A Framework for Interlacing Test and Design

Marc Horstmann[1], Eckehard Schnieder[1], Patrick Mäder[2], Severine Nienaber[2], Hans-Martin Schulz[2]
[1]*Institute for Traffic Safety and Automation Engineering, TU Braunschweig*
[2]*EXTESSY AG, Wolfsburg*
m.horstmann@tu-bs.de, e.schnieder@tu-bs.de,
p.maeder@extessy.com, s.nienaber@extessy.com, h.schulz@extessy.com

Abstract

This paper presents a test framework which interlaces the test process with the development process in the domain of embedded control systems. The framework shows the following characteristics:

- *Strong integration of test and development process by solving test management issues inside development tools instead of specialized test management tools*
- *Test in very early development stages by assuming a model-based development process where executable models are available*
- *Reuse of test sequences at different abstraction levels and tool flexibility by tool-coupling*

The paper demonstrates concept and realisation of the test framework, which offers a manual but comfortable test suite derivation where test cases can be archived in the requirements database and interlinked with the requirement specification modules. Once the test sequences are defined, an automatic test instrumentation, execution and evaluation can be done. The test instrumentation is flexible, such that the system under test can be either a model or the realized hardware/software system.

1. Introduction

The development process of embedded systems within the domain of traffic engineering is changing. Due to the complex, distributed and hybrid (continuous/discrete) system character the development process changes from traditional development strategies containing informal textual specification and manual coding techniques to a model-based and tool-supported development process. It starts with requirements engineering and leads to automatic production code generation, which has to be integrated into its designated electronic environment (e.g. a microcontroller) and then into the hardware environment (e.g. a mechanical environment).

Besides constructive quality assurance methods also the analytical part has to be improved. The main method of the later case consists in testing; due to the fact that it can be applied also to very complex systems where other quality assurance methods do not work, especially as soon as hardware becomes part of the system under test.

Concerning the degree of automation in testing three fields can be differentiated:

- Test case derivation
- Test instrumentation, execution and evaluation
- Test process

An automatic model-based test case derivation is still a matter of scientific concern. Nevertheless there exist some case studies that have been reported successful in practice [1], but in general this step is done manually and driven by human intuition.

For test instrumentation, execution and analysis a higher degree of automation can be reached. Nevertheless, a lot of tests are still executed manually. Traditionally, the test process is separated from the development process and starts at the point when the first executable system parts are realized.

In this paper, an approach for a combination of test process integration and automated test instrumentation, execution and analysis automation is presented. The test case derivation is carried out manually, but in a comfortable manner and integrated into the presented framework.

The test framework is developed and used as part of the joint project STEP-X (Structured development process for automotive applications) [5], [13], which is a cooperation between the Volkswagen AG and the Center of transportation at the Technical University of Braunschweig.

2. Concept for Interlacing model-based Development and Test

The test framework that is presented in this paper is characterized by integrating existing concepts and tools instead of developing new custom solutions [11] [4].

Following this strategy, test management issues are solved inside development tools instead of using special test management tools.

To take a maximum advantage of the model-driven development process, executable models of different development stages are used for performing tests. Thus, tests can be executed at very early stages in the development process.

Assuming a black box view on the different development models in combination with constant interface definitions, the test sequences can be reused for different models of different abstraction levels. E.g. for user acceptance tests this is possible due to an early defined user interface that will not change during the development stages. By using tool coupling the test framework enables this possibility even if the tools change between different levels.

2.1 Integration of test and requirements management

Figure 1 shows the role of the requirements management in the STEP-X development process [3] [8]. Besides forming the requirements specification with its different views, e.g. user or system specification, the requirements management connects the specification with models of different development stages. The benefit of this backbone is to get a possibility of forward and backward tracing. E.g. the realization of a certain user requirement at the functional high level design stage can be checked.

The test management is integrated into the requirements management (cf. the right part of Figure 1). By doing so, the tester gains the same advantages, e.g. traceability and consistency, as during the development before. It is possible to navigate from a specific requirement to the corresponding test sequences or test suites and vice versa. Consistency beneficiates even more as test and development simultaneously use the same documents, e.g. concerning system topology and interfaces or system parameters.

