Integrated Model-based Safety Engineering with I-SafE

Pablo Oliveira Antonino, David Santiago Velasco Moncada,
Thomas Kuhn, Daniel Schneider, Mario Trapp; Fraunhofer IESE

Even in the age of computerization, safety engineering is still a matter of textual documents and even pen and paper. One major consequence of this are inconsistent and incomplete specifications of safety-critical systems, which are a core reason of catastrophic failures. To improve the completeness and consistency of safety-critical systems specifications, we present an integrated multi-analysis and multi-viewpoint safety engineering tool called I-SafE, which is a solution that supports general safety analysis as well as the specification and analysis of safety requirements traceability to architecture and failure models.

Safety engineering artifacts still have been defined by means of natural text in documents, spreadsheets, or databases. One major issue caused by that is inconsistency between safety requirements, failure models, and architecture [1]. However, safety requirements often result from a safety analysis of the architecture, and, lately, must be allocated to elements of the architecture[1]. In this regard, the existing inconsistencies and incompleteness lead to intensive efforts being required to update the artifacts impacted by the changes, and, consequently, significantly decrease the efficiency of the safety assurance architecture [2].

To contribute to overcoming this challenge, in this paper we introduce I-SafE: Integrated Safety Engineering, which is an Enterprise Architect[1] based tool that supports the specification of traceable safety requirements, failure models, and architecture models, contributing, then, to ensure safety-by-construction, as safety is considered early in the process of the system design.

Running Example

The I-SafE features described in this paper will be illustrated using a simplified version of a fictitious electric motor drive (E-Drive) system, which is depicted in the Fig. 1.

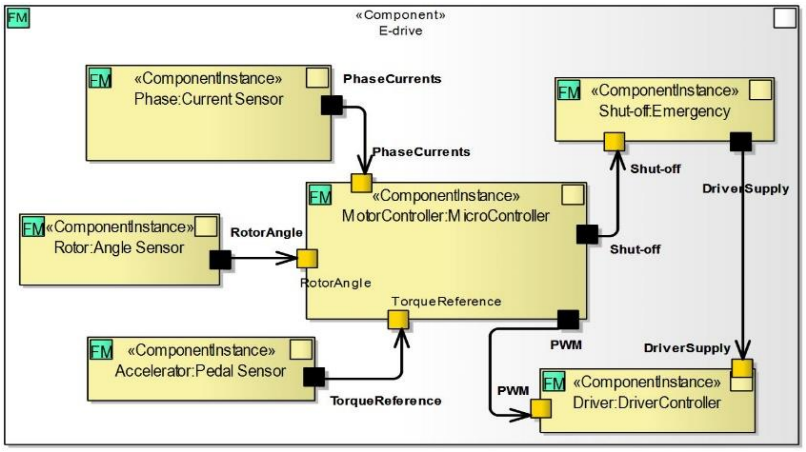

Fig. 1. Logical model of the E-Drive system.

Specifying Architecture Models with the Embedded Modeling Profile

[1] http://www.sparxsystems.com

I-SafE supports the specification of Functional, Logical, and Technical aspects of the architecture, which are based on the Embedded Modelling Profile [3]. Examples of the architecture modelling toolbox provided by I-SafE is depicted in Fig. 2.

