World Academy of Science, Engineering and Technology
International Journal of Electrical and Information Engineering
Vol:9, No:11, 2015

A Study of Quality Assurance and Unit Verification Methods in Safety Critical Environment

Miklos Taliga

***Abstract*—**In the present case study we examined the development and testing methods of systems that contain safety-critical elements in different industrial fields. Consequentially, we observed the classical object-oriented development and testing environment, as both medical technology and automobile industry approaches the development of safety critical elements that way. Subsequently, we examined model-based development. We introduce the quality parameters that define development and testing. While taking modern agile methodology (scrum) into consideration, we examined whether and to what extent the methodologies we found fit into this environment.

***Keywords*—**Safety-critical elements, quality management, unit verification, model base testing, agile methods, scrum, metamodel, object-oriented programming, field specific modelling, sprint, user story, UML Standard.

I. Introduction

IN the present study we examined development and testing environments in the development of software, for medical technology and automobile industry, that contains safety-critical elements. Embedded systems can be encountered in both fields and we found that development in an object-oriented environment is based on the V-model. We observed factors influencing the verification of the object-oriented environment and the difficulties that arose from programming over the full development cycle. Subsequently we examined model-based automated test- generation as a new field of research. A number of methods and tool may be mentioned regarding this field, including but not limited to: the STATEMATE state chart software tool, the AGEDIS (Automated Generation and Execution of Test Suites for Distributed Component-based Software) [4], the Mutation Analysis Technique combined with a model examiner, the abstract state machine (ASM), the BZ-Testing Tool, etc. In the present case study we examined ASM and UML based models in detail, thus, we focused on specification based testing criteria as quality parameters.

II. The Case Study

A. Service Safety

Service safety is crucial in both fields. In order to achieve it, the basic goal is creating useful, highly reliable (Telekom service, five nines 99.999%), reparable (maintainable) software. In safety-critical systems, standards determine the frequency of errors. Safety Integrity Levels (SILs) show the number of errors in safety-critical functions hour wise, that is, basically, the number of errors during a given life cycle. Service safety can be influenced by error occurring during software development and software operation. During software development, design and implementation errors may occur. Their numbers can be decreased by quality assurance and a better choice of development methodology. It must be taken into consideration, however, that the larger and more complex the software becomes, the harder it is to realize flawless design and implementation. For that reason, great emphasis must be put on both verification and validation procedures. If the software is already working, hardware error and operator error may still occur. Hardware error means a choice of hardware with improper specifications [2], [11]. To achieve service safety, the service provided by the software must be able to meet its requirements, which must then be verified by measurements and analysis. In software development, data security and service safety are closely connected, with both integrity and confidential data handling being important aspect.

B. Testing Related to Object-Oriented Programming

From the perspective of service safety, standards used in the development of embedded systems only provide by and large suggestions for verification and validation methods and tools. In practice, that meant that software developers chose software development technologies and environments that were well known to them. Also, methodological and technological tools to these development tools were at their disposal to plan design and execute appropriate tests. During the verification of object-oriented systems the system itself is tested, which requires mainly, input and output data from the system. These data, naturally, are derived from observations during system operation. However, the same principles of the OO programming paradigm that enhance code-recycling and efficient development, make testing more difficult, since every object is affected by encapsulation and data hiding. For that reason, the observed class or method must have a publicly accessible, open-to-all interface (or interfaces) that can be used to observe the behaviour of the object or class. Additionally, their data must still be possible to use in testing. The shared data space for observing certain systems used by the medical technology development company could be mentioned as an example.

As seen on Fig. 2, the shared memory space between the Display system and the Graphical User Interface can be used for the verification procedure.

Miklos Taliga is a PhD student with the Budapest University of Technology and Economics, Department of Control Engineering and Information Technology, Budapest, H-1117 Magyar Tudosok Korutja 2 Hungary (e-mail: taliga@iit.bme.hu, www.iit.bme.hu).