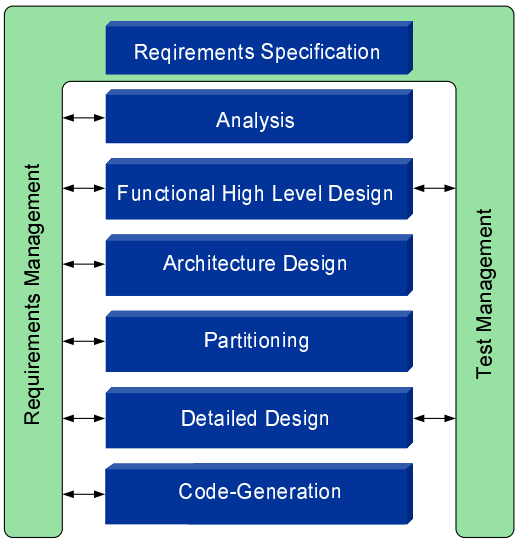

Figure 1: Integration of test management

2.2 Flexible test instrumentation by tool coupling

In traditional software testing following the V-Model, unit, integration, system and acceptance tests are performed sequentially dependent on the level of system development in a bottom-up manner. For example acceptance tests are performed once, by the time where hard- and software parts are complete and integrated.

A model-based development process in combination with tool coupling by co-simulation allows important advantages for the test process. A test instrumentation that is done for a certain case tool can be used for several model combinations. A model combination consists of a control part model which is the aim of the whole development and a plant model that is used to take the environment behavior into account. Figure 2 shows the different combinations, where control part and plant model can be modeled with the same tool as the test instrumentation is done for, but it is also possible that one part or both are modeled with different tools or exist as hardware components.

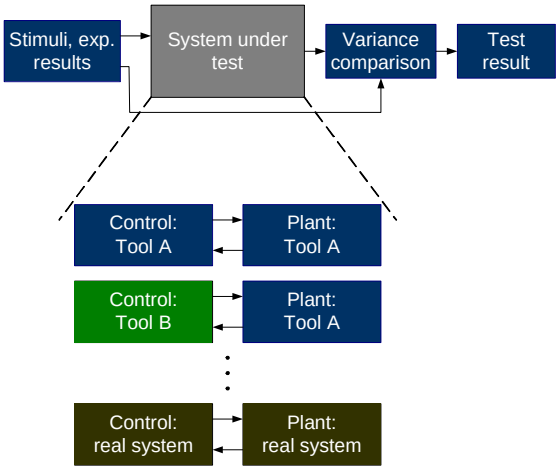

Figure 2: Test instrumentation

In case of constant interfaces for different development levels, e.g. in case that mainly functional abstraction mechanisms are used, it becomes possible to reuse test sequences at different abstraction levels. E.g. for user acceptance tests, where the interfaces are defined early in the development process this becomes possible.

For coupling different CASE tools the EXITE (EXtessy Inter Tool Engineering) toolbox by the EXTESSY AG is used [7]. The client server structure of EXITE allows an integration of several CASE tools into the structure which enables developers and testers to be more or less independent on certain tool manufacturers. The development process as the test process becomes robust against changes in the tool chain.

3. Test Purpose and Test Case Identification

Before specifying or generating test suites, it is necessary to define a test strategy and test purposes. The test strategy specifies the abstraction levels where tests will be executed and the different systems under test, e.g. units or the complete system. The test purpose describes informally what is to be tested.

After this step, the identification and formalization of test cases can be done. The presented test framework uses manual test case derivation since for the kind of systems in the focused domain it will be always necessary to have test cases based on expert knowledge. Of course it would be desirable to have an additional possibility to generate model-based test cases, which will be one subject of future work.

3.1 Test purposes for manual test derivation

Since the test framework realizes the test case derivation manually based on expert knowledge, three different test purposes are identified:

Specification oriented: The idea is to define at least one test sequence for each requirement. This sequence(s) will give two parts of information after execution: Firstly, whether the treated requirement is existing in the implementation and secondly, whether it is working for the test sequence as specified. Of course it can not ensure an absence of errors for this function. As test is defined by Myers [9], the general test purpose is to show failures of a system. In case of performing tests without detecting failures, the tests enhance the confidence to the system.

Risk oriented: Different subsystems or functions can be classified and weighted concerning the risk that results from the subsystems. The test effort will be divided according to the risk oriented classification of functions. In difference to the specification oriented test purpose the risk oriented test purpose is not quantifiable, but it can direct the test resources.

