Application of Safety Analyses in Model Driven Development

Javier Fernández Briones, Miguel Ángel de Miguel, J. P. Silva, Alejandro Alonso

Department of Telematics Engineering, Technical University of Madrid (UPM),
Ciudad Universitaria s/n, 28040 Madrid, Spain.
e-mail: jfbriones@dit.upm.es

Abstract. Some high integrity software systems require the rigorous validation of safety properties. Assessing whether software architectures are able to meet these requirements is of great interest: to avoid the risk that the implementation does not fulfill requirements due to a bad design, and, to reduce the development cost of safety critical parts of the system. Safety analyses like FMECA and FTA are two methods used during preliminary safety assessments. We have implemented tools to automatically generate safety analyses from the models of the architecture: a UML profile for safety, modeling languages to express safety analyses, and a model transformation chain. Safety analysts can use these tools to annotate the models, analyze the architecture, and recommend system engineers mitigation means to apply for improving the architecture.

1. Introduction

Even though software already inundates many industries and human activities, there is still interest in constructing, using complex software, more pervasive systems with safety requirements. Some features have been requested to the development of such systems, to highlight: a better reuse, new solutions for managing complexity, and enhanced ways for undertaking safety concerns in complex software systems.

Component-based development and model-driven architecture (MDA) are two relatively new methods for software development which tackle the problem of reusability and complexity. With the former, software architectures are assembled with components potentially from a variety of sources, written in different programming languages and running on several platforms. The latter takes software models as leitmotiv in a development process based on continuous model transformations; converting representations of the structure and functionality of an application into implementation models dependent of a specific platform.

Software analysis aims to provide information about the behavior of the software during all phases of the development, whilst formal methods are of more application during the implementation phase. Software safety analyses are an important requisite in the stage of certification, but also, its use at the early phases of the development claims to be a means for reducing costs, keeping the system assessed as safe. FMECA (Failure Mode Effects and Criticality Analysis [8], [10]) and FTA (Fault Tree Analysis [9], [10]) are two ways to analyze the safety and reliability of a system.

Safety analysis models are classically created within a safety tool or just using an office suite application, but they clearly depend on system elements which are usually expressed in a development modeling language. The isolation of both approaches, analysis and design, creates inconsistencies and duplicates efforts. We convincingly consider that some safety analyses are subject of automation.

- Those that extensively use information stored in a uniform way by requiring that safety characteristics are produced using a modeling language.
- Those where we learn from the results of the analysis (an organized table, a graph, or some meaningful ciphers) more than from the process.

We are carrying out our work ([4], [5], [6], [7]) in the context of air navigation systems (ANS) where a component and model-driven based development is suitable and where safety needs are significant. Eurocontrol works to create regulatory requirements (ESARR) and guidelines to improve the safety of air navigation. One of these guidelines is the EATMP Safety Assessment Methodology (SAM [2]) which does not address certification issues but prepares and supports a certification process as intends to be a means of compliance to ESARR 4. In a cyclic and evolutionary process like the one proposed by SAM, automation could be of great importance given that the same safety analyses have to be reworked several times.

The purpose of our work is to aid safety and software engineers to find the quality software architecture that best meets cost and safety concerns taking into consideration the trade-off between them. We propose that safety and software teams work with the same models, the ones of the model-driven approach, but now including safety characteristics. Working with the same models improves the communication between them by having the same vocabulary but also avoids the need to keep model consistency. Safety engineers can see one or more “safety views” of the architecture necessary for their analyses, while software engineers can work with the same models they were using, but with the recommendations of the safety engineers. Automatic safety analyses allow for the evaluation of the architecture at zero-cost so that the safety team can propose means to eliminate or prevent hazards at any time. Both teams will choose the architecture and the combination of mitigation means that best fulfill safety constraints with the minimum cost. Due to the use of software models as the starting point for the integration we get the extra benefit that the relationship between models and code could be established.

This paper introduces the tools we developed to support safety analyses of architectures. Further clarifications about safety are detailed in [7]. Section 2 gives an overview about the use of safety analyses during the specification of the architecture in a MDA. Section 3 describes how to create analyzable architectures by annotating development models. Section 4 illustrates the languages for modeling FMECA and FTA. The model transformation to create safety analysis models from annotations is presented in section 5, whilst section 6 briefly shows the current implementation.

