Automatic Linking of Test Cases and Requirements

Thomas Noack
Berlin Institute of Technology
Daimler Center for Automotive IT Innovations (DCAITI)
Berlin, Germany
E-Mail: thomas.noack@dcaiti.com

***Abstract*—The paper proposes a 3-layered method which automatically creates trace links between test cases and reused requirements (test-links). While the first layer automates the manual test-link reuse, subsequent layers apply elaborate filter mechanisms. More specifically, Case-Based Reasoning is used in the third layer for detecting scenarios where test-link reuse is questionable. The proposed 3-layered method is explained with the help of a clarifying example.**

Keywords-Reuse; Requirements; Test cases; IBM DOORS

I. INTRODUCTION

Daimler uses the V-Model to manage methods and tools which guide the development process. Each vehicle series project passes the V-Model from the requirements stages over implementation to the test stages. The vertical integration defines which stages are performed by internal engineers and which stages are performed by external suppliers. Due to the low vertical integration in the automotive domain many engineers nowadays work mainly with engineering artifacts which are located in the upper stages of the V-Model. These artifacts are system requirements, test cases and trace links between them. The actual implementation is often done by suppliers.

Each new vehicle series inherits engineering artifacts from previously completed vehicle series projects. That means, reuse takes place from a source to a destination. The observation of the Daimler development process revealed several interesting facts. The reuse direction is horizontal from a source V-Model instance to a destination V-Model instance. Reusing system requirements is done by copying and adapting the requirements specification. Interestingly, test case reuse is not done via copying in practice. Instead, it is done by setting thousands of test-links from the existing test cases to the copied and adapted system requirements. This paper introduces a method to automate the complex task of setting the test-links between test cases and reused system requirements.

The paper is structured as follows: Firstly the Daimler specific DOORS®[1] modules and their interaction are introduced. After that the 3-layered method to reuse test-links is presented. The paper continues with a minimal example and related work. In the conclusion section, comments are made and future work is drawn.

II. DOORS® MODULES IN THE UPPER V-MODEL

Daimler uses DOORS® for requirements and test engineering. DOORS® manages specification documents in so called modules. Figure 1 shows the modules in the upper V-Model stages of the Daimler development process. The proposed method focusses on the relationship of the three following modules.

A. System Requirements Specification (SRS)

A vehicle is described by many SRS - one SRS for one system. Examples for systems are *Wiper Control* or *Outside Light Control*. The main engineering artifacts in SRS are vehicle functions. Examples for vehicle functions are *wash windshield* or *activate turn-signal right*. Each vehicle function is refined by specific (non-)functional requirements.

B. Test Specification (TS)

Test cases are the engineering artifacts in TS. A test case is characterized by test actions, pre- and pass conditions, assignments to test levels, test goals and many other properties. Test cases link to the corresponding requirements they verify. Each SRS has at least one corresponding TS.

C. Test Concept (TC)

The TC contains the test plan which defines, what must be tested when for which purpose. The test plan artifacts are: test object type (What?, e.g., *vehicle function*), test level (When?, e.g., *vehicle integration test*) and test goal (Which purpose?, e.g., *correct interaction on interfaces*). The TC defines which test-links must exist between TS and SRS in order to fulfill the test plan. Therefore, each test case in the TS is classified according to the test plan artifacts.