I/o oriented: Another possibility for the test purpose is an i/o-oriented test case derivation, where the state space of combinations is defined as full coverage. As for systems of the focused domain, the reaction depends on the systems history, the i/o oriented coverage is weak. Nevertheless quantifiable information will be given.

An easy way of specification for primary discrete controlled systems is to use classification trees [2]. This notation combines a tree structure of system interfaces and possible states of these interfaces with tables. The test sequences are textual described and then formalized by defining the corresponding interface states. Therefore, the tool CTE XL (Classification-Tree Editor eXtended Logics) [6] is used, which is developed at DaimlerChrysler AG.

3.2 Identification and formalization of test cases

It will be assumed for the following that system requirements are documented as textually described sequences constrained by timing and performance aspects. Additionally it is assumed that as part of these requirements system interfaces and system parameters are defined in separated documents. Within the STEP-X project, DOORS by Telelogic is used for requirement management issues. The specification consists in several DOORS modules. The upper part of Figure 3 shows the important modules for the test definition: Functional descriptions, interfaces and parameters.

The interface module defines possible interface states and has to be transformed into CTE XL, where the test case derivation is done. Thus, this kind of test case

derivation can be done at a very early stage within the development process.

After the test case derivation in CTE XL, the test suite will be imported into DOORS (cf. Block "Test sequence transformation CTE → DOORS" in Figure 3).

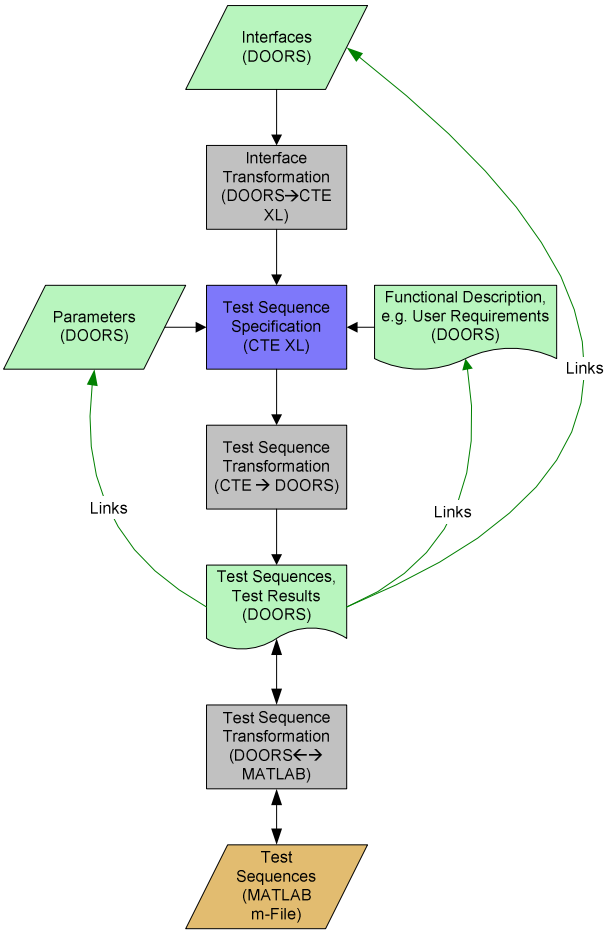

Figure 3: Framework structure for test specification

In DOORS, the test suites can be interlinked with other modules, e.g. the functional specification (cf. Figure 3). Thus, one can have a look to the test status of a requirement by following a certain link and will get information whether the function is tested or not and if the test was successful or not.

If the test sequences are stored and interlinked in DOORS, the test framework offers an automatic instrumentation for different abstraction levels. For all levels, a MATLAB m-file is generated (cf. Block "Test sequence transformation DOORS → MATLAB in Figure 3) since the whole test execution and test evaluation is controlled out of MATLAB/Simulink. Figure 4 shows the corresponding DOORS export window of the test framework. The test suite to be instantiated can be defined by marking single test sequences or a whole test suite.