2. Application of Safety Analyses in Model-Driven Development

Safety engineers can support software engineers in the election of an architecture that fulfils safety requirements with the minimum cost; and they can do it all along the

process of architecture specification. Performing safety analyses from the beginning of the architecture specification has the following goals:

- Assess the safety of a system architecture form the beginning,
- Reduce the development cost of safety critical parts of the system.

Consequently, safety analyses proposed here do not have the (sometimes only) objective traditional safety analyses have of demonstrating the assurance of a system to a certification authority, even though they can prepare and support a certification process. Safety analyses in this paper must be considered in the bounds of preliminary safety assessments. This sort of assessments cope with the risk that system implementations do not fulfill safety requirements because of a bad design, but they can also be used to deal with the trade-off always found between safety and cost by supporting the identification of the architecture which achieves the best balance.

The process starts after system engineers have modeled a prototype of the architecture. Safety engineers can annotate this prototype with safety goals of the system (safety objectives), restrictions the system has to fulfill (safety requirements), ways some parts of the system can fail (failure modes), etc. They have to make the annotations in a formal manner to allow a later export of data. Automatic generation of safety analyses from the models of the architecture is the key of the process to avoid building the safety analyses each time he wants to assess the last version of the architecture. Safety analyses have to be presented in a way familiar to safety engineers given that they can be used in documents for safety certification. Both, system and safety engineers, can communicate using the same “documents”, the models of the architecture, to propose improvements if the architecture does not fit the requirements in terms of cost and safety. It is convenient that results of the analyses are incorporated back into the architecture as, again, formal annotations. This process can be automatic as well, but for the moment has not been implemented. The whole process is a cyclic procedure that ends when the architecture is evaluated as able to meet all the requirements (functional, non-functional, cost and safety).

2.1. Using MDA in the Process

Automatic generation of safety analyses from UML models requires the integration of at least two tools: a modeling tool (where the architecture is expressed) and another tool used to perform safety analyses. For many reasons, the best way to perform the integration is with the support of MDA standards, processes, and guidelines. Fig. 1 shows Meta Object Facility (MOF [12]) modeling stacks which represent the integration.

The architecture (Air Traffic Management models) is defined based on the UML meta-model and a safety profile. On the other hand, the safety analysis models (Fault Tree Analysis and Failure Modes, Effects, and Criticality Analysis) are based on a safety analysis meta-model. Both meta-models can be expressed in MOF. Since we are working with a component-based architecture, the instances of the system are components in execution. In contrast, instances of safety analysis models are failures of the system, if they unfortunately happen. A key transformation is necessary to convert safety-annotated system models into safety analysis models. These two types of models represent completely different concepts, hence its difficulty and

importance. Other transformations are required to adapt integration models to models usable within the tools (modeling and safety).

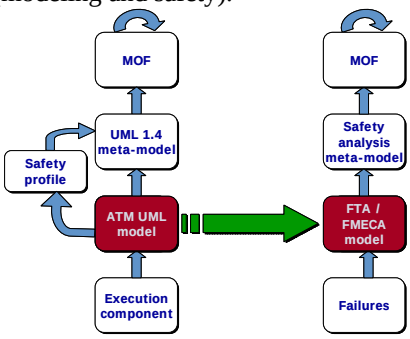

Fig. 1. MDA Rationale.

3. Creation of Safe-Aware Architectures

The set of annotations we introduce in this section assumes: software architectures based on components, models of the architecture that include the capabilities supported in the system, each capability is supported by a set of logical components, and each logical component is supported by a set of physical components. Furthermore, we assume the existence of modeling facilities to support traceability among previous elements.

To model safety concepts we have preferred to integrate safety properties in the general description of software architecture rather than to create specific views or models for the safety analyses. This enables us to readily relate software models and safety properties. To support this approach we have reused standard profiles ([4], [11]) and we have created a UML profile that directly represents safety concepts.

3.1. Safety Concepts

Safety concepts identified so far are pretty tailored to our aim of applying safety analyses in a preliminary safety assessment like the one proposed by Eurocontrol. The following are the main concepts we need for creating such safety analyses. Future work will provide more general concepts to be able to apply other safety analyses to software architectures, and thus serves as more general assistance.