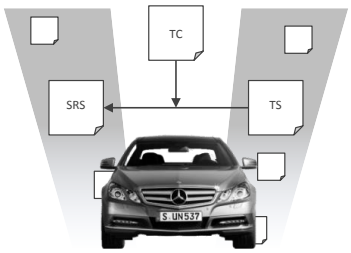

Figure 1. DOORS® modules in the upper V-Model

 ISBN: 978-1-61208-307-0

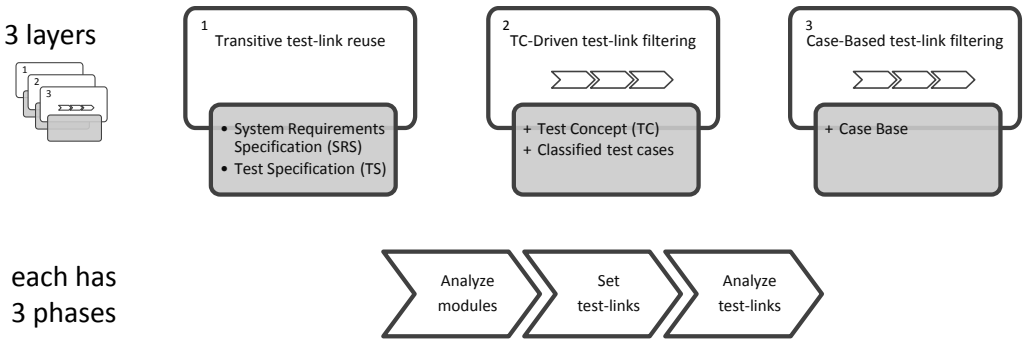

Figure 2. 3-layered method to reuse test-links

III. 3-LAYERED METHOD TO REUSE TEST-LINKS

This section describes the proposed 3-layered method for automating the reuse of test-links between TS and SRS. Figure 2 depicts the layers of the method.

Each layer consists of the same three phases. The specific tasks of each phase differ depending on the layers characteristics as indicated in Figure 3. A subsequenting layer enhances the phases of its predecessor with additional tasks. The general tasks performed in the three phases are as follows.

- *Analyze modules*: Extract information from the SRS_{Src} (Source), SRS_{Dst} (Destination), TS and TC.
- *Set test-links*: Set links from TS to SRS_{Dst} on the basis of the above analysis results.
- *Analyze test-links*: Assess the links and highlight the link status in SRS_{Dst} and in TS.

The first layer can be directly integrated into the Daimler development process because the process stipulates the existence of the involved modules SRS_{Src}, SRS_{Dst} and TS.

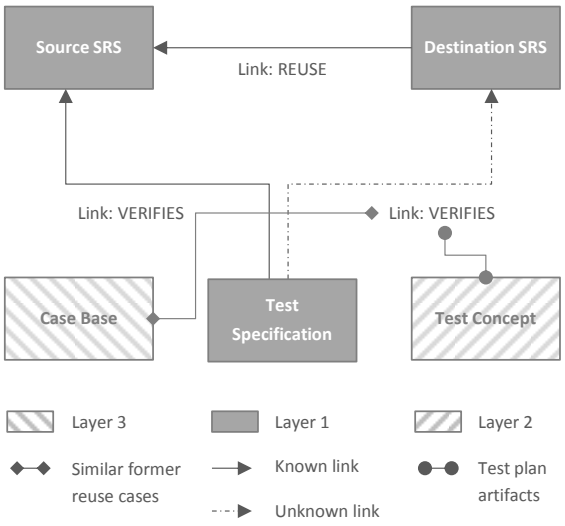

Figure 3. DOORS® modules needed by the layers

In Figure 3, the filled boxes and link arrows show the minimal reuse situation presumed by the first layer. When a new vehicle series project is launched, requirements are reused by copying the complete SRS_{Src} module. The resulting SRS_{Dst} is then adapted to the requirements of the new vehicle series. While the test-links from TS to SRS_{Src} do exist, the test-links from TS to SRS_{Dst} do not exist. The first method layer automatically sets the not existing test-links and highlights the link status in SRS_{Dst}.

The TC shown in the lower right corner of Figure 3 is the additional module needed by the second layer. The TC defines which test-links must exist in order to fulfill the test plan. The connection between TC and TS is established by classifying the test cases within TS according to the test plan artifacts of TC. By the virtue of taking test plan artifacts into account, the resulting test-links and highlighted requirements are much more comprehensive compared to the linking and highlighting of the first layer.