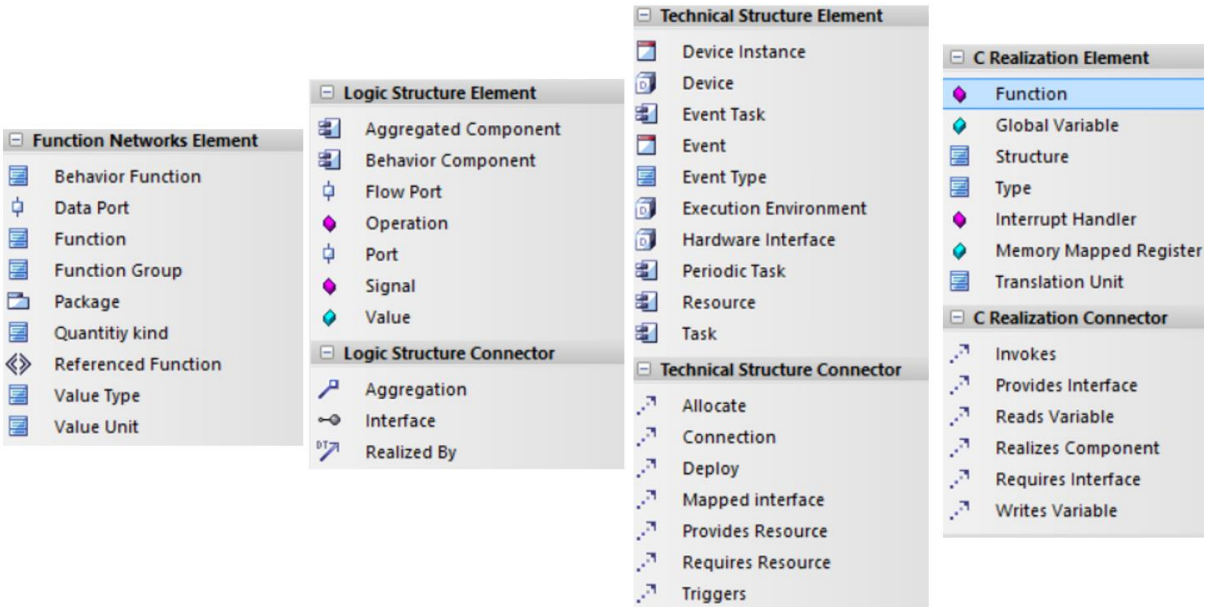

Fig. 2. Architecture Modellig toolbox provided by I-SafE.

Creation of Failure Models with I-SafE

I-SafE supports creating failure models of the types Component Fault Trees - CFTs, Failure Modes and Effects Analysis - FMEAs, and Markov Chains that are associated to architecture elements. Due to space constraints, only the CFT and FMEA support are described in this paper.

Component Fault Trees (CFT) extend standard fault trees with the concept of modularity in component based specifications. For example, Fig. 3 depicts a CFT created with I-SafE for the emergency shut-off component of the E-Drive system illustrated in Fig. 1.

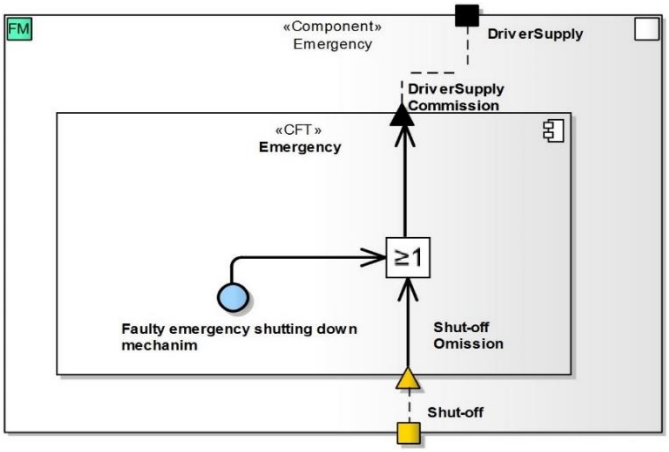

Fig. 3. CFT for the E-Drive's emergency shut-off.

The I-SafE's support regarding the specification of FMEA is based on interface-focused IF-FMEA [4] for each system component. For instance, Fig. 4 depicts an FMEA for the E-Drive's Pedal Sensor shown in Fig. 1.