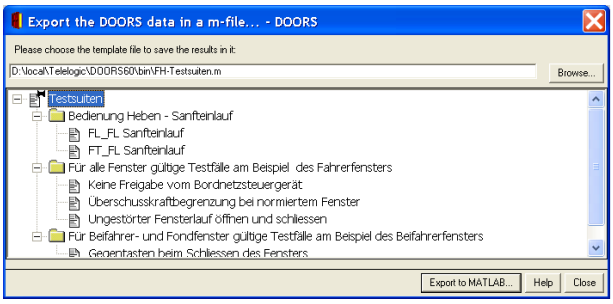

Figure 4: DOORS – MATLAB export window

4. Test Instantiation and Test Execution

An abstraction level where test sequences are instrumented, can either be a model or the realized system with hard- and software components. Figure 5 shows the abstraction levels for tests in the STEP-X process. For testing the realized system with its software, electronic and mechanical components, a test PC is necessary additionally in order to realize the mapping of stimuli in the models to real system stimuli. Therefore, a GENIX Box by add2 is used, where analog, digital and CAN-Interfaces are available. As real time OS, the xPC toolbox by the The Mathworks is used.

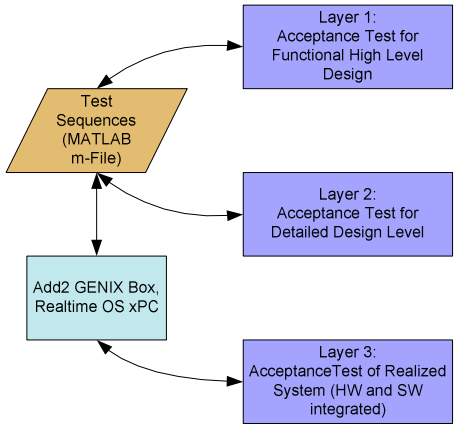

Figure 5: Test implementation

The generated MATLAB m-file can be used for the test execution at different abstraction levels, cf. Figure 2 and Figure 5. For layer 3 in Figure 5, a hardware demonstrator has been realized. The example of use is an electronic window lift control that is part of the STEP-X project. Figure 6 shows this configuration.

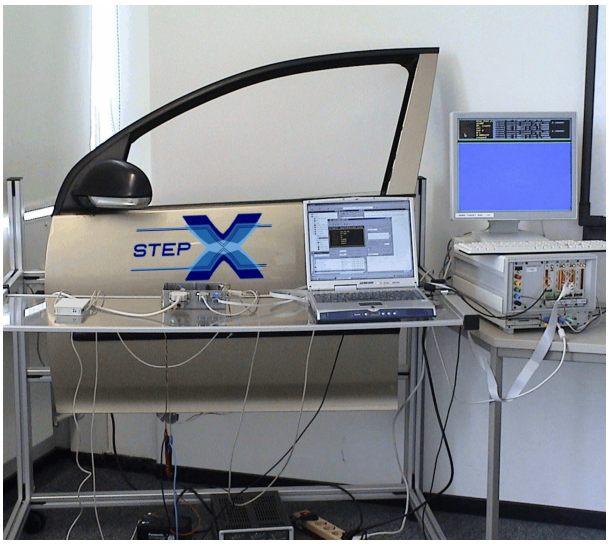

Figure 6: Electronic window lift demonstrator

5. Test Evaluation

Aim of the test evaluation is to detect weather the measured output signal matches expectations or not. Thus, for real systems as focused by the test framework, differences between these two signals will occur. These differences can be due to tolerances in timing or state behavior.

Therefore it is necessary to define these tolerances for timing and states in the requirement specification. The survey of remaining within the specified tolerances has to be automated since otherwise the effort for the test evaluation would become the bottleneck of the whole test framework. Besides, the results would depend on the person who does the evaluation.

A suitable automatic test evaluation should offer to define these tolerances for timing and amplitude of the output signal based on the definitions in the requirement specification. Tolerable are local or global stretching of values up to the defined tolerance, where e.g. changes in the chronological order are not allowed.

Ritter et al. [10] present an algorithm that becomes part of a two step procedure which is presented by Wiesbrock et al. [12]. This procedure offers local signal stretching by calculating a difference matrix which is used for a re-parameterisation of the measured output signal. It allows shifting each value inside the specified tolerances in order to fit to the expected signal. After that, a standard tolerance tube approach is used to compare the equalized measured output against the expected output.