The Case Base shown in the lower left corner of Figure 3 is the additional module needed by the third layer. Case-Based Reasoning relies on two assumptions [2]: (1) similar problems have similar solutions and (2) similar problems occur continuously. Transfered to test reuse, these assumptions mean that (1) similar reuse situations result in similar reuse decisions and (2) similar reuse situations occur continuously. The cases of a Case Base are structural representations of previously applied knowledge [3]. Reuse knowledge is represented by differences between previous SRS_{Src}, SRS_{Dst}, TS and TC. The benefit of Case-Based filtering is that typical situations, which disable test-link reuse, can be recognized automatically.

Discussions with Daimler engineers led to an interesting conclusion. Case-Based Reasoning can, in the given context, only be used to detect situations, where test-link reuse is not possible. If, for example, the interface specification of a destination requirement changed, it can be assumed that a integration test case probably must be reviewed. On the other hand, it can not be automatically assumed that a test-link can be reused only because the interfaces did not change.

 ISBN: 978-1-61208-307-0

The layers and its phases are described as follows.

A. First layer: Transitive test-link reuse

1) Analyze modules: If a requirement is reused in SRS_{Dst}, it has a reuse-link to the corresponding requirement in SRS_{Src}. The textual similarity between each source and destination requirement is calculated and stored in SRS_{Dst}. New and heavily adapted requirements of SRS_{Dst} have no reuse-links to SRS_{Src}.

2) Set test-links: Test-links are reused transitively. That means, if a test case in TS verifies a requirement in SRS_{Src} and if this requirement has been reused by a requirement in SRS_{Dst}, then the test case in TS also verifies the requirement in SRS_{Dst}. If the destination and source requirement are textually identical, the test-link is reused. Otherwise, it must be reviewed by the test engineer in the next phase.

3) Analyze test-links: After the test-links have been set in the previous phase, SRS_{Dst} is analyzed. Three scenarios can occur for each destination requirement: (a) it is identical to the source requirement and hence a test-link can be reused directly (b) it has been changed slightly and, therefore, the test-link must be approved by the engineer (c) it has no test-link because either it has been changed heavily or it is a new requirement or the source requirement has no test-links.

B. Second layer: Test-Concept-Driven test-link filtering

1) Analyze modules: Based on the analysis of the TC, a 3-dimensional test plan cube is constructed. The cube dimensions are the test plan artifacts: test object type, test level and test goal. An example of a cube cell would be the triple (vehicle function, integration test, verify interface).

2) Set test-links: This phase enhances the first layer with a filtering mechanism enabled by the cube. In particular, the necessity of the test case is examined by passing the test case classification to the cube. Only if a test case is considered as needed by the cube, i.e., as needed to verify a test goal for a test object type on a test level, the test-link is set.

3) Analyze test-links: While the first layer can only make statements about the pure existence of test-links the second layer also considers test plan artifacts. Therefore, for each destination requirement the following more detailed scenarios arise: (a) a test case for a specific test object type is missing (b) a test case for a specific test goal is missing and (c) a test case for a specific test level is missing.

C. Third layer: Case-Based test-link filtering

1) Analyze modules: In this most sophisticated layer, Case-Based Reasoning (CBR) is utilized to filter test-links based on previous reuse experience. More specifically, a current reuse situation is constructed for each potentially reusable test-link by extracting relevant information from SRS_{Src}, SRS_{Dst}, TS and TC. The situations are then converted into structural case representations to enable similarity search in the next phase.

2) Set test-links: For each constructed current reuse situation, a similar case in the Case Base is searched. Therefore, similarity measures, as shown in [4], are applied. If a similar negative reuse case, i.e. where link reuse has been questionable, is found for a current test-link, its reuse possibility is also marked as questionable.

