AN EXPERIENCE OF DEVELOPING MISSION CRITICAL SOFTWARE FOR BIRD ATTITUDE CONTROL USING UML

Olaf Maibaum
DLR, Simulation and Software Technology
D 38108 Braunschweig, Lilienthalplatz 7

Mallikarjun S. Kande
ISRO Satellite Centre
Airport Road, Vimanapura, Bangalore – 560 01, India

ABSTRACT

The cry for quality and software release schedules are nightmares for the software engineers. The unification of last decade techniques in the Unified Modeling Language (UML) give a rise for stable platform for software modelling. We show how the UML fit the software engineering process and report the experience made with UML during the development of the BIRD attitude control system.

1. INTRODUCTION

There is a continuous evolution in the methodologies to overcome the problems associated with the software complexities. The aesthetic problems associated among the software team players is also a major challenge of the modelling the software. Keeping the behavioral model of the team members along with the software itself, many modelling tools were created. But even the best out of the best suffered many problems, such as learning time, ease of use, code generation capability, etc., keeping the software engineers reluctant to use these tools.

It looks like the UML is the first choice for modelling software systems in the beginning of this century. In the next section we give an overview of the phases of the software engineering process for mission critical software systems. In the third section we present a short overview of UML and associate the diagrams of the UML with the phases of the software engineering process. In the fourth section we report our experiences with the use of UML during the development of the BIRD attitude control system and close in the last section with our conclusions.

2. SOFTWARE ENGINEERING PROCESS

It's a myth that the software engineering process only consist in programming. Also the definition of requirements and the maintenance after delivery of the software system are part of the software engineering process.

The phases of the software engineering process for european space systems are determinate by the ESA software engineering standards [1]. The software engineering

standard define six phases for the software life cycle, the user requirements definition phase, the software requirements definition phase, the architectural design phase, the detailed design and production phase, the transfer phase, and the operations and maintenance phase.

In the user requirements definition phase the problem and the operational environment for the software system are stated. The next phase in the software engineering process is to define the software requirements which should be implemented by the software system under development. These phase include the construction of the logical model and identification of the user requirements. If all software requirements are defined, the architectural design phase can initiated. The result of this phase is a physical model of the software system with the major components.

After a stable physical model is defined the detailed design and production phase starts. These phase is composed by the module design, coding, and unit, integration, and system tests. These phase ends if all software requirements are implemented and tested.

The last two phases in the software engineering process consist in the installation and maintenance of the software system. These phases are out of the scope of this paper.

3. USAGE OF UML IN THE SOFTWARE ENGINEERING PROCESS

The unification of last decade techniques of software development like OOA, OOD, OOSE, and OMT gave rise to a stable platform for software modelling methodologies, mainly the Unified Modeling Language (UML) [2, 3]. The UML effort started official in October 1994. In January 1997 UML 1.0 was offered for standardization to the object management group (OMG).

UML was developed to achieved three goals:

1. To model system, from concept to executable artifact, using object-oriented techniques
2. To address the issues of scale inherent in complex, mission-critical systems
3. To create a modelling language usable by both humans and machines

The UML is build on a visual formalism which can translated by UML tools in machine readable form. It enclose several kinds of diagrams. In the following the important ones to model embedded real time systems in the first four phases of the software engineering process are described. An overview of developing real time systems with UML is given by Douglass[4].

.1. User Requirements Definition Phase

To investigate the user requirements in the first phase of the software engineering process the UML provides the use case diagram. The purpose of a use case diagram is to list the actors and the use cases and show which actors participate in each use case.

The term of an actor in an use case diagram do not match with an actor of an embedded system. The term actor stands for an active component of the system. These mean that an actor in an use case diagram can be an actor of an embedded system but also a sensor or another subsystems of the whole system.

.2. Software Requirements Definition Phase

The UML provide four kinds of diagrams to describe the logical model. The sequence, collaboration, statechart, and activity diagram. It's useful that for every use case at least one of these diagrams exists. If a use case permit several sequences to perform itself, for each sequence one sequence or collaboration diagrams should exist.