In STEP-X this procedure is realized in MATLAB. Especially the performance of the procedure is important due to an enormous effort that results from the size of the differential matrix. Therefore an algorithm is used that is developed for solving Job-Shop-Scheduling problems, which has shown a very good performance.

Inputs for the algorithm are a matrix of measured and expected values. Additionally the DOORS ID of the test case is part of the matrix in order to allow the import of the test results after test execution and evaluation.

After the evaluation is finished, the results are saved in a file that can be imported in DOORS in order to set or update the status of the test case or test suite.

6. Conclusion

This paper presents a test framework, where the test process is advanced by a strong interlacing of test process and system development. While the test case derivation is done manually, the test instrumentation, execution and evaluation is done automatically.

Synergies for the level of test specification and test management are given by sharing the same databases and establishing link structures with the development process.

The test instrumentation and execution allows testing either models of several tools due to the connection of all components with a co-simulation middleware, or the real system with its hard- and software components. If the interfaces for models of different development stages (e.g. functional high level design and detailed design) are constant, test sequences can be reused for different development stages.

The test evaluation has been automated with a powerful algorithm that allows analyzing a broad range of signals.

7. References

[1] Broy, M.; Jonsson, B.; Katoen, J.; Leuker, M.; Pretschner, A.: Model-based Testing of Reaktive Systems – A Seminar Volume. To appear in LNCS, Springer Verlag, Summer 2004

[2] Grochtmann, M. und Grimm, K.: Classification Trees for Partition Testing. Software Testing, Verification & Reliability, vol. 3, no. 2, pp. 63 – 82, 1993

[3] Harms, M.; Horstmann, M.; Mutz, M. ; Krömke, C.: STEP-X: Strukturierter Entwicklungsprozess für eingebettete Systeme im Automobilbereich.“ Journal „Automotive Electronics“ 1/03, GWV Fachverlage, Wiesbaden 2003

[4] Horstmann, M.: Ein Konzept für die Verflechtung von Entwurf und Test eingebetteter Systeme im Automobilbereich. In: Prof. Dr.-Ing. E. Schnieder, Hrsg.: Tagungsband der EKA 2003, S. 215-230, Juni 2003. EKA 2003.

[5] Horstmann, M.; Harms, M.; Mutz, M.; Bikker, G.: Methodik und Werkzeugkette für einen modellbasierten Entwicklungsprozess im Automobil-bereich. Tagung Automatisierungs- und Assistenzsysteme für Transportmittel, S. 134-153, VDI-Verlag, Düsseldorf, 2003

[6] Lehmann, E.; Wegener, J.: Test Case Design by Means of the CTE XL. Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark, December 2000

[7] König, S; Bikker, G.: Developing and Implementing a Framework for CASE Tool Coupling - Object Orientation upon Tool Level. European Concurrent Engineering Conference 2000, Leicester, 17.-19. April 2000

[8] Mutz, M.; Harms, M.; Horstmann, M.; Huhn, M.; Bikker, G.; Krömke, C.; Lange, K.; Goltz, U.; Schnieder, E.; Varchmin, J.-U.: Ein durchgehender modellbasierter Entwicklungsprozess für elektronische Systeme im Automobil. In: VDI Gesellschaft Fahrzeug und Verkehrstechnik, Hrsg.: Elektronik im Kraftfahrzeug, S. 43-76, September 2003. VDI Tagung Elektronik im Kraftfahrzeug, VDI Verlag GmbH.

[9] Myers, G.: The Art of Software Testing. Wiley and Sons, 1979

[10] Ritter, C.; Willibald, J.; Sax, E.; Müller-Glaser, K. D.: Auswertemöglichkeiten von Systemantworten im Steuergeräteumfeld – ein Überblick. Projektbericht FZI-ESM-DCB-TN-1-1.0, Karlsruhe, März 2001.

[11] Schnieder, E.:Methoden der Automatisierung. Vieweg & Sohn Verlagsgesellschaft, Braunschweig-/Wiesbaden, 1999. ISBN 3-528-06566-4

[12] Wiesbrock, H.; Conrad, M.; Fey, I.; Pohlheim, H.: Ein neues automatisiertes Auswerteverfahren für Regressions- und Back-to-Back-Tests eingebetteter Regelsysteme. Softwaretechnik-Trends, Band 22, Heft 3, 2002.

[13] www.step-x.de
Homepage of the project STEP-X