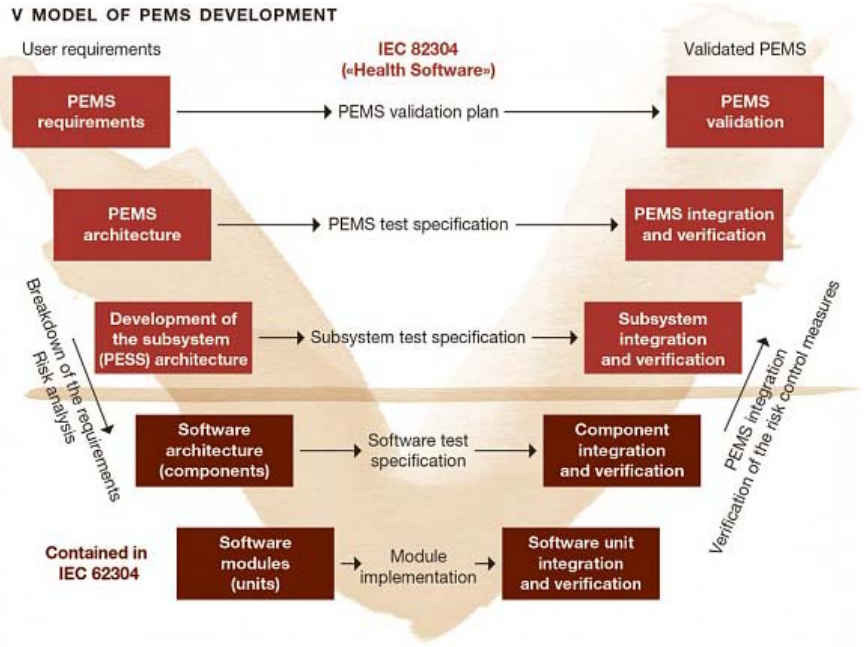

Fig. 1 V-model (IEC 62304 Standard)

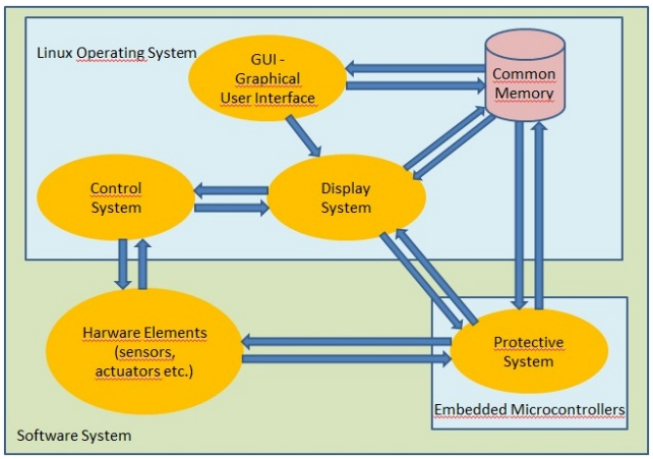

Fig. 2 Generated task environment

C. Field Specific Modelling for Testing

In field specific modeling, a static structure representing the problem at hand must be built by defining the meta- models. The qualities of the structural elements can be described by the properties of the static element attributes. The problem with defining meta-models is that it does not provide a realistic picture of the dynamic problems that arise with the model. It is thus necessary to create languages for the description of dynamics, which enable a more accurate and efficient formulation of problems in various fields.

Both the Finite State Machine (FSM) and the Timed Automation (or Hybrid Automaton) belongs to the dynamic descriptive languages. To be tested, the meta-model must be transformed into an Abstract State Machine (ASM) [8]. The transformation is done by semantic anchoring which results in semi-formal language that can be used to produce automatic test cases. The ASM check can be used to check if the dynamics description is correct. In large and complex systems the static structure of the system may affect the ultimate implementation: as a solution, the source model can be transformed to different field specific language [3].

Whether the system operates correctly can be checked in different simulation environments. One such environment is the MATLAB Simulink that is highly suitable for the simulation of embedded systems [1].

