on a system model (e.g. developed in Capella) affect some components in another models (e.g. a DX model) involving common blocks in the metamodel.

REFERENCES

Aiello, F., A. Garro, Y. Lemmens, and S. Dutré. 2017. "Simulation-based verification of system requirements: An integrated solution". In *Proceedings of the 2017 IEEE 14th International Conference on Networking, Sensing and Control, ICNSC 2017*, Institute of Electrical and Electronics Engineers Inc.

Bocciarelli, P., D'Ambrogio, A., Falcone, A., Garro, A., and Giglio, A. 2016. "A model-driven approach to enable the distributed simulation of complex systems". *In Proceedings of the 2016 Complex Systems Design & Management (CS&DM)*, pp. 171-183. Springer.

Burger, E., J. Henss, M Küster, S. Kruse, L. Happe. 2016. "View-based Model-driven Software Development with Model Join". *Software & Systems Modeling* vol. 15, n. 2, pp 473–496, Springer.

de Weck, Olivier L. 2015. "Feasibility of a 5X speedup in system development due to meta design*". In Proceedings of the 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*.

Falcone, A., A. Garro, A. D'Ambrogio, and A. Giglio. 2017. Engineering systems by combining BPMN and HLA-based distributed simulation. *In Proceedings of the 2017 IEEE International Conference on Systems Engineering Symposium (ISSE)*, pp.1-6. Institute of Electrical and Electronics Engineers Inc.

Feiler, P. H., and J. Delange. 2016. "Automated Fault Tree Analysis from AADL Models". *ACM High Integrity Language Technology International Workshop on Model-Based Development and Contract-Based Programming (HILT), held in Pittsburgh, PA, 6-7 October 2016.*

Garro, A., J. Groß, M. Riestenpatt gen. Richter, and A. Tundis. 2014. "Reliability analysis of an Attitude Determination and Control System (ADCS) through the RAMSAS method". *Journal of Computational Science* vol. 5, n. 3, pp. 439-449.

Garro, A., and A. Tundis. 2015. "On the reliability analysis of systems and SoS: The RAMSAS method and related extensions". *IEEE Systems Journal* vol.9, n. 1, pp. 232-241, Institute of Electrical and Electronics Engineers Inc.

Garro, A., and A. Tundis. 2012. *A model-based method for system reliability analysis*. Simulation Series, 44 (4 BOOK), pp. 126-133.

Garro, A., A. Tundis, D. Bouskela, A. Jardin, N. Thuy, M. Otter, L. Buffoni, P. Fritzson, M. Sjölund, W. Schamai, and H. Olsson. 2016. "On formal cyber physical system properties modeling: A new temporal logic language and a Modelica-based solution". In *Proceedings of the 2016 IEEE International Symposium on Systems Engineering, ISSE 2016,* Institute of Electrical and Electronics Engineers Inc.

Meier, J., and A. Winter. 2018. "Model Consistency ensured by Metamodel Integration". *In Proceedings of the 2018 6th International Workshop on the Globalization Of Modeling Languages, GEMOC, CEUR.*

Micouin, P., 2013. *Model Based Systems Engineering: Fundamentals and Methods*. Wiley.

Obeo 2019. "Sirius". https://www.obeodesigner.com/en/product/sirius. Accessed Feb. 4, 2019.

Ratiu, D., M. Zeller, and L. Killian. 2015. "Safety.Lab: Model-based Domain Specific Tooling for Safety Argumentation". *In Proceedings of the 2014 International Conference on Computer Safety, Reliability, and Security, SAFECOMP 2014,* Springer.

Retouniotis, A., Y. Papadopoulos, I. Sorokos, D. Parker, N. Matragkas, and S. Sharvia. 2017. "Model-Connected Safety Cases". *In Proceeding s of the 2017 Model-Based Safety and Assessment: 5th International Symposium, IMBSA 2017, Springer.*

AUTHOR BIOGRAPHIES

ALFREDO GARRO received a PhD in Systems and Computer Engineering from the University of Calabria (Italy), where he is currently an Associate Professor of Computer and Systems Engineering. In 2016, he was Visiting Professor at NASA JSC, working with the Software, Robotics and Simulation (ER) division. His main research interests include multi-agent systems, M&S, systems and software engineering, and reliability engineering. His list of publications contains about 100 papers published in international journals, books, and proceedings of international and national conferences. He is the co-founder (in 2014) and director (from 2018) of the Departmental Research Laboratory "System Modeling And Simulation Hub Lab" (SMASH Lab). He is vice chair of the Space Reference FOM Product Development Group of SISO. He is an IEEE senior member and the Vice-President/Next-president of the "Italian Chapter" of INCOSE. His email address is alfredo.garro@unical.it.

VITTORIO VACCARO is a university student in Computer Engineering at the University of Calabria (Italy). He attended a six-month internship period at Siemens PLM Software (Belgium) in which he worked in context of Model-Based System Engineering aimed to improve some modelling tools under development. His email address is vittorio.vaccaro@unical.it.

STEFAN DUTRE received a PhD in mechanical engineering of the Katholic University of Leuven in 1997. He joined CADSI NV in 1997 as mechanical consulting engineer in multi-body dynamics. In 1999 he joined LMS Engineering services as a project engineer, since 2002 as program and account manager responsible for bigger services projects and accounts. In 2008, he became part of the technology and innovation team of Engineering Services. Since 2009, he belongs to the Aerospace industry solutions team, as Business development manager. In that role, he did business development at different AS&D accounts, positioning Siemens solutions through marketing campaigns and events, interacting with important aerospace accounts. Since 2014, he is responsible for the Siemens Simulation and Testing solutions strategy with a focus on the MBSE solutions for the aviation market. This includes defining the product innovation and R&D for the model based system engineering (MBSE). His email address is stefan.dutre@siemens.com.

JEF STEGEN received his master in Energy Engineering from the Katholic University of Leuven (KUL, Belgium) in 2016. In the year hereafter, he successfully graduated from the postgraduate Innovative Entrepreneurship with a focus on software development and economics. Currently, he is working as a research engineer within Siemens Industry Software on the INES research project which is focused on Model-Based System Engineering of complex mechatronic systems within the aerospace domain. His email address is jef.stegen@siemens.com.