3) Analyze test-links: The third layer extends the previous layers with additional analysis possibilities with respect to not reusable test-links. For each classifying property more fine-grained scenarios arise, e.g., (a) interface has been changed thus an integration test case is questionable or (b) safety relevance has been changed thus a safety test case is questionable.

IV. Example

Figure 4 depicts an example to show the application of all three layers. The following subsections describe, how the 3-layered method extends the current module landscape.

A. Current state of the modules

Currently, SRS_{Dst}, SRS_{Src} and TS are stipulated by the Daimler development process. While the TC has been rolled out lately, the Case Base is a new module which only exists conceptually. The SRS modules in Figure 4 contain the textual requirements Src_X and Dst_X and columns of their properties.The TS contains test cases which trace link to SRS modules.

B. Reuse relationship between SRS_{Src} and SRS_{Dst}

Src_{1+2} in SRS_{Src} and Dst_{1+2} in SRS_{Dst} are in a reuse relationship. Since Dst_1 has not been changed textually, it is 100% similar to Src_1. Dst_2 has been modified in order to adapt the changed needs of a new vehicle series. The textual similarity of Dst_2 and Src_2 is 80%. For further considerations we assume that 80% is within the borders of the reuse threshold. Dst_3 has changed heavily and the reuse relationship to Src_3 could not be detected technically. Dst_4 is a new requirement.

C. Transitive reuse of test-links

The test cases Test_X in TS have test-links to the requirements Src_X in SRS_{Src}. These test-links between Src_{1+2} and Test_{1+2} are reused to link to the corresponding requirements Dst_{1+2} in SRS_{Dst}. While the test-link between Test_1 and Dst_1 has been reused directly, the test-link between Test_2 and Dst_2 must be reviewed because the requirements Dst_2 and Src_2 are not identical. The test-link between Dst_3 and Test_3 is not set because Dst_3 and Src_3 have no reuse-link.

D. TC and classified test cases in TS

The TC stipulates that each test object of the type *vehicle function* must be verified on *integration* (Int) and *system* (Sys) test level. The test goal *functionality* must be verified by both, *integration* and *system* test. *Correct interaction on interfaces* has to be verified on the *integration* test level

