INTEGRATING FMEA IN A MODEL-DRIVEN METHODOLOGY

Fabio Scippacercola[1,2], Roberto Pietrantuono[1,2], Stefano Russo[1,2], Alexandre Esper[3], and Nuno Silva[3]

[1]*Consorzio Interuniversitario Nazionale per l'Informatica, Via Cinthia, 80126 Napoli, Italy, Tel.: +39 081 67 6770*
[2]*DIETI, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy, Tel.: +39 081 768 3820,*
Email: {fabio.scippacercola, roberto.pietrantuono, stefano.russo}@unina.it
[3]*Critical Software, SA, Parque Industrial de Taveiro, lote 49, 3045-504 Coimbra, Portugal, Tel.: +351 932 574 030,*
Email: {aresper, nsilva}@criticalsoftware.com

ABSTRACT

Failure Mode and Effects Analysis (FMEA) is a well known technique for evaluating the effects of potential failures of components of a system. FMEA demands for engineering methods and tools able to support the time-consuming tasks of the analyst. We propose to make FMEA part of the design of a critical system, by integration into a model-driven methodology. We show how to conduct the analysis of failure modes, propagation and effects from SysML design models, by means of custom diagrams, which we name *FMEA Diagrams*. They offer an additional view of the system, tailored to FMEA goals. The enriched model can then be exploited to automatically generate FMEA *worksheet* and to conduct qualitative and quantitative analyses. We present a case study from a real-world project.

1. INTRODUCTION

Failure Mode and Effects Analysis (FMEA) is a technique to systematically identify the potential failures of parts of a system – be they subsystems, assemblies, components, or functions[1] – and to analyze their effects on the system. Since FMEA helps to understand the effects of potential failures of the system, it is widely adopted in critical systems engineering.

The FMEA produces a *worksheet*, which is the main artifact of the analysis, and influences the amount and type of information FMEA considers. Indeed, different worksheet types may be adopted, depending on context, specific goals, customer requirement, and preferences of the workgroup, but they report as minimum for each failure mode of a component the component-level effects resulting from failure, the system-level effects, and the causal factors. The worksheet is useful to identify parts of the systems that need to be re-engineered to mitigate, or avoid, certain system failures.

In our previous work [1], we proposed a SysML-based approach to support FMEA by enabling formal knowledge representation – thus automated reasoning. SysML [2] is a design language standardized by the International Council of Systems Engineering and the Object Management Group, increasingly used in critical domains, for embedded [3] as well as for large-scale systems [4]. The approach (Fig. 1) starts from a system model in SysML. Design artifacts are augmented with FMEA-oriented information using annotations. The additional information allows transformation to a Prolog knowledge base. Queries to the Prolog engine provide FMEA results.

In this paper we focus on the first step of this approach (FMEA-oriented modeling in Fig. 1). We show in detail how to refine SysML design models with FMEA-oriented information, while reasoning on faults, failure modes, use cases and requirements by means of models. To this end, we introduce a novel custom SysML diagram, namely the *FMEA Diagram*, that aims at being easy and practical to analyze and document the propagation and effects of failures.

The model-driven FMEA starts from a system design consisting of: requirements specification (modeled with

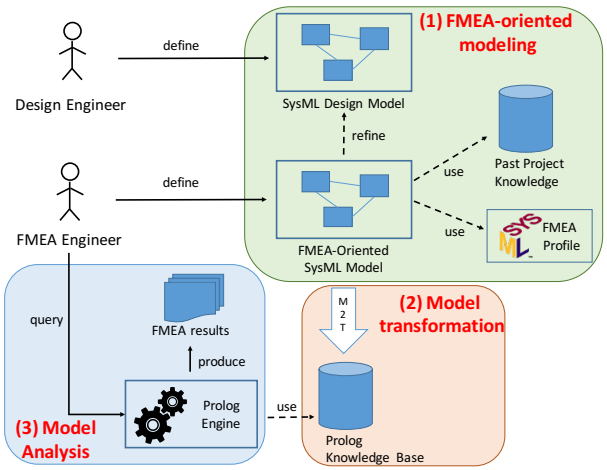

Figure 1. Overview of the model-driven FMEA methodology.

