Safety-related software development using a model-based testing workflow

Paul Urban
Udo Brockmeyer, PhD
January 08, 2013

Developing safety-related software, where failure can result in injury or loss of life, such as in airplanes, automobiles, trains, or medical devices, requires extra care and effort. The delivery of safe code that is compliant with strict development standards and guidelines such as DO-178C, DO-178B, ISO 26262, IEC 61508, or IEC 62304, can result in increased time and cost of the project. This article describes how to use the Rhapsody Reference workflow included with the IBM Rational Rhapsody Kit for ISO 26262 and IEC 61508 for development of safety-related applications. You will learn about the Rhapsody Reference workflow, and how model-based testing with the Rational Rhapsody TestConductor Add On can be used to verify the model and the generated code. It reduces the time to deliver high-quality software yet still complies with safety standards.

Challenges of safety-related software

Embedded software is increasingly at the heart of today's innovative products. It's key in defining the functionality and controlling the electrical and mechanical systems in the products we rely on every day. In airplanes, automobiles, trains, or medical devices, for example, failure can result in injury or loss of life. Extra care and effort is required to ensure the safe operation of the system for the safety of users and to avoid costly product recalls.

For code where safety is critical, companies must comply with strict development standards and guidelines, such as DO-178C and DO-178B for commercial avionics, ISO 26262 for automobiles, IEC 62304 for medical devices, or IEC 61508 for general functional safety. It is each company's responsibility to produce evidence that they follow good development processes, such as traceability from requirements to implementation, sufficient testing, and that their tools do not introduce errors in the product. The extra time and testing that are necessary to verify that the software complies with safety requirements can significantly increase development time and cost.

Automating testing with model-based testing

With model-based testing, you can be capture test cases graphically. This creates expressive test cases that are easier to understand and communicate across the development team. The test cases can be traced to requirements, and the impact of requirement changes can be understood easily. The IBM® Rational® Rhapsody® TestConductor Add On adds model-based testing
capabilities that are based on the UML Testing Profile to the Rational Rhapsody Developer, Rational Rhapsody Designer for Systems Engineers, or Rational Rhapsody Architect for Software editions (see the Resources section for links to the product overview and Evaluate pages). The testing profile tailors the development environment for testing by adding concepts such as test architectures and test behaviors into UML. Test architectures extend the existing UML 2.0 structural concepts to describe the test elements that are involved and their relationships. Similarly, the test behavior extends the existing UML 2.0 behavioral concepts to encompass all observations and activities during the test.

The Rhapsody TestConductor Add On can automatically create test architectures for the system under test. Tests cases can be graphically created using UML sequence diagrams, state charts, or flowcharts. The graphical representation of the test cases allows for better communication of the tests to aid the understanding of the behavior of the design. The tests can be executed and results monitored to automate unit and regression testing. By testing earlier in the development process, at the design model stage, quality assurance managers and software engineers can efficiently and effectively verify designs with respect to requirements to identify problems sooner.

Benefits of model-based testing in safety-related development
For safety-related software, it is mandatory to have full traceability from requirements to software architecture to code. Additionally, traceability from requirements to test cases that check the correctness of requirements against the developed software is required. Implementing elements such as a test architecture or test cases as model-level concepts allows bidirectional traceability to be realized directly at the model level. This enables automated analysis of both requirements coverage and structural coverage of model and code. Additionally, when test cases are specified graphically by using notations such as UML sequence diagrams or state machines, verification is easier and more efficient than it is for traditional, code-centric test cases. A model-based approach allows the development of both design artifacts and test artifacts in a unified framework. Therefore, it adds agility to the development and test process, and that improves efficiency and lowers cost in comparison to processes with separate development and test phases. IBM Rational Rhapsody TestConductor Add On automates many of the testing activities, including creation of test architectures and execution of test cases. Therefore, testers can focus on the correctness and completeness of their test cases rather than spending time on tedious and error-prone tasks, such as creating test harnesses. Model-based test architectures and test cases are easier to maintain than traditional test scripting languages, because of their graphical nature and explicit documentation.

Overview of the Reference workflow with model-based testing
The Rational Rhapsody Reference Workflow (see Resources) describes an approach for model-based development, including automatic code generation and model-based testing for safety-related software development. Figure 1 shows the major activities in this reference workflow. The upper part of the workflow describes activities to design and implement the safety-related software. The lower part of the workflow describes activities to verify the software.

The approach addresses design and implementation together with appropriate test and verification. Textual requirements guide the development of a formal UML/SysML model, which is
then is translated to code by using code generation. Both refinement steps are accompanied with appropriate guidelines and checks.

The refinement step from textual requirements to a design model ready for code generation is verified by performing systematic requirements-based testing on the model level by leveraging model simulation, using IBM Rational Rhapsody animation. This is also called Model-in-the-Loop (MiL) testing. The generated code can be verified on a host computer by executing the same set of test cases used during MiL, but without including Rational Rhapsody animation. This is also called Software-in-the-Loop (SiL) testing. An automated equivalence check of the test results (back-to-back testing) between MiL and SiL is performed to verify the results. This can be complemented by executing the set of tests on the target processor, also called Processor-in-the-Loop (PiL) testing. Test execution on model and code provides structural coverage measurement to assess the completeness of the tests and avoid including unintended functionality. Requirements coverage is measured during execution of the test cases.

**Figure 1. Activities of the IBM Rational Rhapsody Reference workflow**

The first activity in the workflow is to translate given requirements into an executable model by using appropriate modeling guidelines. Model-based tests are then added to ensure that the model indeed correctly captures the requirements. Coverage metrics (requirements coverage and model coverage) can measure the completeness of the model-based test suite. Code generation is used to generate an implementation from the model. Back-to-back testing, or equivalence testing, between the model and code constitutes the key element for code verification. Running a test suite on both levels verifies that the model and code show the same behavior. Code coverage metrics are used to ensure completeness of the test suite according to the predefined code coverage criteria.