 ISBN: 978-1-61208-307-0

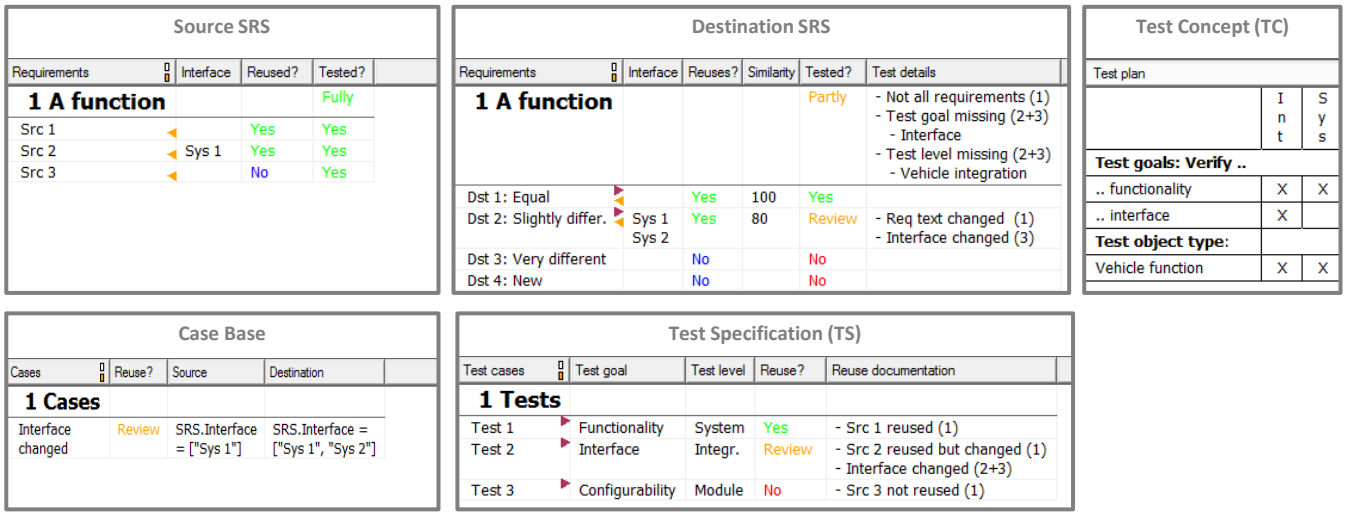

Figure 4. Minimal example (DOORS®module state after running layer 3)

only. The test cases in the TS are classified by the test plan artifacts *test goal* and *test level*. $Test_3$ verifies the test goal *configurability*. Because this test goal is not considered by TC, a potential test-link from $Test_3$ would not be set.

E. Case Base

The requirements Src_2 and Dst_2 have a common property: *Interface*. While Src_2 has an interface to the system ['Sys 1'], the system dependencies of Dst_2 are ['Sys 1', 'Sys 2']. This current reuse situation is transformed to a case.

The search in the Case Base returns the case *Interface changed*, which is identical to the current reuse situation. The reuse decision *Review* of the case is applied to the current situation between $Test_2$ and Dst_2.

V. State of the Art

There are two possible sources from which reused test cases can originate: they come from already existent test cases or have been generated from reused test models. Because this work is located in the upper V-Model stages it clearly focusses on the first possible source.

Three works mainly inspired the 3-layered method. Geppert et al. describe, how textual test cases can be transformed to product line test cases [5]. Nebut et al. propose a requirement-based approach for testing product families [6]. They generate textual test cases from the so called Use Case Transition System, which is formed by trace links between Use Cases. Minor and Hanft use Case-Based Reasoning for reusing test cases by analyzing textual similarity [7].

VI. Conclusion and Future Work

This *Work in Progress* paper proposed the basic functionality of a 3-layered method which has been developed for supporting the industrial test case reuse process pragmatically. The first method layer has been piloted successfully in the automotive domain with real specification modules from the Wiper Control System and the Rain Closing System. Implementation details and field study results of each layer follow in future publications.

The proposed method does not only exclusively support the automotive domain. It is located in the upper V-Model stages and thus can be applied generally to each environment, where test cases and system requirements are connected by test-links.

References

[1] IBM, "Rational DOORS," http://www-03.ibm.com/software/products/us/en/ratidoor/ [Last access: 19/06/2013].

[2] D. B. Leake, "CBR in Context : The Present and Future," in *Case-Based Reasing: Expericences, Lessons and Future Directions*. MIT Press, 1996, ch. 1, pp. 1–35.

[3] R. Bergmann, J. Kolodner, and E. Plaza, "Representation in Case-Based Reasoning," *The Knowledge Engineering Review*, vol. 20, pp. 209–213, 2006.

[4] F. Brosius, "Distanz- und Ähnlichkeitsmaße (engl.: Distance and Similarity Measures)," in *SPSS 21*. mitp, 2013, ch. 31, pp. 693–709.

[5] B. Geppert, J. Li, F. Rössler, and D. M. Weiss, "Towards Generating Acceptance Tests for Product Lines," in *Proceedings of the 8th International Conference on Software Reuse*, Madrid, Spain, 2004, pp. 35–48.

[6] C. Nebut, F. Fleurey, Y. L. Traon, and J.-M. Jézéquel, "A Requirement-based Approach to Test Product Families," in *Proceedings of the 5th International Workshop on Software Product-Family Engineering*, Siena, Italy, 2003, pp. 198–210.

[7] M. Minor and A. Hanft, "The life cycle of test cases in a CBR system," *Advances in Case-Based Reasoning*, pp. 455–466, 2000.

 ISBN: 978-1-61208-307-0