[1]Here we will generically refer to parts as *components*

SysML Requirement and Use Case diagrams); components definition (*Block Definition Diagrams*, BDDs, and *Internal Block Diagrams*, IBDs); requirements allocation to components; system operational behavior (Activity diagrams). *FMEA Diagrams* are then built, to provide a view from the viewpoint of a FMEA analyst on the relations among faults, components, use cases and failure modes.

The paper is structured as follows. Section 2 discusses the motivations and the related work. Section 3 describes how model-driven FMEA is conducted. Section 4 presents a case study based on a real-world embedded system. Section 5 concludes the paper.

2. MOTIVATIONS AND RELATED WORK

FMEA is a time-consuming technique, performed by skilled professionals. It basically consists in identifying potential *failure modes*, analyzing possible *effects* and their severity, and suggesting mitigation means. Standards recommend how it should be conducted in specific domains [5], [6]. Spreadsheets are traditionally used, but many ad hoc tools are nowadays available: examples are Byteworx [7], FMEA-Pro [8] and Raytheon's ASENT [9]. They provide guidance, help to reduce mistakes, and offer useful features, e.g. for consistency checking, traceability and documentation.

Despite this, most FMEA activities are carried out in a manual way. One reason is that the analysis is often performed late, after main forward engineering stages (requirements analysis, design, development), and with little integration with them. FMEA analysts usually start reconstructing knowledge from design documentation. The needed information may be spread in many documents, in different formats, that depend on multiple teams. Analyzing these sources is a laborious and error-prone task. We aim to bridge the gap between the designer and the FMEA analyst, getting their views closer to each other. This is meant to be achieved by using SysML and integration into a model-driven process.

Starting FMEA from SysML artifacts and integrating the analysis in a model-driven process provides as benefits:

- The FMEA analyst can *reason in the same conceptual framework of the design models* to analyze threats to non-functional requirements.
- A relevant *initial effort of FMEA is saved.* FMEA is typically performed by a RAMS[2] team, who receives the input documentation (requirements, design) and needs to extract the system components and their functionalities and interactions, normally through the creation of diagrams. A great advantage is that the RAMS team may receive all this ready information as an input.
- The support for automation provided by model-driven techniques and tools can significantly reduce the effort and cost of FMEA; the worksheet and other *artifacts may be derived through model transformations*. Moreover, the fragmentation of how FMEA is performed by different teams is reduced.
- *Early exploitation of design artifacts for FMEA* becomes possible. In many industrial settings, development and testing follow a *V-model* engineering process according to the well-established MIL-STD-498 standard [10]. One major goal of the V-model is early verification and validation. The envisaged model-driven FMEA favors better consideration of RAMS requirements in the first stages, by allowing early feedback from the FMEA team. Such process-level improvements reduce the risk of major redesign due to threats to RAMS requirements.

These advantages are clearly inter-related. The relative importance depends on the context. For instance, when FMEA is outsourced to independent companies, early FMEA feedback may be of greater interest for the outsourcing organization than easing the task of the outsourced team.

The idea of deriving FMEA artifacts from SysML models is not new. Hecht et al. showed how to automatically generate the worksheet from SyML BDDs, IBDs and STDs (State Transition Diagrams) [11], through an intermediate transformation into an AltaRica state machine model. Before them, David et al. derived FMEA from IBDs, SDs (Sequence Diagrams) and the AltaRica language [12], while Xiang et al. from IBDs and the algebraic specification language Maude [13]. Less emphasis has been put at the process level, namely on how FMEA can be integrated in a model-driven methodology.

Our approach differs from past proposals in that SysML models are transformed into a Prolog knowledge base. However, *here we are concerned specifically with the problem of supporting the FMEA engineer in the early phases*, in reasoning on design models at components level to analyze failure modes, propagation and effects; that is, we focus on the FMEA-oriented modeling step (1) of Fig. 1. To this aim, we propose a structured approach that uses also Requirements and Use Case diagrams and is driven by *FMEA Diagrams*.