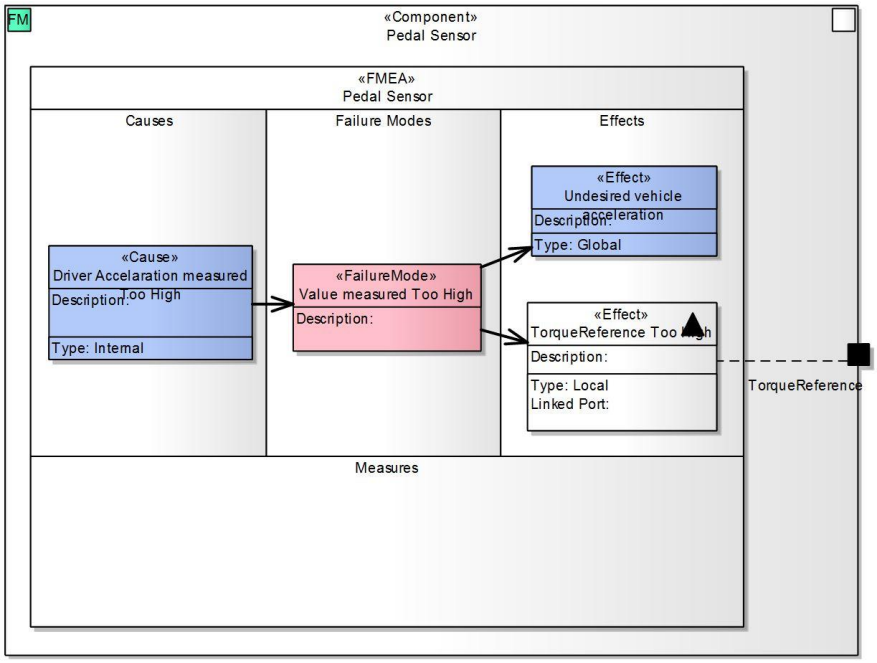

Fig. 4. FMEA for the E-Drive's Pedal Sensor.

TRACING SAFETY REQUIREMENTS SPECIFIED WITH NATURAL LANGUAGE TO FAILURE MODELS AND TO THE ARCHITECTURE

In order to conveniently support the creation of trace links, I-SafE provides an autocomplete mechanism that suggests elements that should be referenced in the safety requirement being specified. These suggestions are made when the text being written has similarities with the names of elements present in the failure models or architecture models. For instance, as shown in Fig. 5, as soon as the user starts to type the text fragment *"The M"*, the suggestions of the architecture component "MicroController" (cf. Fig. 1), along with other elements that have similarities with this string, such as the MicroController CFT (cf. Fig. 2), are shown in the suggestion list.

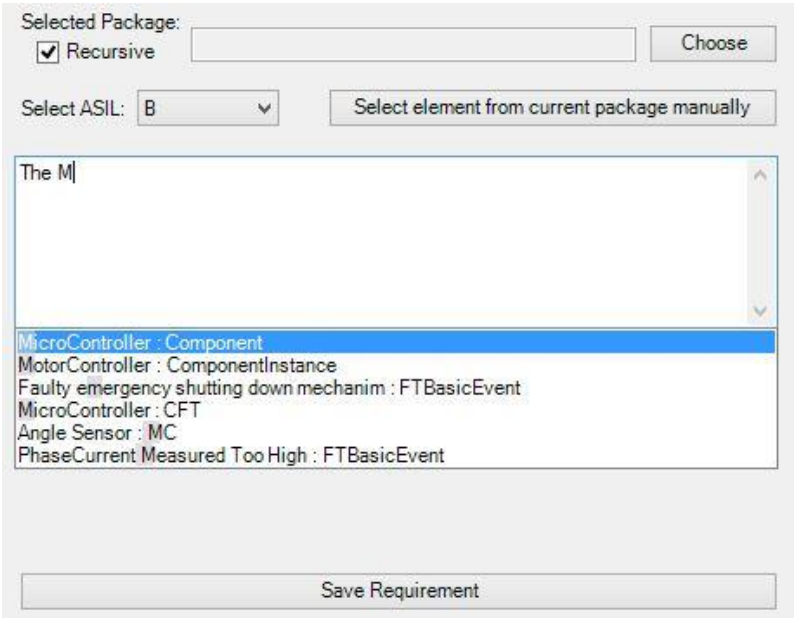

Fig. 5. I-SafE autocomplete mechanism.

I-SAFE VISUAL TRACE

I-SafE provides a visual trace mechanism that allows engineers to visualize all elements related to each safety requirement specification. It allows visualizing (i) architecture elements and elements of failure models that are explicitly referenced in textual safety requirements specifications; (ii) architecture elements that are not explicitly mentioned in safety requirements specifications, but that are related (over a series of indirections) to those that are explicitly referenced.; and (iii) Other specifications related to safety requirements being analyzed, such as Conditional Safety Certificate [5] of a given component related to a safety requirement.

