Saving the Software Specification by Converting the old SA/RT Models into UML

Thomas Weyrath
Elektroniksystem- und Logistik-GmbH
Email: Thomas.Weyrath@eurocopter.com

Herbert Schreyer
Eurocopter Deutschland GmbH
Email: Herbert.Schreyer@eurocopter.com

Abstract

The avionic software of the both military helicopters Tiger and NH90 was developed during the 1990s. Their specifications based on Structured Analysis / Real Time (SA/RT), a methodology that was state-of-the art in the 1980s. But nowadays the tools for modeling SA/RT are no longer supported and the knowledge to understand and maintain the models is going to diminish over time. Hence, ways must be found to keep the expertise on the specifications, models and software for future work. In this paper, we describe with the example of software design models, how we want to convert our software models from SA/RT to UML.

Topic: ***Avionics***
Keywords: ***Model-based system engineering, UML, SA/RT***

1 Introduction

The System Support Centre NH90/Tiger (SSC) is a cooperative organisation of German Federal Armed Forces, Eurocopter, ESG, Cassidian and MBDA. It comprises a team of 200 engineers. Its main task is the avionic software maintenance and modification of the military helicopters NH90 and UH Tiger. The maintenance comprises more than 12.000.000 lines of code in 16 avionic computers, including the ground support system. Additionally, the documentation of the development is immense. It contains over 500 documents of specifications and design descriptions. Most of them have hundreds of pages, some of them over a thousand. In view of the fact that the helicopters have a life cycle of more than 40 years, the software maintenance and modification is a challenge, but it has to be done.

1.1 Rationales behind our work

Over the years the helicopter software will be changed due to modifications in the software environment, new customer requirements or simply by fixing errors. Even an extensive facelift towards modern technologies cannot be ruled out. Some points are important:

- The software changes will have to be done efficiently and reliably. This includes that all state of the art processes and methodologies of software maintenance and modification have to be well known.

- Human expertise of the past and present must be maintained as long as our helicopters exist. The software development of both helicopters started in the 1990s. A lot of engineers of this time has gone into retirement or will be retiring in the next ten years. Their valuable knowledge must not be lost.

- Specifications must be improved continously. Most of them are either text-based or use Structured Analysis / Real Time (SA/RT). And, what is worse, the NH90 used the Harel whereas Tiger used the Hatley und Pirbhai approach.

- Without changing the methods and specifications future obsolescence problems occurring with tools, databases and formats are unavoidable.

- And - last but not least - when we change the specification we want to introduce well-accepted software concepts, like modularization, structuring and abstraction to improve the readability and understandability.

1.2 What have we done?

With the help of external consultants, the SSC has established a customized UML methodology for software requirements analysis and software design, which refines an EADS guideline for using UML (see [3]). This methodology is described for the part of the requirements analysis in [4].

After this, those consultants have trained SSC employees in both UML and UML methodology. In addition, the methodology has been successfully applied in the context of a redesign of an embedded avionic computer. Furthermore, it has been applied in several software projects of the SSC that aim at improving the testing environment of the airborne computers.

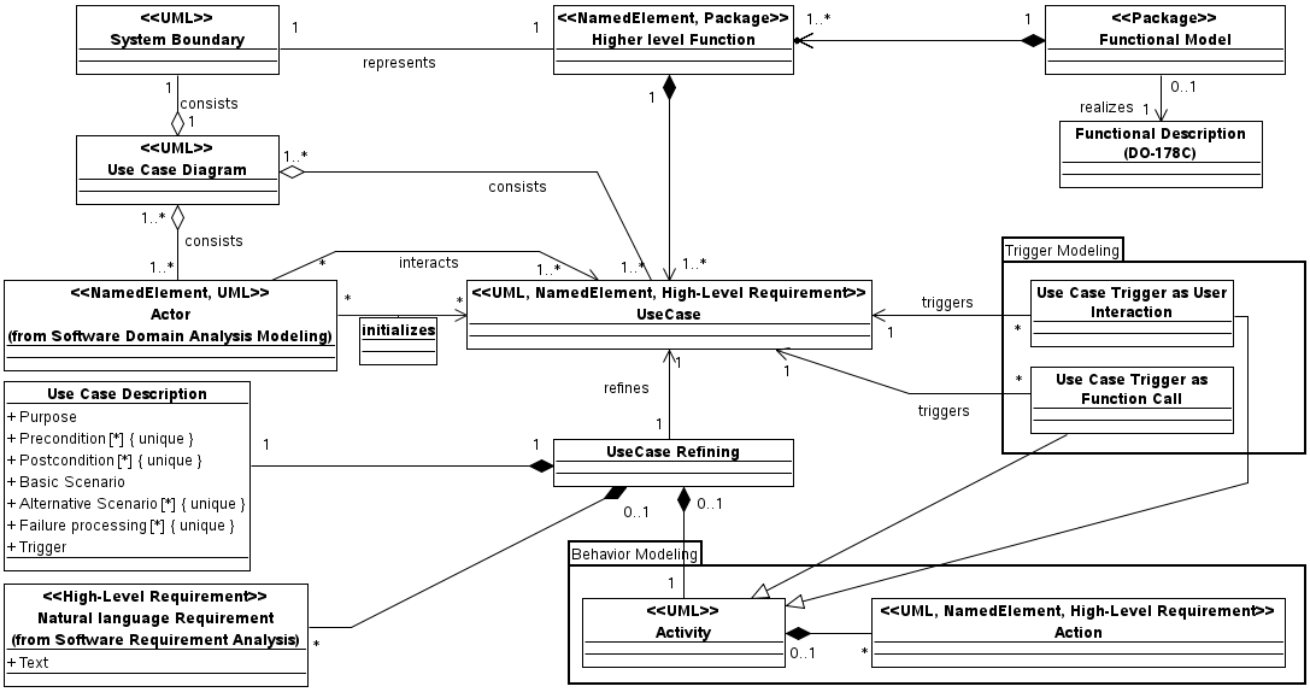

Figure 1. Meta model of the software use case modelling.