- A **safety objective** is resulted from hazard analyses and defines a safety goal. When the goal is broken a hazard occurs, the effects of which reveal a severity. Indicators used to discern the severity of the effects are: exposure (exposure time, and number of aircraft exposed), recovery (annunciation, detection, contingency measures, and diagnosis), rate development of the hazardous condition, etc.
- A **safe-aware capability** is a software capability with safety objectives associated or that can affect some of them indirectly.

- **Safe-aware component**s represent logical and physical components that support safe-aware capabilities. A safe-aware component has a software assurance level (SWAL) used to establish the level of confidence that its implementation needs to accomplish. A good level of confidence means a disciplined process that limit the likelihood of development faults that could impact safety.
- A **safety dependency** is a safety-related relationship of cause-effect between two model elements. They can be used to assert, as proposed in [7], that a safe-aware component affects the safety of another safe safe-aware component, that a safe-aware component affects the safety of a safe-aware capability or that a safe-aware capability affects the safety of another safe-aware capability.
- **Mitigation means definition**. Mitigation means allows avoidance, detection, propagation control or mitigation of failure effects. The definition of a mitigation mean characterizes it independently of its application, according to: the phase where it has to be applied (topic), the type of mitigation (failure control), and the level where it has to be applied (application level).
- **Mitigation means application**. The application of mitigation means on safe-aware components provides a way to reduce the risk considering component's failure modes and the particular mitigation mean applied. The combination of a specific set of mitigation mean in a component produces a specific risk reduction.

4. Safety Analysis Meta-models

Two modeling languages have been defined so far: a FMECA and a FTA meta-model. Models created based on them are agnostic to any specific implementation of these types of analyses in a safety tool.

4.1. FMECA

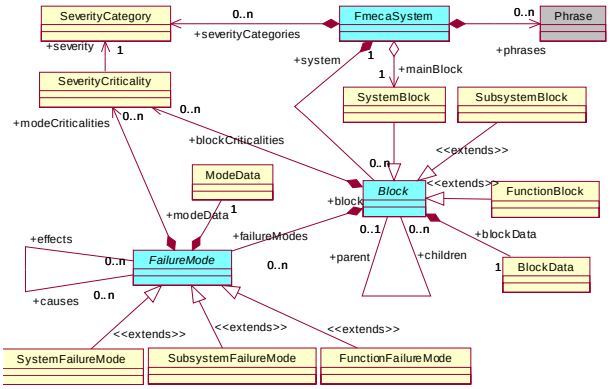

Fig. 2. FMECA safety analysis meta-model.

Failure Modes, Effects, and Criticality Analysis is an inductive approach to system design and reliability. It identifies each potential failure within a system or manufacturing process and uses severity classifications to show potential hazards associated with theses failures. There are 3 main elements in a FMECA analysis:

- `FmecaSystem`. FMECA is constructed as a hierarchy of blocks. FMECA system holds all the properties global to the analysis and the main block of the system.
- A `Block` represents every component in the hierarchy. Blocks are specialized as `SystemBlock`, `SubsystemBlock`, and `FunctionBlock`. This specialization is done because the same parameters in different blocks can have different meaning; i.e. a severity in a block can be input or result of the analysis.
- `FailureMode`. Each block can have associated several failure modes. These failure modes can be primary or derived from a cause-effect relationship. Primary failure modes are represented with the class `FunctionFailureMode`, while derived ones are usually represented with `SubsystemFailureMode`. End-effects are the last step in the process of derivation of failure modes and they are represented with the class `SystemFailureMode`.

4.2 FTA

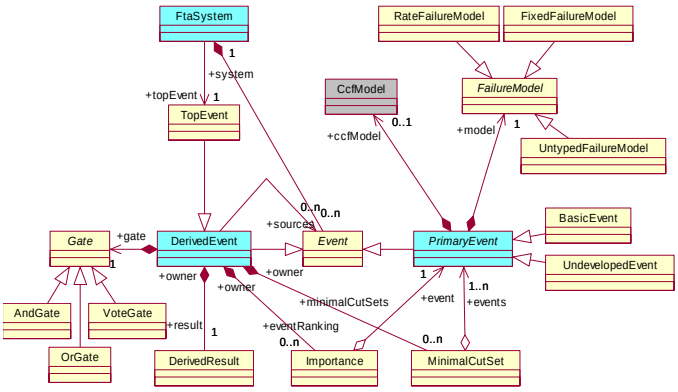

Fig. 3. FTA safety analysis meta-model.

