Chapter 5

Requirements Traceability within Model-Based Testing: Applying Path Fragments and Temporal Logic

Vanessa Grosch
University of Ulm, Germany

ABSTRACT

Requirements traceability enables the linkage between all development artifacts during the development process. Within model-based testing, requirements traceability links the original requirements with test model elements and generated test cases. Current approaches are either not practical or lack the necessary formal foundation for generating requirements-based test cases using model-checking techniques involving the requirements trace. This paper describes a practical and formal approach to ensure requirements traceability. The descriptions of the requirements are defined on path fragments of timed automata or timed state charts. The graphical representation of these paths is called a computation sequence chart (CSC). CSCs are automatically transformed into temporal logic formulae. A model-checking algorithm considers these formulae when generating test cases.

INTRODUCTION

In practice, model-based testing still lacks the confidence of the actual users. Usually only structural or similar coverage criteria (Ryser et al., 1998; Gaston & Seifert, 2005) are used to select the tests. This type of test selection is not satisfying, even if the test engineers understand the algorithms behind test case generation. For example, the structure-oriented criteria do not answer the question whether a test case should be run during a new software release or whether it might be left on hold until the next release. Requirements traceability with an adequate depth of information contributes to requirements-based test case generation and could help solve this type of issues. The test cases should be connected to dependant test model fragments and original system.
requirements. When this is done, the context of a test case is then linked to the original requirements as well. Requirements traceability leads to a better understanding of test cases by introducing a powerful requirements description and a more efficient reuse and maintenance of test models. The need to test according to the requirements is demanded by quality and safety norms.

There are several different approaches to requirements traceability. They range from simple state annotation to formal requirements specification. In the former case, the content of a requirement cannot be captured by labeling a single transition; a timed sequence of transition changes is needed.

Three examples give an idea of the problems occurring. (1) A requirement states “The maximum delay between a turn indication request and the turn indicator flashes is 600ms”. In a model containing the whole turn indication functionality, many transitions have to be taken between the turn indication request and the turn indicator flashing. The transitions may have different timing restrictions. Only annotating all the transitions, which is a lot of effort and error-prone task, would reflect the statement of a maximum delay of 600ms. (2) Another requirement states “A turn indication request that got interrupted by an emergency flashing request will continue when withdrawing the emergency request”. This requirement needs the parallel visit of the two states turn indication requested and emergency flashing requested, ensuring that turn indication requested was visited first. Then, the emergency flashing requested state has to be left while the turn indication requested state is still visited. These timing constraints cannot be reflected by just annotating the requirements transitions. (3) The last example addresses the tip indication of turn indicator lights which results in three flashes. The requirement states that “Repeating tip indication requests result in the last three flashes”. As a consequence summing up the flashes is forbidden. To reflect the requirement, at least one repetition of the tip indication request is needed. This results in a cycle and revisit of already visited test model elements. Cycles in a path are usually avoided by test case generators and cycles cannot be described by simple annotation of transitions with the requirements identifier.

However, a completely new notation to achieve a formal and powerful requirements specification that is able to handle the examples above, is not efficient, because the test engineer defining the requirements trace needs to specify the same requirement in two different ways. Some examples of formal graphical requirements specifications will be given in chapter Related Work.

This paper offers an approach to requirements traceability through application of state-based test models in a balanced manner. Balanced means that the approach uses the states of the test model to reduce the effort of composing the requirements trace, it adapts the modeling paradigm of the test model and therefore does not request another interpretation of the requirement and it adds new information of the sequential flow of a requirement as a cluster of paths through the test model. It aims at narrowing the gap between the operational specification of a system designed as state charts and the declarative specification expressed in temporal logic. As result a formal but at the same time easy to create approach is described.

The requirements descriptions, called computation sequence charts, offer enough dynamic range to address most behavioral aspects (e.g., hard real time, distribution, etc.). The creation of these requirements descriptions is quick because they are composed of pre-existing test model elements. As an additional advantage, these same requirements descriptions can be formalized and therefore used in automatic test case generation using model-checking techniques.