

ITEA



Information Technology for European Advancement

Task 1.6:

Testing Software-Based Systems - State-of-the-Art - (D 1.6.1)

Version 01 - Public

Edited by Jens Herrmann

Software Development Process for Real-Time Embedded Software Systems (DESS)

ITEA COMPETENCES involved:

- 1) Complex Systems Engineering**
- 2) Communications**
- 3) Distributed Information and services**

September 2000

ITEA

Contents

1	Introduction	3
2	Testing Software-based Systems	4
3	State-of-the-Art Testing at the Dess-Partners	6
4	Evaluation and Outlook	10
	Appendix: State-of-the-Art-Contributions of the DESS-Partners	12

1 Introduction

The goal of DESS is to define an innovative object-oriented, component-based software development methodology for embedded real-time systems, to create supporting environments by integrating state-of-the-art tools and to prove the appropriateness of the methodology by implementing several validation test cases. DESS is the acronym for *Software Development Process for Real-Time Embedded Software Systems*.

One important part of the development method defined in DESS deals with the procedure of **testing** embedded systems. The aim of this particular part of the work (work package 1.6) is to provide guidelines for testing. The basis for the guideline development comprises the current state-of-the-art testing processes of all DESS partners which are documented in this paper.

The state-of-the-art testing processes which are described on the following pages and which has been evaluated according to the aims of the DESS project provide an appropriate basis for developing the mentioned guidelines. It comprises many important aspects relevant to the guidelines, e.g.:

- test management (planning and organization),
- process models for testing, usually consisting of unit, integration, system, and acceptance testing phases,
- methods for test case design,
- testing from the point of view of the user/customer (e.g. on the basis of user profiles),
- sequence testing procedure,
- testing real-time aspects, and
- regression testing.

This report is structured as follows: Chapter 2 deals with testing and explains the aims and procedures used within this field. Chapter 3 summarizes the state-of-the-art software testing of the DESS partners. This will be evaluated with regard to the aims of the DESS project in Chapter 4. This chapter includes an outlook on continuing work on the topic testing within DESS. Please refer to the Appendix for a detailed descriptions on the testing technologies of the DESS partners.

2 Testing Software-based Systems

Generally speaking, testing is concerned with discovering whether certain properties (qualities) exist. This is achieved by determining and evaluating the actually existing properties. Testing in the case of software-based systems means, for example, to test the user-friendliness, functional correctness, safety, temporal or memory behavior of the system. In this context we use the following definition of the test aim:

The aim of testing is to gain confidence that a test object possesses the required properties.

There are more ways than one to reach this goal. For instance, the test can attempt to prove the consistency (correctness) of the test object and its requirements. Since testing is of an experimental nature this can only happen in certain cases. On the other hand, the test objective can also be error detection. Theoretical deliberation and practical experience have shown that the indirect testing approach, i.e. error detection, is a very effective approach to quality assurance.

Testing the correct dynamical functionality of a system is one of the most important tests. A very common procedure is the following: first of all test cases are designed. Then test data are generated on the basis of the test cases with which the test object is afterwards put into execution. The obtained results are then compared to the expected results:

- test case design,
- test data generation,
- expected values determination,
- test execution, and
- test evaluation.

Testing should always be oriented towards customer satisfaction. The customer demands that the system functions correctly according to his requirements, in this way decisively influencing what is meant by quality in this context. The requirements defined by the customer thus determine the testing procedure.

A very effective, indirect approach to test case design which combines both testing for functional correctness and error detection is the following: Test cases are designed on the basis of the requirements with the aim to detect errors. An error occurs if the behavior of the test object deviates from the requirements. If the procedure detects an error we have the basis for quality improvement. If no errors are detected the correctness of the specific test case is proven. This testing approach is called *falsification of requirements*.

Complex systems usually require complex tests, i.e. testing is concerned with a huge number of tests. The most important task when testing large systems is to cope with the complexity of the test.

Testing large systems usually requires a different procedure than testing smaller systems. One possibility to cope with the complexity of the test is to split it into several manageable smaller tests. It is also important to determine on a strategy which ensures that the sum of all single tests will result in a thorough test of the overall system.

ITEA CONFIDENTIAL

A general testing approach of complex systems oriented towards their internal architecture starts with testing the system units or modules. These are first of all tested in isolation. After completing the module tests, the modules are integrated into a higher level for further testing. In this higher level only those test aspects which have so far not been taken into consideration are applied, e.g. interfaces between modules. The stepwise composition of the tests where the system environment (e.g. hardware) is more and more included leads to larger and larger tested system parts, until the system has been completely integrated and tested.