Fault Tree Analysis is used during the safety assessments to represent the logical interaction and the probabilities of occurrence of component failures in a system. It provides quantitative information (probability theory is used), and qualitative (in the way of minimal cut sets: combinations of events leading to failure of the system). We can recognize 3 main classes in the diagram: FtaSystem, Gate, and Event.

- `FtaSystem`. A FTA system holds all the properties global to the system. The first thing to be done when starting a fault tree analysis is to define the hazard to be analyzed, which is represented as the `TopEvent`.
- `Gate`. A gate is a logical function of some inputs. Typical logical functions are AND, OR and VOTE. The output of a gate is a derived event of its inputs: either `PrimaryEvents` or the output of other gates (`DerivedEvent`).

- `Event`. Main events in a fault tree analysis are the PrimaryEvents. Primary events have associated a failure model. The most typical failure model is `RateFailureModel` created from a constant failure and repair rate.

5 Safety Analyses

There is no single way to perform a safety analysis from safety characteristics identified such as hazards, failure modes, mitigation means, etc. Even two safety analysts can perform FMECA or FTA in different ways according to their purposes, contrasting with other fields like in real time system where the purpose is clear: "make the system schedulable". Our purposes for safety analyses, framed in a general process of a preliminary safety assessment, are: to evaluate how an architecture can cause some hazards to occur, to estimate whether the architecture can fulfill the safety objectives of the system, to allocate SWALs to software components to accomplish these objectives, and to discover mitigation means for preventing hazards to occur.

For the creation of safety analyses models from safety annotations in UML we convert (Fig. 1) development models into safety analysis models. We designed this transformation using a set of rules. Next subsections will describe succinctly how it converts UML entities (system elements and safety annotations) into safety analysis meta-models entities (for instance blocks and failure modes for FMECA and events and gates for FTA).

5.1. FMECA Models Creation

Safe-aware capabilities and safe-aware components from system models shape the structure of a FMECA model by constituting the FMECA blocks. A function block can only represent a bottom-level safe-aware component (that one which does not depend on any other). A top-level safe-aware capability (that one with at least one safety objective associated directly) is represented by a top-level sub-system block. The only way to include a mitigation mean in a FMECA model is in unison with the safe-aware component it is affecting. A safety objective is an invariant that the system must fulfill, so that when it is not fulfilled we identify a failure mode of the system (the system block). In a preliminary safety assessment failure modes of safe-aware components are hard to identify, and we use some common keywords like corruption, loss, and error; even though other schemes are valid. In FMECA, a dependency among blocks is a parent-child relationship whilst a dependency among failure modes is a cause-effect relationship (limited from a child to its parent). In system models, safe-aware elements are interrelated by means of safety dependencies which we use to create the hierarchy of the analysis model. This makes necessary to replicate FMECA model elements in some cases; for instance, when a safe-aware component affects two safe-aware capabilities we create two blocks for the safe-aware component.

5.2. FTA Models Creation

FTA models should start by defining a hazard. As we said for FMECA, the non-fulfillment of a safety objective constitutes a hazard. Consequently, it seems obvious that a different FTA model has to be created for each safety objective in the system, so we will end up with several FTA models for only one system. In FTA it can be reasonable to create a FTA model from a lower-level hazard although this is not supported by our FTA model generation. Primary events will represent the failure of a bottom-level safe-aware component. The other important element in FTA, gates, will represent the failure of a higher level safe-aware component or capability. Since there is not more information included in safety dependencies we must suppose the worst case and use the OR gate; a gate will produce true if one of the components supporting it fails. Mitigation means impede failure propagation in a degree according to the risk reduction, so that they are represented as AND gate. Only those failure modes, declared in the mitigation mean definition, will be mitigated.