In the example shown in Fig. 6, the safety requirement has an explicit trace to the MotorController element (cf. Fig. 1). After activating the Visual Trace mode of I-SafE, whenever the user clicks on the safety requirement element the diagram shown in Fig. 1 opens and only the referenced element MicroController is highlighted.

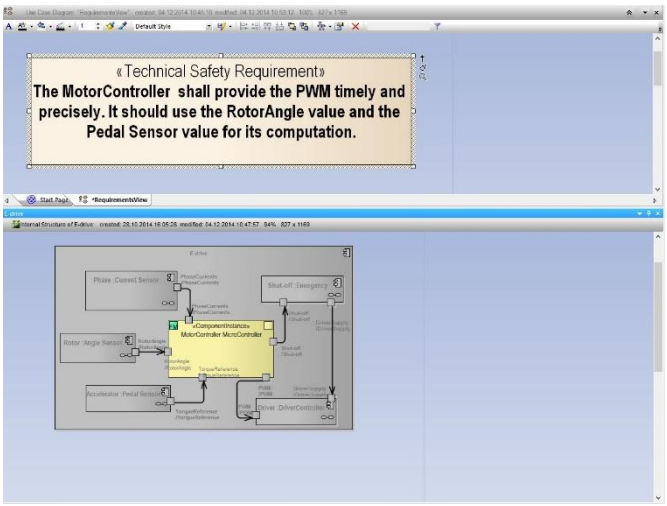

Fig. 6. I-SafE mechanism for visualizing artifacts related to a safety requirement.

AUTOMATED COMPLETENESS AND CONSISTENCY CHECKS

I-SafE provides support for executing completeness and consistency checks between safety requirements and architecture design, aiming at detecting and alerting engineers to existing incompleteness and inconsistencies.

With respect to completeness, I-SafE checks whether (i) every safety requirement describes mitigation strategies for failures that are described in at least one failure propagation model; (ii) every failure propagation model describes the failures of at least one safety-critical architecture element; and (iii) every safety requirement describes failure mitigations referencing at least one safety-critical architecture element. The completeness checks are displayed to the user as shown in Fig. 7, where a list of safety requirements is displayed along with their types and the completeness violation.

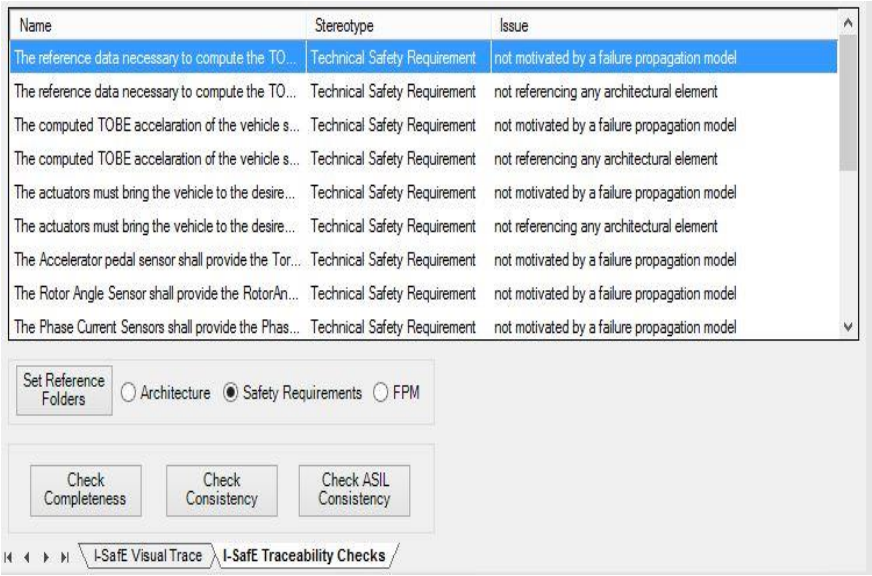

Fig. 7. I-SafE completeness checks outputs.

With respect to consistency, one of the main checks offered by I-.SafE is on Safety Integrity Level – SIL inconsistencies, which are caused when safety requirements and the safety-critical architecture elements that address them have incompatible safety integrity levels. For instance, Fig. 8 shows a list of architecture elements that have ASIL incompatibility. The basis for these and for all the other completeness and consistency checks implemented in I-SafE is described in [6]. The other consistency checks supported by I-SafE are not described due to space limitations.