3 State-of-the-Art Testing at the Dess-Partners

The following section summarizes the testing experiences of the DESS partners. For further details please see Appendix.

The contribution by Partner1 to testing software-based systems consists of four parts. These include the following subjects:

- test management,
- test case design,
- sequence testing, and
- real-time system testing.

Tool support has been developed for the first three topics.

The named testing technologies have been developed by Partner1 Research in cooperation with the Partner1 business sectors. They have already been successfully applied within and in part outside the company.

The mentioned technologies are a selection of the testing technologies developed by Partner1 which are relevant to the DESS project. There are further technologies which are partly still under development. It has yet to be examined as to what extent these are of interest to the DESS project and consequently to the test of object-oriented, component-based embedded real-time systems. These technologies are briefly mentioned in the Partner1 contribution.

The testing technologies are based on a process model which consists of the following phases:

- test case design,
- test data selection,
- expected results prediction,
- test execution, including monitoring, and
- test evaluation.

The *test case design* specifies those test cases with which the test object is to be tested. Test cases describe input situations and include a set of test data each. During the *test data selection* phase concrete data based on the test cases are selected. After the expected results have been specified the test will be executed. During *test execution* the behavior of the test object is examined by means of *monitoring*. In the final phase, the *test evaluation*, the results of the test run are evaluated, i.e. the obtained results are compared with the expected ones.

Details on the technologies presented by Partner1:

- **Integrated Test Management within the Tool Environment TESSY:** The testing tool TESSY supports the above mentioned test phases (from test case design to test evaluation) with a main focus on unit testing. Units in this case are C-functions. The support for integration testing, i.e. the next testing stage based on unit testing,

ITEA CONFIDENTIAL

is under development. Further testing activities supported by TESSY are, for example, regression testing and run time determination for test objects. TESSY is available on UNIX and Windows-NT.

- **Test Case Design based on the Classification-Tree Method and the CTE:** The Classification-Tree Method developed by Partner1 supports systematic test case design. The input domain of a test object here is partitioned into disjoint, test-relevant classes by means of classifications. The Classification-Tree Method also supports testing with test case sequences and the test of real-time aspects. For the support of this method Partner1 developed a graphic notation in the form of a tree representing the partitioning, and a combination table which describes the test cases. Tree and table are graphically combined, the tree forming the head of the table. The method is supported by the Classification-Tree Editor CTE. The CTE can be used either in combination with TESSY or stand-alone.
- **Sequence Test Support by MTest:** MTest, developed by Partner1, is a tool which supports sequence testing of models of embedded systems. Models are (executable) abstractions of the system under development which occur very early in the development phase, for instance state automata and Matlab/Simulink diagrams. MTest currently supports only Simulink models, however, support for other model types is planned. The first step when testing with MTest is to describe testing scenarios in the form of test sequences. Typical examples from the field of automotive technology are driving maneuvers and track routing. The sequences are described with the help of an extension of the above mentioned Classification-Tree Method. Tool support for describing sequences is offered by the CTE/ES (CTE for embedded systems). After test sequences have been specified concrete test data are defined with which the model can be executed. Subsequent to the automatic generation of the testing context, the test will be executed and evaluated.
- **Testing Real-Time Systems using Evolutionary Algorithms:** Real-time systems are systems with temporal requirements. An error in the system means that these requirements have been violated by system execution. The test searches for such inputs which lead to longer or shorter execution times of the real-time system than required. If such inputs are found an error has been detected. In order to find test data violating the temporal requirements an evolutionary algorithm is used. This is an iterative process which repeatedly looks for the best test data with regard to error detection. The process starts with a set of data and determines their fitness with regard to the execution time. Best-suited test data are those which are able to violate the temporal requirements. Those data which lead to extremely short or long execution times are of great interest. After having determined the suitability of this data, new test data are generated on the basis of these which will consequently be of greater suitability for error detection. Afterwards the suitability of these data is determined etc. This iterative procedure ends after a temporal error has been detected, or after a previously fixed number of iterations is completed.

The description of the state-of-the-art technology in the field of testing at Partner2 is divided into two parts. One part describes the testing process at Partner2, the second part an applied tool support.