5.3. Example

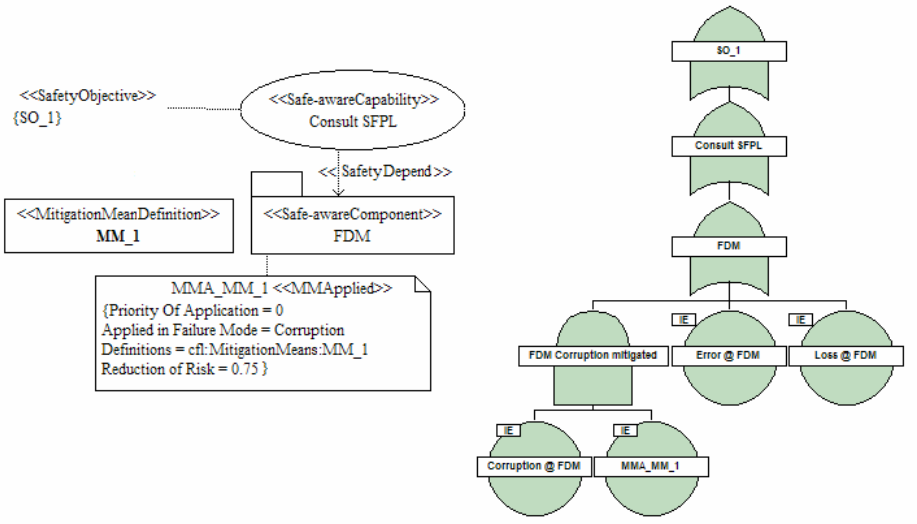

Fig. 4. Development model and annotations on the left. FTA in the safety tool on the right.

We will study a simple but comprehensive example with only one capability, “Consult SFPL (System Flight Plan)”, by which an ATC controller request information about a flight plan. This capability needs to fulfill a safety objective, “SO_1: The probability of detected and undetected corruption for greater than 5 minutes shall be no greater than 10-5 per hour”. The capability is hypothetically supported by a single component, “FDM (Flight Data Management)”. In order to achieve the safety goal the safety engineers have allocated a mitigation mean to this component “MM_1: Processing servers are replicated on different nodes”. Fig. 4 illustrates how, with the help of the safety profile, the constraint SO_1 is linked to the

capability Consult SFPL expressed as a use case. We represent the component as a package annotated with the stereotype SafeAwareComponent. A constraint with stereotype MMApplied indicates the application of MM_1.

The model transformation currently automates our particular proposal for the construction of FMECA and FTA. Previous annotations constitute the input of the transformation, whilst Fig. 4 shows the output for a FTA after been imported into a safety (and graphical) tool. This analysis will assess the safety objective SO_1 which is transformed into a TopEvent. “Consult SFPL” and “FDM” are mapped into DerivedEvent resulted from OR gates, whereas “Error” and “Loss” failure modes of FDM into BasicEvents. Since the failure mode “Corruption” has been mitigated, an AND gate is used to create a reduction of the failure mode’s probability. This means that the corruption of FDM will only show up at system level when both the failure mode occurs and the mitigation mean does not impede its propagation.

This is a simple example with a few elements and thus one could easily create manually the FTA within the safety tool. The potential of the automatic generation of safety analyses becomes apparent when the system model grows or/and when there is real benefit to maintain consistency between system models and safety analyses. During validation we tested the analyses generation with a large safety model: 21 safe-aware capabilities, 19 safe-aware components, 5 mitigation mean definitions and 9 mitigation mean applications were found in a system model much larger. The inherent evolutionary property of PSSA make of this environment an ideal test.

5.4. Results

The last step of the process would be the assessment of the analyses by the safety staff within the safety tool. This tool will provide tables (FMECA), trees (FMECA) and diagrams (FTA) depicting the models. Safety analysts can now inspect them to see how the elements of the architecture related with safety work together. FMECA can indicate the severity of function failure modes by setting the severity of system failure modes. FTA can provide us the list of minimal cut sets, i.e. the combinations of bottom-level failures leading to a hazard when happening together; we should specially address combinations with only one element.

One of our final aspirations with the safety analyses is to allocate SWAL to safe-aware components. Reducing the SWAL of a component means reducing the cost of its development. Naturally it cannot be done arbitrarily and justification material has to be provided, such as safety analyses and mitigation means. Allocating a SWAL to a component does not imply calculating a failure rate for software; hence SWALs cannot be used by safety assessment process as can hardware failure rates. In case of hardware, we could map components’ SWAL to failure rates in safety analysis models, and thus the safety tool could provide quantity results.