A FMEA diagram is a custom SysML diagram which is split in five logical sections (Fig. 2). The diagram places the *Component Under Analysis* and its *failure modes* in the middle, using the graphical notation of *use cases*. This idea has some similarities with *misuse cases* [14], which have been proposed as a means to analyze security threats and applied for safety analysis [15][16]. The difference is that we are modeling directly the failure modes of a component in relation to the use cases it is involved in, rather than - in an indirect way - the sequence of actions that the component can perform, whose failure modes can cause harm to some stakeholder. As for use cases, their exploitation for guiding FMEA has been initially proposed by Allenby et *al.* [17]. We leverage the possibility to derive some failure modes by translating the use cases for the CUA considering common deviations from the expected service (e.g. in content and/or timing).

[2]Acronym for Reliability, Availability, Maintainability and Safety.

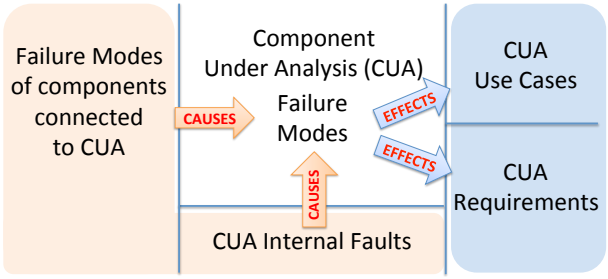

Figure 2. Logical layout of a FMEA diagram.

3. MODEL-DRIVEN FMEA

The first step of the methodology consists in modeling the system with SysML. Functional requirements are modeled with Use Case and Requirement Diagrams. Relations among the requirements, such as dependencies and containments, are identified at this stage. The system architecture is modeled with BDDs and IBDs, and the *requirements are allocated to components*. System operational aspects are specified with behavioral diagrams, e.g., Activity Diagrams.

FMEA-specific activities start from this design model. For each component the analyst defines one *Use Case Diagram* modeling its functionalities. Then, the analyst uses the *FMEA Diagram* to reason on functional and structural dependencies of the *Component Under Analysis* (CUA), which is the component object of FMEA analysis.

The failure modes involving the CUA are due to the activation of internal faults, or to the propagation of failures from adjacent components. On the left side, the FMEA Diagram (Fig. 2, see Fig. 9 for an example) shows the components connected to the CUA, and their failure modes. The lower side of the diagram specifies the internal faults of the CUA, that are modeled as *behaviors*. The activation conditions are modeled associating the failures and faults – which appear, respectively, on the left and lower side of the diagram – with the CUA's failure modes. Additional information can be added to the diagram, such as the logical condition that must hold for the failure occurrence, and the probability and severity of the events. The right side shows the effects of a failure, namely, which use cases are affected (upper part) and which requirements allocated to the CUA are violated (lower part). By looking at FMEA Diagram from left to right, the analyst easily spots the chains *fault, error, failure* for the CUA.

To build a FMEA diagram, the analyst:

1. Places the CUA at the centre of the diagram and specifies its *failure modes* based on the use cases on the right side. In case of too many failures modes, these can be grouped into subsets and assigned to *port*s.
2. Defines the *internal faults* of the CUA, as one or more behaviors, adding them in the lower part of the diagram.
3. Models the *activation conditions* of the failure modes, linking, by associations, external failures and internal faults with the failure modes, and specifying logical conditions, probability and severity.
4. Models the *effects* of a failure, linking, by relations, the failure modes with use cases and requirements that have been assigned to the CUA, as listed in the right side of the diagram.

The analysis proceeds bottom-up: each FMEA Diagram offers a synoptical view of functional and structural elements useful for FMEA from the viewpoint of a single component. Once the analyst has augmented models with FMEA information, the subsequent transformation (2) and analysis (3) steps of the methodology in Fig. 1 can take place.

4. CASE STUDY

The case study is provided by the EMC^2 project[3]: it is an embedded system for mixed criticality car applications, that enables software with distinct safety-critical requirements to run concurrently on a multi-core platform.

The system requirements are specified through Use Cases (Fig. 3). An info-entertainment software runs concurrently with a safety-critical in-car emergency call (*eCall*) application. This activates assistance in case of accidents, automatically sending relevant information (e.g., car position) to rescue services.