ITEA CONFIDENTIAL

The testing process is part of the software development process SEPA used at Partner2. SEPA stands for *Software Engineering Process Architecture*. It is an integrated process applied for all software products within Partner2 and adapted to special conditions of specific projects. It can be expected that SEPA will be changed in parts due to its future orientation towards embedded systems.

The test model comprises the following phases: component, integration, system/acceptance, and regression testing. The single phases are described by

- inputs and outputs of the single process phases,
- documents used in these test phases, and
- involved persons or teams and responsibilities.

Component testing tests single components or functions. Examples of such components are given in the paper provided by Partner2. Integration testing, also called subsystem testing, consists of testing a group of components. Specific properties are tested, for example conformity, robustness, and performance. The system and also the acceptance test examines the overall system on user/customer platforms. The basis for these tests are *usage scenarios* and *user tasks descriptions*. Regression testing deals with the test after changes have been made to a system.

A short note on process management can be found towards the end of the test process paper by Partner2. To support the management, test plans are created which represent parts of the project plans. Metrics are used to manage test activities. Furthermore, a description on how the general testing process is adapted to specific project requirements is available.

The second contribution on the state-of-the-art technology in testing at Partner2 describes a tool support for functional and stress testing of embedded systems. The tool is called *Validation Test Bank (VTB)* and offers a number of functions for the automatic support of various testing activities, as well as the management of information gathered during testing.

The following testing activities receive automatic support:

- Test data definition.
- Test execution including test run monitoring. Single tests as well as sequences can be executed.
- Test evaluation including analysis of the temporal behavior of the executed tests.
- Test documentation.

The VTB supports two test approaches: testing from the point of view of the user, and testing from a system internal perspective. These two strategies complement each other well and offer a powerful overall test. The tests from the point of view of the user have obtained a high degree of automation. User activities can be simulated and the test object provided with these activities.

VTB can be applied in two different modes:

- **Debug and manual run:** in this mode the user interface is simulated. The user can

ITEA CONFIDENTIAL

enter single commands for the test object. Test results can be subsequently recorded and automatically compared with expected results specifications.

- **Automatic run:** test sequences can be batch-processed. The tester chooses a file containing the tests and determines the test duration with a *cycle number* or an *expiration date*. Test results are recorded in the form of a history file and can be evaluated afterwards.

The paper by Partner2 also describes further aspects of interest, e.g.:

- The file index structure managed by the VTB,
- requirements for files containing test-relevant information,
- the VTB backup mechanism, and
- general means to manage VTB, e.g. access rights (password allocation), adding new files, and handling of new VTB versions.

Partner3 has a process model for developing software-based systems consisting of four phases: *requirements definition*, *architectural design*, *detailed design* and *coding*. The results of these phases (e.g. documents, code) are tested. White-Box-unit tests are put to execution by the developed projects. Black-Box-unit and integration tests are performed by an independent team. System and acceptance tests will also be performed by a separate team in the future. At present they are still carried out by the developed projects.

The aim of the test is to prove that test objects behave correctly in a real environment according to the previously defined test cases. A *Software Verification and Validation Plan* is created as test plan. This plan includes test case specifications. Tool support for test execution is provided by a tool called *Target Test Environment*. The test activities method is called TMap.

Partner3 considers the following points as challenges:

- Expanding the types of objects that can be tested. Mainly test objects in C are tested at present. The test of further applications, including program paradigms such as Windows and object-orientation, are planned.
- Test management is to take place on different kinds of platforms.
- Improving the maturity of the test process.
- Applying metrics.
- Strengthening the importance of White-Box testing within testing activities.

4 Evaluation and Outlook

The following section deals with the evaluation of the testing techniques illustrated in the preceding chapters. The evaluation is orientated towards the aims of the DESS project. The goal, as mentioned above, is to provide guidelines for the test of object-oriented, component-based, embedded real-time systems. First of all we will evaluate the contributions made by the DESS partners separately, and then conclude with an overall assessment.

The contribution by Partner1 describes a process model and four specifically developed testing technologies.

- TESSY, a tool supporting all important phases of unit testing, which will in future also support integration testing.
- The Classification-Tree Method for test case design including tool support.
- MTest, a tool support for sequence testing of models.
- A method for testing real-time systems based on the use of evolutionary algorithms.

Further testing technologies that have been developed by Partner1 are also briefly mentioned in the paper provided by Partner1.