The methodology provided by Eurocontrol ([2], succinctly explained in [7]) strictly forbids allocating a failure rate to the software and we have to assume that software fails. Allocation of SWALs is accomplished by looking at the overall system design in its operational environment. Therefore, it is considered as key to keep the link from the safe-aware component to the end effect and its maximum tolerable frequency of occurrence. Development models provide the link between a safe-aware component

and the hazard but not to the end effect, it can be easily obtained from other documents though. We have worked on the safety analysis creation to provide quantity results by automating some hand-made processes ([7]). Quantity values can be used to derive SWALs.

6. Current Implementation Scenario

As previously mentioned, two tools had to be integrated (see Fig. 5): a UML modeling tool and a safety tool. Current implementation uses Eclipse Modeling Framework (EMF [3]) as common framework: to create meta-models, instance repositories, and to implement transformations. We use a UML modeling tool (it could be either a UML 1.X tool or a UML 2.0 tool) to create ATM models, to define and deploy profiles, and to export models to EMF. Finally, we also use FMECA and FTA modules of a safety tool to perform analysis calculations, create reports, etc.

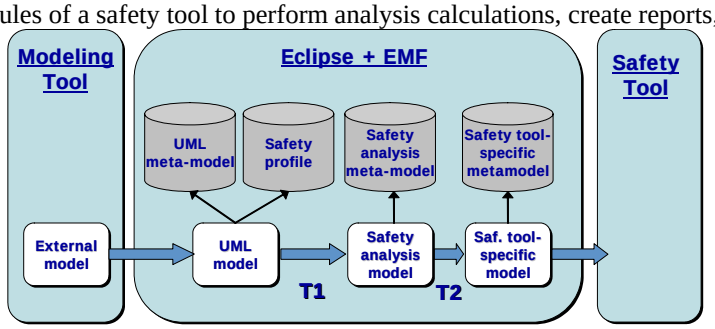

Fig. 5. Implementation Scenario.

7. Related Work

Other approaches exist for integrating safety engineering into the software development cycle. Most of them agree to highlight the benefits of applying safety techniques early in the development cycle and the drawbacks of its appliance at code level in large scale systems. Some of the approaches are based on development models. [14] explores the features, within UML, applicable to the development of a safety case for a product under development. After comparing formal methods and a rigorous process, the paper pragmatically decides not to introduce truly formal methods in the development cycle because of the additional complexity without tangible benefit. However we demonstrate how to get some benefits in preliminary safety assessments without affecting complexity.

Precisely modeling safety semantics is required to later analyze safety. The UML profile for quality of service and fault tolerance [11] includes some notations for the description of risk assessment, but it limits the scope to safety mitigation means and fault tolerance. There are also similarities between our work and the UML profile from SAE Architecture Analysis & Design Language (AADL) [13]. AADL is a language used to design and analyze the software and hardware architecture of real-

time systems and their performance-critical characteristics. The UML profile for AADL expresses AADL concepts in UML, whilst we directly express safety concepts in UML. AADL integrates reliability and safety analysis methods, by equipping components with reliability models, which are Markov chains that relate fault events and error states. Jürjens [17] include safety requirements in the UML models for a later analysis of requirement satisfaction. Some other work in York University [18] present very useful concepts for expressing safety constraints using OCL.

Few (if any) safety standards utilize visual modeling techniques provided by UML, so formal methods and safety analyses are necessary to fulfill standards. Pai and Dugan propose UML extensions for the description of software redundancy, reliability dependencies, and reconfigurations; they also tackle the transformation of UML models into Fault-Tree Analysis (FTA) models [16]. These notations are useful for the analysis of systems that mitigate risk with redundancy methods but they do not consider other mitigation means. The extensions are used to annotate deployment modeling elements and not architectural modeling elements such as components. Other works use UML activity, sequence and state diagram to generate FTA [19].

8. Summary and Discussion

We think safety and software engineers can work together to complete the specification of the software architecture with the tools presented in this paper. Automatic safety analyses are fundamental to succeed since the architecture evolves during this stage and safety analyses need to be reworked for each evolution, nevertheless some potential issues may arise as a consequence of the way humans interact with automation. For instance, safety analysts need to learn how to formalize safety parameters in the architecture with the UML profile.