The requirements and their dependencies are modeled with Requirement Diagrams (Fig. 4). For instance, requirement REQ-0305 on the execution of real-time tasks

[3]Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environment (www.artemis-emc2.eu)

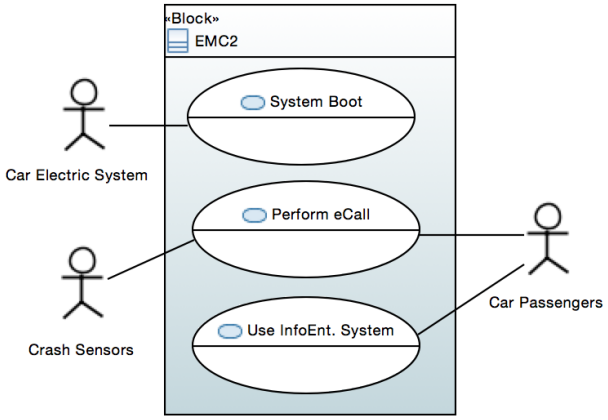

Figure 3. Excerpt of Use Case Diagram of EMC^2 *prototype.*

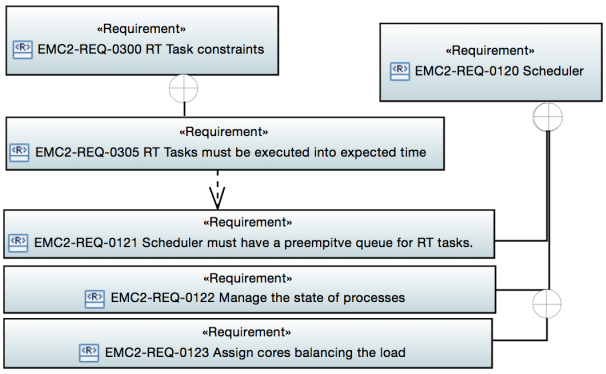

Figure 4. Excerpt of SysML Requirements Diagram of EMC^2.

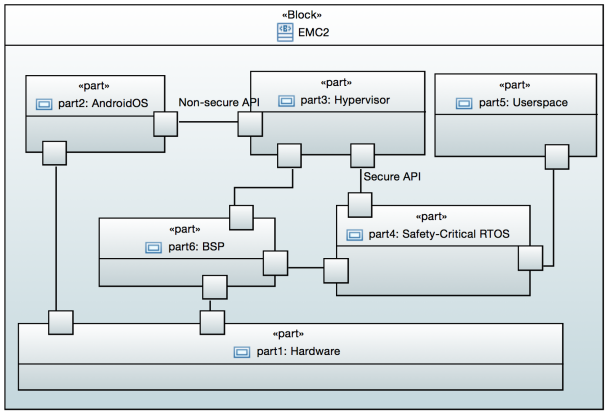

Figure 5. SysML Internal Block Diagram of the EMC^2 *prototype.*

depends on REQ-0121, which requires a preemptive scheduler.

The system architecture is modeled with the IBD of Fig. 5. It consists of a real-time operating system (RTOS), that manages safety-critical tasks and resources running concurrently on distinct CPU cores, with a commercial off-the-shelf operating system (Android OS), contained within a virtualized environment managed by the Hypervisor. The less critical tasks run in user space atop the RTOS. Finally, the Hypervisor and the RTOS interact with a Board Support Package (BSP) abstracting hardware-specific services. The RTOS internal structure (Fig. 6) comprises a Scheduler, and other components managing Clock, Devices, Resources, and System Calls. Requirements are allocated to components at this stage.

The FMEA starts with the analysis on the functionalities of each component. The analyst models component functionalities by means of Use Cases (Fig. 7), based on behavioral diagrams, such as the Activity Diagram of Fig. 8. Note that here Use Cases are abstractions of component functionalities just in the FMEA perspective.

FMEA Diagrams come into play for conducting the analysis of failure modes, propagation and effects in a sys-

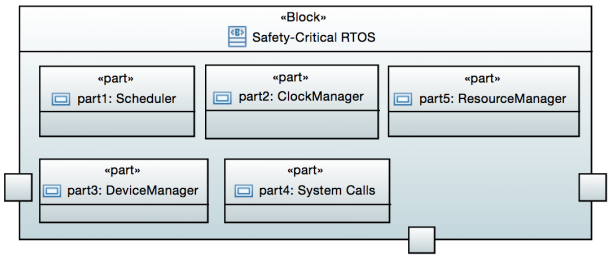

Figure 6. Internal Block Diagram of the Safety-Critical RTOS Component.