With regard to the testing guidelines developed within DESS it has to be resolved to what extent concepts which support TESSY can be integrated into the testing process for component testing. The units tested with TESSY are C-functions. The components considered by DESS are of a more complex nature than C-functions. It is likely that some concepts from TESSY can be used for the test of components. However, these would have to be extended by further technologies in order to be able to test component properties which go beyond C-functions.

The Classification-Tree Method developed by Partner1 is definitely a valuable contribution to the testing guidelines because the test case design generally plays an important role in testing. This will also be the case for the guidelines. The method also supports sequence testing and the testing of real-time aspects. Especially the latter is an important topic within DESS.

MTest might possibly be of interest for DESS. Should sequence testing establish itself as a part of the guidelines it would be possible that concepts from the MTest will be used. The test of models, which is supported by MTest, is most probably, and according to the current status of development, of little interest for DESS.

Of stronger interest, on the other hand, is the Partner1 method for testing real-time aspects because this property plays an important role within the DESS project. This method, which detects run time errors, must definitely be evaluated for the guidelines.

A note on the contribution by Partner2: the test process during software development corresponds to the commonly applied testing procedure: the testing process is split into smaller, more manageable tests, i.e. component, integration, system, and acceptance tests, facilitating the overall testing procedure. A positive aspect is the fact that Partner2 has mastered the testing process to a certain level, because a detailed description of which is available. Furthermore it has to be mentioned that the component concept is

ITEA CONFIDENTIAL

already part of the process and that the testing process will be adapted to the needs of embedded systems in the future. The aspects process model for testing, component testing, and embedded systems are important features within DESS.

The tool *Validation Test Bank* (VTB), used by Partner2, offers support for activities during the test of embedded systems, from test data definition to test evaluation including test documentation. It is desirable, however, to extend the tool support to the important phase of test case design (perhaps, if suitable, by applying the test case design method introduced by Partner1). VTB supports a combined testing procedure from an external and an internal user perspective. The procedure is also called combined Black-Box and White-Box testing and offers a powerful approach to testing. Another advantage of VTB is that it supports the execution of single tests, as well as test sequences, and the analysis of the temporal behavior. These aspects, i.e. the systematic approach to testing, embedded systems, combined Black-Box and White-Box testing, analysis of temporal behavior, are all relevant topics for the guidelines created by the DESS project.

Partner3' s testing process also offers a systematic approach to software development. Different tests are applied to the developed partial products: unit, integration, system, and acceptance tests are executed. It is planned for the future to introduce an independent internal testing authority that will carry out the major part of the testing processes. This will prove to be an effective approach to the testing process. Already now parts of the testing process are carried out by an independent testing authority.

Plans are created for the test, and tool support is offered for test execution. An orientation for the testing work is offered by TMap: a systematic process model for testing. Partner3 regards the improvement of the maturity of the existing test process as an important challenge. On the whole Partner3 follows a very systematic, tool-supported test process with the aim to improve the process. The pursued systematics is also of interest to DESS. However, a prior adaptation to the distinctive features of the DESS project needs to be done (components, object-orientation, real-time, embedded systems). A very interesting aspect for DESS is the concept of a (continuing) improvement of the testing process.

The current state of testing processes of the DESS partners offers a good and solid platform for the development of guidelines for the test of object-oriented, component-based embedded real-time systems, and thus also for further work in the field of testing within the DESS project. What is still missing is a stronger focus on the systems handled within DESS, i.e. a stronger adaptation to existing testing technologies on the system properties object-orientation, components, real-time, and embedding is necessary. Furthermore, it is likely that new technologies will have to be developed, which are not covered by the experiences of the DESS partners, but which are necessary to test this special class of systems. One example is the testing process of components which takes into consideration the special properties of components and thus the component definition developed within DESS.

ITEA CONFIDENTIAL

Appendix: State-of-the-Art-Contributions of the DESS-Partners

Partner1: *Testing software-based Systems - Partner1 Contribution*. Internal DESS Paper, March 2000

Partner2 (1): *Testing software-based Systems - Partner2 Contribution: Test Process*. Internal DESS Paper, June 2000

Partner2 (2): *Testing software-based Systems - Partner2 Contribution: Validation Test Bank*. Internal DESS Paper, June 2000

Partner3: *Testing software-based Systems - Partner3 Contribution*. Internal DESS Paper, March 2000