**Example of performing ISO 26262 verification activities using model-based testing**

ISO 26262-6 mentions many recommended or even highly recommended methods to perform the development and testing activities. For example, Table 1 list methods of software unit testing that are recommended by ISO 26262-6. Depending on the Automotive Safety Integrity Levels (ASIL) required for the project for ISO 26262 A to D, with D as the most critical level, you will need to use
some or all of these methods to achieve compliance. These methods include requirements based test, interface test, fault injection test, resource usage test, and back-to-back comparison test.

### Table 1. ISO 26262-6 recommended unit testing methods for Automotive Safety Integrity Levels (ASIL)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Automotive Safety Integrity Levels (ASIL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Requirements-based test</td>
<td>Highly recommended</td>
</tr>
<tr>
<td>Interface test</td>
<td>Highly recommended</td>
</tr>
<tr>
<td>Fault injection test</td>
<td>Recommended</td>
</tr>
<tr>
<td>Resource usage test</td>
<td>Recommended</td>
</tr>
<tr>
<td>Back-to-back test</td>
<td>Recommended</td>
</tr>
</tbody>
</table>

Requirements-based testing is highly recommended for ASIL A to D. In other words, requirements-based testing is mandatory. Requirements based testing is hence an important verification activity in the Rational Rhapsody Reference workflow, too.

**Figure 2. Performing requirements-based testing with the Rational Rhapsody TestConductor Add On**

During requirements-based testing, you need to systematically verify whether the system under test (SUT) behaves as specified in the requirements defined for it. For each requirement one or more test cases must be defined that check the behavior of the SUT. IBM Rational Rhapsody offers four different ways to specify the behavior of test cases: sequence diagrams, statecharts, activity diagrams, and plain code. Depending on the requirement to be checked, one of these formalisms might be more suitable than others. For example, suppose that the SUT is a model of a stopwatch. This is one of the requirements of the stopwatch:

"REQ_INIT: After starting the stopwatch, the stopwatch shall display 0 minutes and 0 seconds (0:0)".

To verify and test this requirement with IBM Rational Rhapsody, you can create a sequence diagram test, as depicted in Figure 3. First, this input is sent to the SUT:
evPressKey(KeyVal=1)

This input means that the stopwatch is started. As the expected reaction, the sequence diagram specifies that the SUT shall emit this event:

evShow(m=0, s=0, b=FALSE)

This means that the stopwatch shall display time as 0:0.

**Figure 3. Defining the behavior of a test case with a sequence diagram**

After the behavior of the test case is defined, the next step is to link the test case to the requirement to be tested by adding a **TestObjective** to the test case that points to the requirement. The test objective explicitly links the test case to the requirement (see Figure 4) for traceability between the requirement and the test case.

**Figure 4. Linking a test case to a requirement with a TestObjective element**
After defining the test case and linking it to a requirement, the next activity is to execute the test case and compute the test result (Figure 5). The test execution report contains information about the test execution, such as the test execution time and the test result.

**Figure 5. Test execution window (bottom left) and test report (right).**

In the same way, test cases for all requirements need to be developed. This process is iterated until all requirements have been covered by appropriate test cases.

ISO 26262-6 also demands determination of test coverage of the requirements to assess whether the requirements have been completely tested. The assessment of which requirements are covered by which test cases, and the reverse, is another activity that is described in the Rhapsody Reference workflow, as depicted in Figure 6.

**Figure 6. Assessment of requirements coverage of test cases**
Rational Rhapsody provides several mechanisms to assess the actual requirements coverage. Figure 7 shows how you can use a **Test Requirements Matrix** to automatically visualize the relationship between requirements and test cases. The requirements are shown on the horizontal axis, and the test cases are shown on the vertical axis. If a test case is linked to a requirement by a test objective, a yellow test objective symbol is shown at the intersection point within the matrix. By looking at the test requirements matrix, you can see which requirements are covered by which test cases and which requirements are not. As this matrix shows, test cases are defined for all requirements, and the execution of all of these test cases was successful.

**Figure 7. Tables show full requirements coverage of test cases and results of the test execution**

As you can see, the development and verification activities of the Rational Rhapsody Reference workflow can be mapped directly to the recommended methods and activities of the ISO 26262 process. This article has described the mapping for requirements-based testing activities and test requirement coverage. The remaining reference workflow activities, such as back-to-back testing and structural coverage measurement, are also mapped to ISO 26262 activities. Similar Rational Rhapsody tool support and automation exist to aid you in efficient use of these capabilities.

A similar mapping of the Rational Rhapsody Reference workflow development and verification activities can be defined for other standards, such as DO-178B, DO-178C, IEC 61508, or IEC 62304.

**Summary**

Delivering safe systems requires following a rigorous process, with considerable time spent verifying that the design is meeting requirements. Using the Rational Rhapsody Reference workflow...
workflow enables you to automate development of designs that must comply with safety standards.

The reference workflow provides several benefits for faster development of safety-critical systems:

- The concrete guidance on how to perform some of the recommended activities of ISO 26262 and IEC 61508 can also be applied to other standards.
- Traceability to requirements helps ensure that the design is meeting requirements and helps in understanding the impact of a change in requirements.
- Rational Rhapsody tools enable you to effectively perform these activities during your project work to maximize product quality and safety.
- The high degree of automation provided with Rational Rhapsody and Rational Rhapsody TestConductor Add On enables you to achieve your quality objectives within their given time and budget.

© Copyright IBM Corporation 2013
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)