We consider that separation of safety and safety analysis modeling is a must. Some safety analyses can be considered as different arrangements of elements from a well-known safety vocabulary (hazard, fault, failure, failure propagation, etc.). Safety analysis meta-models enable to model safety analyses. Our UML profile is a suitable instrument to model safety, rather tailored to Eurocontrol although can be extended.

Modeling and characterizing safety vocabulary enables to transform safety models into safety analysis models. The way we create the analyses can be fairly general to an MDD development in the duty of evaluating high-level models; nevertheless, work may be necessary to adapt the creation of safety analyses to our needs. Safety engineers could assist in this transformation. Having safety analysis meta-models independent of a specific tool offers us the possibility to change the safety tool with less effort and knowledge of safety than if we directly map from the safety meta-models to the safety tool.

Data for safety analyses could arise from system model elements, for instance functional dependencies between components. Although it is not a trivial task it could be automatically detected when a functional dependency is a safety dependency as well. Two safety concepts could be seen as trivially considered in this work: component failure modes and safety dependencies. Failure modes might be fully described whilst token words used instead are only clues for their identification.

Depends-on relationships among components used here cannot be enough to find mitigation means to stop propagation. Nevertheless these two concepts can be enough for a preliminary safety assessment.

Acknowledgements. The work presented here has been co-funded by the European Commission under the IST 6th FP 2002-2006 (MODELWARE project [1]), and by the Spanish Ministry of Education (TIC2005-08665-C03). We would like to thank our partners, especially Thales ATM. This paper reflects only the author's views.

References

1. Modelware Web Page: http://www.modelware-ist.org/
2. European Organization for the Safety of Air Navigation, "Air Navigation Systems Safety Assessment Methodology", http://www.eurocontrol.int
3. F. Budinsky et al., "Eclipse Modeling Framework", Addison Wesley Professional (2003)
4. M. de Miguel, B. Pauly, T. Person, J. F. Briones. "Model-Based Integration of Safety Analysis and Reliable Software Development," words, pp. 312-319, 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems WORDS (February 2005)
5. J. P. Silva, M. de Miguel, J. F. Briones, Alejandro Alonso, "Safety Metrics for the Analysis of Software Architectures", Workshop on Visual Modeling for Software Intensive Systems (VMSIS) at the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (September 2005)
6. M. de Miguel, J. F. Briones, J. P. Silva, A. Alonso, "Model Based Integration of Safety Analysis and Development", 9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing ISORC (April 2006)
7. J. F. Briones, M. de Miguel, J. P. Silva, A. Alonso, "Integration of Safety Analysis and Software Development Methods", 1st IET Conference on System Safety (June 2006)
8. MIL-STD-1629A, "Military Standard, Procedures for Performing A Failure Mode, Effects and Criticality Analysis" (1980)
9. NUREG-0492, "Fault Tree Handbook", U.S. Nuclear Regulatory Commission (1981)
10. N. Levenson, "Safeware: System Safety and Computers", Addison Wesley (1995)
11. Object Management Group, "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms", OMG document number ptc/2005-05-02 (2005)
12. Object Management Group, "Meta Object Facility (MOF) Core Specification", OMG document number formal/2006-01-01 (2006)
13. AADL, "SAE Architecture Analysis & Design Language" http://www.aadl.info/
14. ARTiSAN Software Tools, "Safety in the Loop" (2002)
15. K. Khan, J. Han, "Composing Security-Aware Software", IEEE Software (January 2002)
16. G. Pai, J. Dugan, "Automatic Synthesis of Dynamic Fault Trees from UML System Models", International Symposium on Software Reliable Engineering, IEEE Computer Society (2002)
17. J. Jürjens, "Developing Safety-Critical Systems with UML", LNCS 2863 (2003)
18. P. Conmy and R. Paige, "Using UML, OCL and MDA to support development of Modular Avionics Systems", Workshop on Critical Systems Development with UML at UML 2004 (October 2004)
19. M. Towhidnejad, D. Wallace, A. Gallo, "Validation of Object Oriented Software Design with Fault Tree Analysis", Software Engineering Workshop, 28th Annual NASA Goddard, IEEE Computer Society (2003)