With the sequence and collaboration diagrams the interaction between the active components in the system can be described. Both diagrams show the flow of messages between the objects of the software system. A sequence diagram shows a set of messages between the objects arranged in a time sequence and the lifeline for the objects of the system. To deal with real time requirements Douglass [4] introduce some message stereotypes to describe the timing behaviour of the messages.

A collaboration diagram models the objects and the links that are meaningful in an interaction. The messages are shown as arrows attached to the relationship lines between the object, where each message is prepended by a sequence number to indicate the sequence of messages.

A statechart diagram models the possible life histories of an object. It shows all possible states for an object and transitions between the states. The transitions are annotated with the message which trigger the state transition and a condition and action to perform the state change. Also the states can annotated with entry actions, exit actions, internal transitions, and substates. The main use of a statechart is to describe reactive subsystems.

A variant of a state machine is an activity diagram that shows the computational activities involved in performing a calculation. An activity state represents an activity, for example the execution of an operation. This type of diagram allows to describe parallel operations.

.3. Architectural Design Phase

The primary diagram for the architectural design phase is the component diagram. The component diagram groups all defined objects in the software requirement definition phase to system components and show the interfaces between the components.

To partition the components a deployment diagram can be used, which group the components into nodes. This partition of the components is for example useful to spread the components among several nodes of a multiprocessor system.

Both diagram types act as medium to model the physical model of the software system under development.

.4. Detailed Design Phase

The main usage of the UML in the detailed design phase is to describe the objects of the system and the relationships between them in detail. For this purpose the UML provide two kinds of diagrams, the class and the object diagram.

The purpose of the class diagram is to describe the static view on a component. Each class contain the attributes and the methods of a class. The attributes specify the state of

the class. The methods are the interface for other classes in the system to use the class. The relationships between the classes are generalization/specialization and association.

The other kind of diagram, the object diagram, is used in the detailed design phase to describe a snapshot of a component. An object diagram is only useful for dynamic systems where objects are created and destroyed during runtime.

4. EXPERIENCES

The software development team of the BIRD attitude control system have decided to use UML as an aid for discussion and documentation of the software system. As diagrams the use case, sequence, component and class diagram as well as the statechart will utilized. But the sequence diagrams will only used to verify the behaviour of a component. The software requirements exists only in a textual manner or for mode control as statechart.

The main problem in the usage of the UML was to find an UML tool which fulfil the requirements for distributed work in a Linux environment. Also it should be possible to integrate the diagrams in the software documentation and to generate code by the tool. However no available tools have to fulfil all of these requirements. As decision the UML tool «Software through Pictures» (StP) from Aonix on a Solaris server was used.

This tool enable a distributed work and the generation of the diagrams in postscript and RTF format. Also it can used by a remote login from a Linux host. The exchange of the diagrams between team members is possible directly in the project repository of the StP. The automatic code generation of the tool was not used because the code do not fulfil the coding standards of the BIRD project.

The advantage of the usage of UML was a common base for discussions about architectural and detailed design between team players with a different native languages and from different scientific back ground. The usage of the UML enforce the modelling of the software structure before coding of a component starts. Also the visual presentation act as a guide in the code jungle during software implementation. These advantages yield in a rise of the software quality and maintenance of mission critical software parts of the BIRD micro satellite.

5. CONCLUSION

The usage of UML to model mission critical real time software support every step in the software engineering process. It promote the communication between participants in a project team with different backgrounds. The visual representation of UML lead in a clear software structure which enforce the maintenance and software quality.

The available UML tools are still under development. But we think that next generation UML tools will fulfil all requirements of the software development process. With such tools the UML becomes the main instrument of software development of mission critical software in the beginning of this century.

6. REFERENCES

1. ESA; ESA Software Engineering Standards; ESA PSS-05-0 Issue 2, February 1991

2. Grady Booch, James Rumbauch, and Ivar Jacobson; The Unified Modeling Language User Guide; Addison Wesley Longmann, 1999
3. James Rumbauch, Ivar Jacobson, and Grady Booch; The Unified Modeling Language Reference Manual; Addison Wesley Longmann, 1999
4. Bruce Powel Douglas; Real Time UML; Addison-Wesley Longmann, 1998