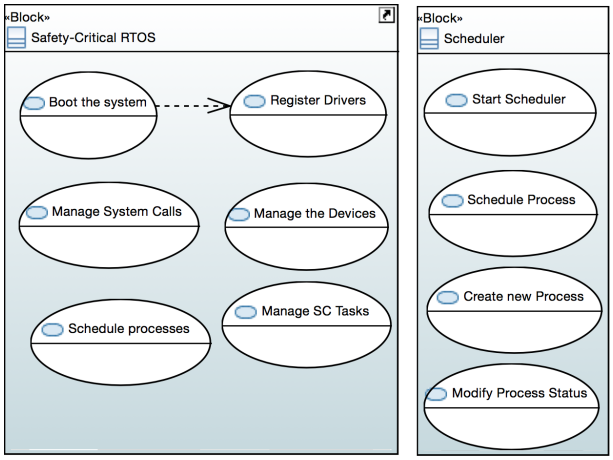

Figure 7. Use Cases for the SC-RTOS and Scheduler components.

tematic way. The engineer proceeds in two rounds. In the first one, (s)he creates one FMEA Diagram for each component; based to the CUA functionalities (top-right side), the analyst defines the CUA failure modes. In the second round, once all components' failure modes have been specified, the engineer re-examines all diagrams: (s)he connects the external failure modes and internal faults with the CUA's failure modes, and these with their effects, linking failure modes to requirements (Fig. 9).

The complete final model is analyzed to derive local and end effects of each failure mode. For instance, let us consider the ClockManager's failure mode *Timer Callbacks are not handled properly* (Fig 9): it propagates to the Scheduler's failure mode *the priority of RT Tasks is not respected*, that has as local effect the violation of requirement *REQ-0121 Scheduler must have a preemptive queue for RT Tasks*. By the dependency in Fig. 4, we conclude the system-level effect of violation of *REQ-0300 RT Task constraints*.

5. CONCLUSIONS

In this paper we have described a structured technique for conducting the initial FMEA steps, when the analyst reasons on design models to identify failure modes, propa-

gation and effects. The technique uses FMEA Diagrams, a custom type of diagrams we have introduced for systematic reasoning at component level, based on SysML system design models.

We envisage the full integration of FMEA into model-driven engineering, in particular in the process based on the V-model, that we have defined and experimented in industrial collaboration [3][4], for early verification of RAMS requirements, and to support automation of FMEA analysis and documentation [1].

ACKNOWLEDGEMENT

This work has been supported by EU with the project CECRIS ("CErtification of CRItical Systems", www.cecris-project.eu) Grant Agreement (GA) n. 324334 of the IAPP programme, and project EMC2 ("EMBEDDED Multi-Core Systems for Mixed Criticality Application in Dynamic and Changeable Real-time Environments", www.artemis-emc2.eu) of ARTEMIS programme, GA n. 611420.

REFERENCES

1. Scippacercola, F., Pietrantuono, R., Russo, S. & Silva, N. (2015). SysML-based and Prolog-supported FMEA. In *Proc. 5rd WoSoCER 'IEEE International Workshop on Software Certification'*, IEEE, pp174–181.

2. OMG (2008). Systems Modeling Language (SysML). www.omg.org/docs/formal/08-11-02.pdf. Version 1.1.

3. Scippacercola, F., Pietrantuono, R., Russo, S. & Zentai, A. (2015). Model-Driven Engineering of a Railway Interlocking System. In *Proc. 3rd MODELSWARD 'International Conference on Model-Driven Engineering and Software Development'*, SCITEPRESS, pp509–519.

4. Carrozza, G., Faella, M., Fucci, F., Pietrantuono, R. & Russo, S. (2013). Engineering Air Traffic Control Systems with a Model-Driven Approach. *IEEE Software*, **30**(3), 42–48.