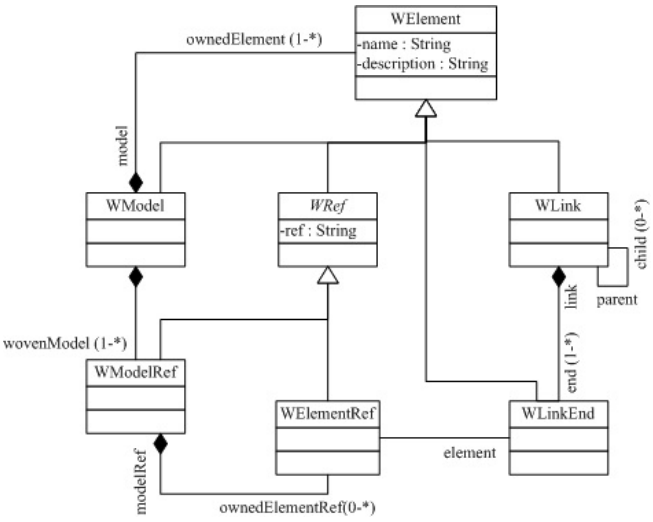

Fig. 3 Core weaving metamodel

D. Model Based Automated Test Generation

From the models belonging to test methods, we will examine the UML state chart model in further detail.