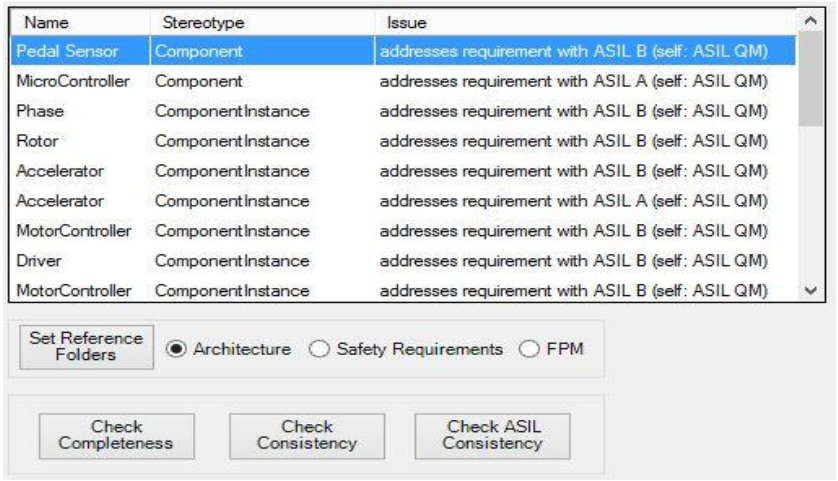

Fig. 8. ASIL violations detected by I-SafE.

Conclusion

I-SafE provides a range of features that can rarely be found in other tools. Among the features presented in this paper, we deem the aspects of integration and traceability particularly important. Integration between different (types of) modular analysis models in the context of a larger system and traceability between safety requirements and related artifacts along the safety engineering chains are features that are bound to ease the daily work of software and safety engineers.

References

[1] P. O. Antonino, M. Trapp, P. Barbosa, E. C. Gurjäo, J. Rosário: The Safety Requirements Decomposition Pattern. SAFECOMP 2015: 269-282

[2] J. Hatcliff et al., 2014. Certifiably safe software-dependent systems: challenges and directions. Hyderabad, India, s.n., pp. 182-200.

[3] T. Kuhn and P. O. Antonino. Model Driven Development of Embedded Systems. Proceedings of the Embedded Software Engineering Kongress 2014. Pages 47–53.

[4] Y. Papadopoulos, J. McDermid, R. Sasse, and G. Heiner, 2001. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering & System Safety, 71(3), pp. 229-247.

[5] D. Schneider and M. Trapp, 2013. Conditional Safety Certification of Open Adaptive Systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 8(2), pp. 1-20.

[6] P. O. Antonino and M. Trapp. Automatic Detection of Incomplete and Inconsistent Safety Requirements. SAE 2015 World Congress and Exhibition, Detroit, Michigan USA, 2015.

Authors

Pablo Oliveira Antonino has been working as a computer scientist at the Fraunhofer IESE since 2009. His work focuses on the construction and analysis of architectures of dependable systems.

Dr. Thomas Kuhn received his PhD from the University of Kaiserslautern, Germany, in 2009. Since 2008, he has been working for the Fraunhofer IESE. Currently, he is head of the Embedded Software Development department and is responsible for architecture analysis approaches and virtual prototypes.

David Santiago Velasco Moncada obtained his M.Sc. degree in computer science from the University of Kaiserslautern in 2011. Since then he has been working in the Embedded Systems Quality Assurance department of Fraunhofer IESE. His research interests include safety and reliability of safety critical systems and software engineering.

Daniel Schneider is the Head of the Embedded Systems Quality Assurance department at Fraunhofer IESE. His research interests include runtime certification of safety critical systems. He has a doctorate in computer science from the University of Kaiserslautern.

Mario Trapp is a division manager at the Fraunhofer IESE. His research interests include model-driven development of high-integrity embedded systems and safety-engineering techniques. He has a doctorate in computer science from the University of Kaiserslautern.

Contact
P. O. Antonino / D. S. V. Moncada
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern

Email: Pablo.Antonino@iese.fraunhofer.de
Santiago.Velasco@iese.fraunhofer.de