5. U.S. DoD. Procedures For Performing A Failure Mode, Effects And Criticality Analysis. http://src.alionscience.com/pdf/MIL-STD-1629RevA.pdf.

6. SAE International (2012). Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications. http://standards.sae.org/arp5580/.

7. Byteworx. FMEA Software. www.byteworx.com.

8. IHS. FMEA-pro. www.ihs.com/products/fmea-pro.html.

9. Raytheon Technical Services Company. ASENT FMEA Software. www.raytheoneagle.com/asent/fmea.htm.

10. U.S. DoD (1996). MIL-STD-498 Overview and Tailoring Guidebook.

11. Hecht, M., Dimpfl, E. & Pinchak, J. (2014). Automated Generation of Failure Modes and Effects Analysis from SysML Models. In *Proc. 25th ISSREW 'International Symposium on Software Reliability Engineering Workshops'*, IEEE, pp62–65.

12. David, P., Idasiak, V. & Kratz, F. (2010). Reliability study of complex physical systems using SysML. *Reliability Engineering & System Safety*, **95**(4), 431–450.

13. Xiang, J., Yanoo, K., Maeno, Y. & Tadano, K. (2011). Automatic synthesis of static fault trees from system models. In *Proc. 5th SSIRI 'International Conference on Secure Software Integration and Reliability Improvement'*, IEEE, pp127–136.

14. Sindre, G. & Opdahl, A. L. (2004). Eliciting security requirements with misuse cases. *Requirements Engineering*, **10**(1), 34–44.

15. Sindre, G. (2007). A look at misuse cases for safety concerns. In *Situational Method Engineering: Fundamentals and Experiences*, Springer, pp252–266.

16. Stålhane, T. & Sindre, G. (2007). A comparison of two approaches to safety analysis based on use cases. In *Proc. ER 'Conceptual Modeling'* (Eds. Parent, C., Schewe, K.-D., Storey, V., & Thalheim, B.), Springer, LNCS-4801, pp423–437.

17. Allenby, K. & Kelly, T. (2001). Deriving safety requirements using scenarios. In *Proc. 5th RE 'IEEE International Symposium on Requirements Engineering'*, IEEE, pp228–235.