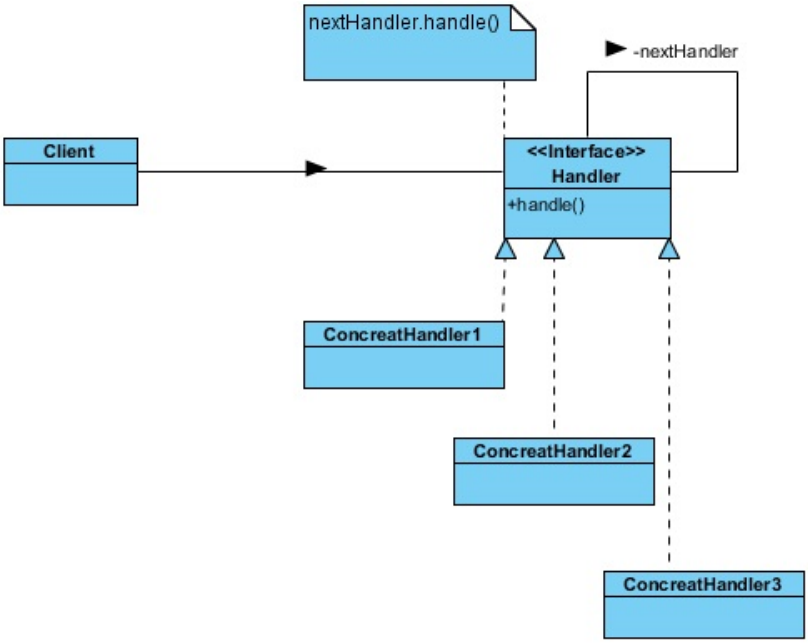

Fig. 4 Abstract State Machine – ASM

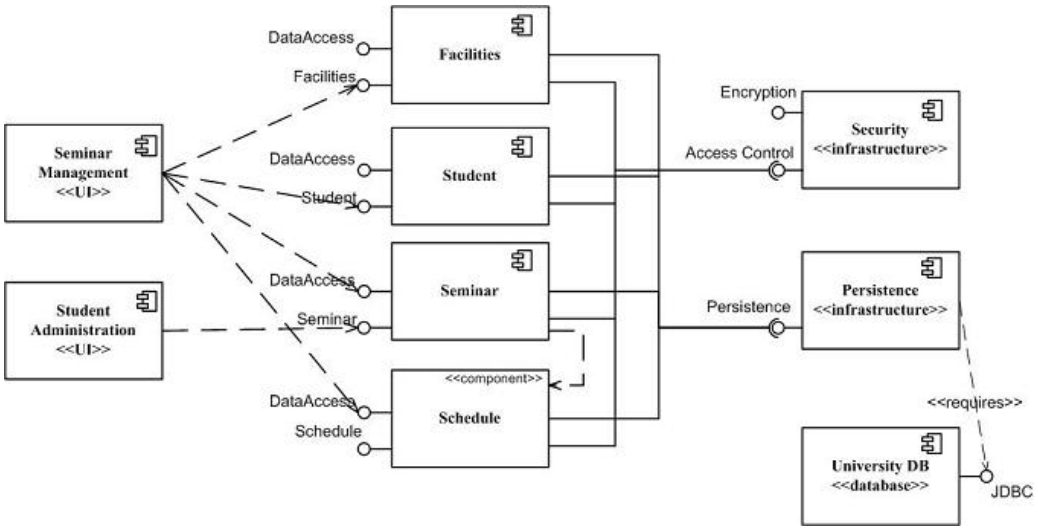

Fig. 5 UML 2.x component diagram

The advantage of this model is that the input/output points and interfaces of the system are more easily identified, and so are the system-related events. The model helps the developers in establishing a unified approach, and provides assistance in recognizing hidden inconsistencies within the specification. Temporal logic formulas are assigned to each test requirement based on a coverage criterion chosen previously in the model. The model created from the UML state chart is transformed to the input language of the examiner [10]. The model examiner, examines the negations of the determined formulas individually, one after the other, based on the test requirements. Running the model examiner results in a converted test series, possibly with negated instances because of the negation [6], [7]. As a high the number of test requirements are processed, negated instances must be highlighted in the test series. Model-based automated test generation can save time and money by generating and running test cases based on an automatically predetermined model, in a shorter time. Yet another argument for automated test generation is the fact that the generated tests can be swiftly reproduced.

III. Object-Oriented Development and Model-Based Testing in Agile Methodology

When examining development and testing systems, the principle of individuals and communication being more important than the development and testing methodologies and tools themselves must be taken into consideration. According to the next agile principle, a working software is more important than comprehensive documentation. In this case,

during automated test generation in embedded systems that may contain safety critical elements, it was important to know which part(s) of the documentation can be left out without compromising the test requirements of safety critical elements. In the case of both development and testing systems, emphasis can be shifted to testing working software if evaluation after testing is connected to automated documentation. According to the next agile methodology principle, close cooperation with the procurer if more important than a strict adherence to the contract. Finally, the agile methodology places more importance on adaptation to change than on following plans rigorously. Fig. 6 illustrates the importance of time, which fundamentally influences planning in development and testing (scrum) [9].

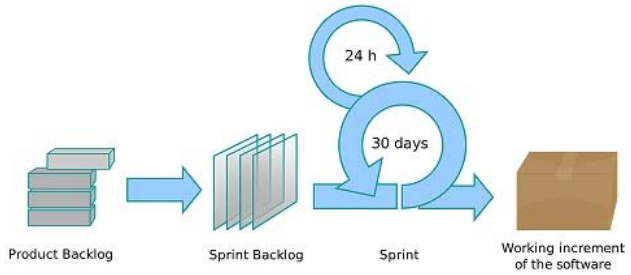

Fig. 6 Scrum process

A. Testing Related to Object-Oriented Programming

When inspecting the criterion "Delivering correct and valuable software to the procurer early and continuously", we found that V-model based testing mechanisms (verification and validation) fit the specification based testing strategy well. Currently, in the two fields of the present study, software developed and tested with the V-model did not contain automated testing tools. Additionally, whether we mention testing and development in automobile or medical technology industry, service safety must be of utmost importance in embedded systems, especially if the contain safety critical elements. No margin of error is acceptable, so the conditions of service safety must be taken into consideration. As that requires thorough and repeated testing, it is difficult to achieve using with the agile methodology. An early and continuous delivery of the software cannot be guaranteed. Automated testing tools that can speed up the testing process and thus ensure early and continuous delivery have to be installed in the system. When inspecting the criterion "the change of requirements is acceptable even in the late stages of development", we found that the development model must be flexible. The V-model fulfills those criteria, so change management used in this model can be utilized according to the agile methodology. Short period software deliver (possibly even cycles of a few weeks) always refers to completing a new product. In the agile methodology, communication between the procurer, the developers and the testers is especially important. When inspecting communication, we found that in V-model based development the procurer could not provide direct feedback during the entire development. In practice, the procurer gets the result of 2 or 3 months' worth of development for user acceptance test. Thus, this practice cannot be implemented in the agile methodology in its current form. In the development of software for automobile and medical technology industry, the development of modules handling complicated tasks may frequently be necessary. In such modules, testing in the early stages of development is difficult, mainly because these modules are highly dependent on other modules. For that reason, such modules may only be tested with complete dependence criteria. With the criterion "the main measure of progress is working software", the criterion of service safety must still be considered, as it is necessary in order to comply with the requirements of testing safety critical elements. The "constant focus on technical excellence and good planning" criterion can be fully adapted from V-model based development and testing.