2 Software Requirements Analysis

2.1 Software Use Case Modeling

For the requirements analysis the functional aspects of the software under development are most important. Therefore we focus on a functional approach (instead of an object-oriented analysis). This is closer to the used development standards, DOD- 2167A and DO178-B [2].

The methodology is use case driven and defines the following steps:

1. Decompose the software system into high level functions to handle the complexity of the model.

2. Perform a use case analysis for each high level function.

3. Refine the use cases with use case descriptions and further natural language requirements and model the use case behavior with an activity diagram.

4. Model the activity that triggers the use cases. That can be function calls, e.g. from an external interface, or an user interaction event.

Figure 1 shows the summary of the key concepts used in the software use case modeling. Key concepts are the use cases, use case diagrams, actions and activity diagrams that specifies the high-level requirements expressed by a specification model.

2.2 Software Domain Analysis Modeling

The software domain analysis modeling refines the results use case modeling and defines the domain model consisting of actors, terms and data types.

2.3 External Interface Modeling

The external interface modeling addresses how the software interact with people, the systems hardware, other hardware, and other software [1]. The interface modeling is limited to the software interfaces that describe the data inputs and outputs between the software system itself and its external software systems.

3 Software Design Modeling

3.1 Software Architectural Design

The goal of Software Architectural Design is to establish the structure of the software. It defines software partitions, identifies software components, their interfaces (provided/used) and defines protocols that are required to communicate with the environment of the software.

The architecture design ist subdevided in five steps:

1. Create the Black Box View to identify the services of the system and its external systems (actors) to establish the system boundary.

2. Create the Decompostion View to show the hierarchical structure of the software.

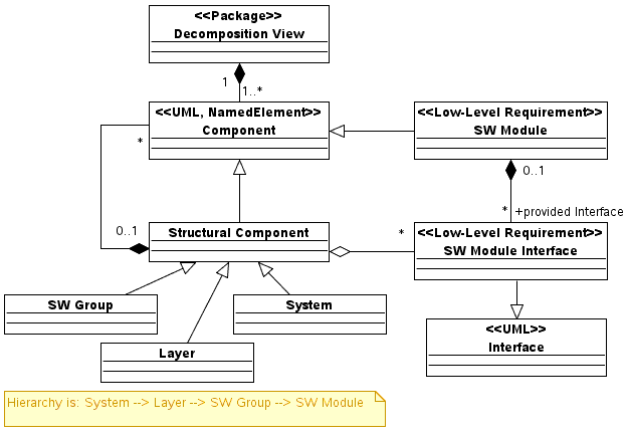

Figure 2. Meta model of the decomposition view.

3. Create the Functional View to elaborate how the use cases specified in the requirements analysis are implemented using the software modules.

4. Create the Process View to describe the dynamic aspect of the software system.

5. Create the Deployment view to show how the artefacts are distributed among the involved hardware components.

Figure 2 shows the summary of the key concepts used in the decomposition view. It consists of a set of component diagrams that shows the hierarchical structure of the software in form of layers, software groups and software modules. This view allows software developers and maintainers to identify the major design constituents of the design subject and to localize and allocate functionality, responsibilities, or other design roles to these constituents.

3.2 Software Detailed Design

The Detailed Design View changed the viewpoint from the software system consisting of software modules to the individual software modules. The topic of this view is the structure and connection of the individual software module as well as the collaboration of its structure to implement the functionality of this module.

4 Traceability from the Software Design to the Software Requirements

The customized UML methodology identified and delimit the requirements contained in the model. Based on this information, we establish the traceability between high-level requirements of software requirements process and low-level requirements of software design process.

References

[1] *IEEE Recommended Practice for Software Requirements Specifications*, IEEE-830-1998.

[2] DO-178B. *Software Considerations in Airborne Systems and Equipment Certification*, December 1992.

[3] P. Gast and O. Bender. EADS GUIDELINE – using UML for software analysis and design. *EADS internal paper*, 2009.

[4] T. Weyrath, B. Schinnerl, F. Schoettl, and H. Schreyer. A new uml tool-based methodology for the software requirements analysis. In *European Congress Embedded Real Time Software (ERTS)*, Toulouse, France, 2012.

Author's Biography

Thomas Weyrath is project manager in the department for Integrated Systems in the Business Area Aviation at ESG. Since 2009, he has been working in the SSC and manages the process improvement project UML. Previously, he worked in the Automotive Area at ESG as software engineer and project manager for 9 years.

Herbert Schreyer works on the aviation system development of the Tiger and NH90 helicopters since 1995. 2005 he joined the SSC where he took over the role of a system architect. Since several years he is charge of evaluation and introduction of new methods of avionic definition.