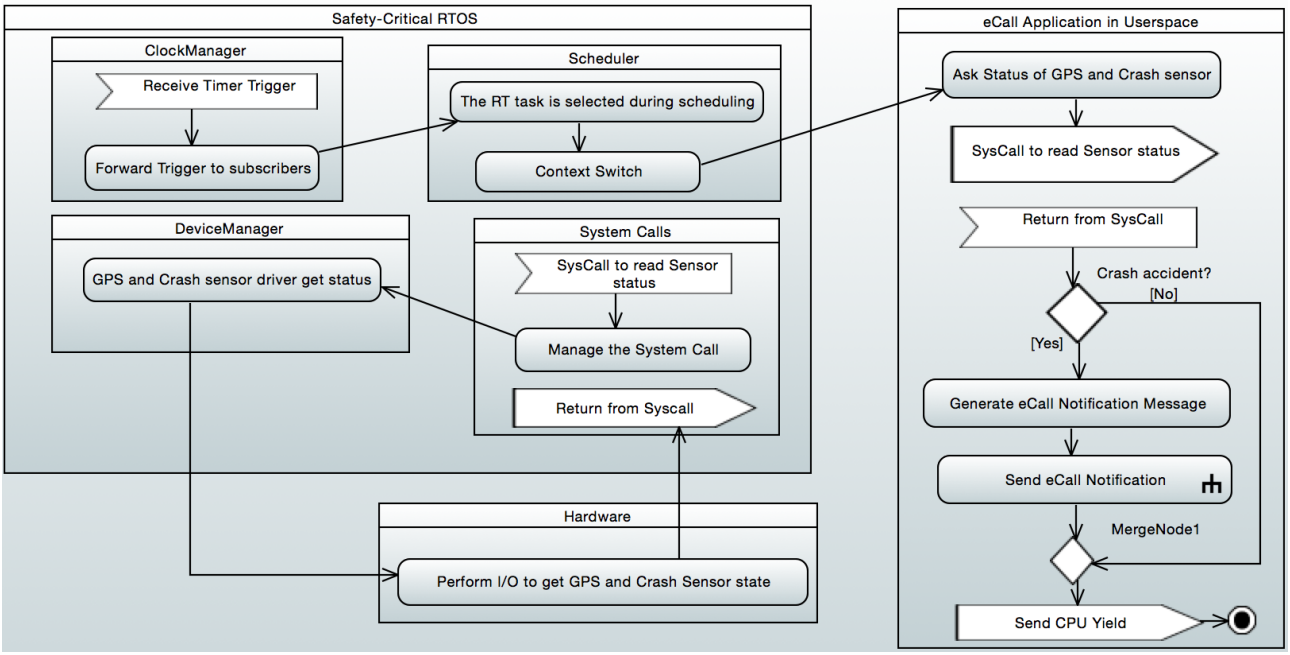

Figure 8. Activity Diagram of the use case Perform eCall.

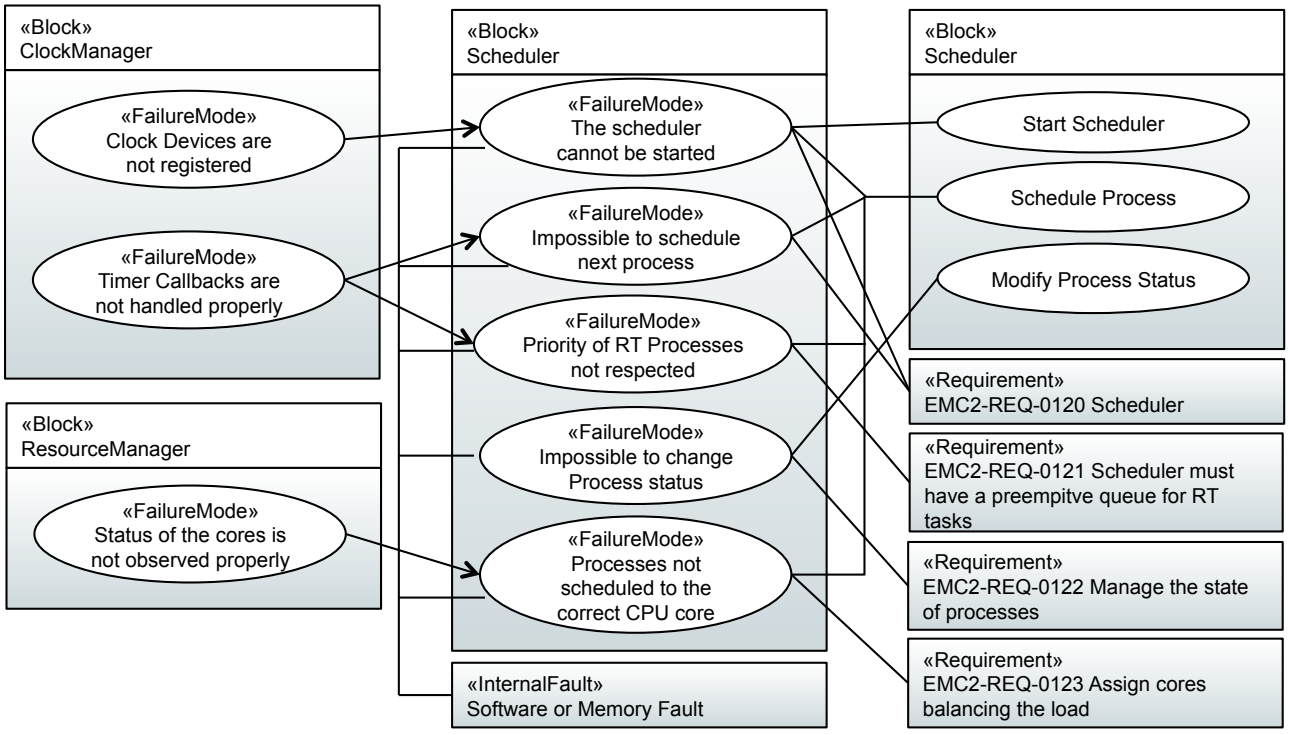

Figure 9. FMEA Diagram for the EMC^2 Scheduler *component.*