B. Model Based Automated Test Generation

In the case of the criterion "Delivering correct and valuable software to the procurer early and continuously", we found that during model based test generation, functions to be developed must be organized into well-defined user stories, while paying attention to the time of sprint. Naturally, that influences model creation. If the creation of the model has fits the time of sprint, delivering software to the procurer early and continuously is possible. If the model in development is highly dependent on another module (e.g. in a module responsible for complex tasks), care must be taken so that even if it is dismantled into subsets based on dependencies, testing still remains definitely sensible and reproducible [5]. When examining the criterion "the change of requirements is acceptable even in the late stages of development", we found that model based test generation is fundamentally affected by the complexity of the model. The principle formulated in the first point still applies, only with the condition that if a multi-dependent module is affected by change in a late stage of development, it will affect model changes in all the modules it is dependent upon. In practice, that means the model of every dependent module has to be modified. That, of course, results in extra work and more time needed, which must be taken into consideration. The complexity of the mode is also important in the short period delivery criterion, with cycles only a few weeks long in model based test generation. During sprint planning, it is important to pay attention to what parts the function in development is separated into. With the criterion "the main measure of progress is working software", just like with object oriented development and testing, high service safety must be in focus. Yet again, the service safety can only be ensured is the requirements of the testing of safety critical elements are met. In the agile methodology, the criteria of working software can be achieved with designing the modules or functions defined in sprints, since user stories and their respective testing tasks guarantee the delivery of a continuously developed and tested module or function. Looking at the criterion "constant focus on technical excellence and good planning", techniques used in object-based development and testing and model-based development and testing can be perfectly adapted to the agile methodology.

IV. Conclusion

In the present case study, we examined the development and testing methods of systems that contain safety critical elements in different industrial fields. We identified and illustrated the quality attributes related to the different development and testing environments. In the following part of our research, we aim to create unified viewpoint system based on these quality attributes, that can provide assistance for development and testing teams working in the aforementioned fields, so that they are able to choose the development and testing toolkits and procedures that suit them best. Additionally, we examined how well the development and testing procedures used in different industrial fields fit into the agile methodology, which is quite widespread today. We have determined the conditions of possible utilization and demonstrated the cases in which case agile methodology is inappropriate for testing the software functions of a given product. Further research would be required in model-based automated test generation, as there is a growing need for such tools in the field of embedded systems, especially when the testing of safety critical elements are involved.

Acknowledgment

The author thanks a medical device developer company and automotive industrial field developers for helping the case study.

References

[1] MATLAB – Simulink http://www.mathworks.com/products/simulink/
[2] ISO/IEC 12207:2008. Systems and software engineering -- Software life cycle processes.
[3] IEC 62304:2006. Medical device software -- Software life cycle processes.
[4] IEC 60601-1 Medical Electrical Equipment Package, 2009.
[5] IEC 61508-3 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems.
[6] ISO 14971:2007. Medical devices -- Application of risk management to medical devices. ISO13485:2003: Medical devices -- Quality management systems -- Requirements for regulatory purposes.
[7] Model-based test generation - http://www.cis.upenn.edu/~rtg/testgen/
[8] Paul E. Ammann, Paul E. Black, and William Majurski, Using Model Checking to Generate Tests from Specifications, Proceedings of ICFEM'98, Brisbane, Australia (December 1998)
[9] B. Legeard et al : BZ-Testing-Tools: A Tool-Set for Test Generation from Z and B using Constraint Logic Programming, In proc. of FATES'02, Formal Approaches to Testing of Software, 2002
[10] UML 2.x component diagram (an Agile introduction) - http://agilemodeling.com/artifacts/componentDiagram.htm
[11] Medical device software standard IEC 62304 et al: http://www.chemgineering.com/en/Scientific%20Articles/$/Medical-device-software-standard-IEC-62304-et-al./22