Unit testing: the key to quality in smarter products

Contents

Introduction

Innovative and high-quality software is now not only a key driver for business success in many industries—it is also critical for building a smarter planet. Yet as companies race to meet tight deadlines, they often have to hack code and drop features. All the while, customer demands for faster delivery to the marketplace coupled with the exponential growth in complexity in systems and software are challenging the delivery of quality systems.

To remain competitive, companies must build quality into all aspects of their products starting in the requirements phase. This white paper focuses on delivering quality in software components, specifically the critical step of unit testing. Advances in model-based testing technology from IBM are designed to provide software engineers the ability to seamlessly integrate unit testing into their model-based development process, even using code developed outside of the modeling environment. This ability enables engineers to leverage code and models to get assessments against key scenarios based on the requirements for their software components. The benefits of increased modeling and automation can include reduced testing time, easier-to-understand tests, improved collaboration between developers and testers, faster time to market, and higher-quality releases.

The challenges of smarter product development

Product development has entered a new era of flexible, real-time customization where the products we rely on are consumed seamlessly with integrated experiences or processes. Today, customers expect products that can adapt to their unique needs, preferences and characteristics.

Embedded software is driving the innovation and acts as the "brains" that make products smarter. At the same time, companies are using microelectronic actuators, sensors and mechanical technologies to make products increasingly interconnected, intelligent and instrumented. For example, consider just the options in today's cell phones: They often include a huge array of features and customization options, including global positioning system (GPS) capabilities that must interact with multiple applications (including software from third parties), web capabilities, cameras, video cameras and, in some cases, thousands of applications that customers can pick and choose from.

This trend of using software to support advanced features and capabilities has happened very rapidly across many industries. Just 10 years ago, many of these options would have sounded like science fiction. Today, they're expected, and the trend is accelerating. Tomorrow, products are likely to grow even more complex, with software continuing to act as the main engine for innovation.

Spiraling costs and risks

Given all of these developments, manufacturers are essentially becoming software companies, so software development can have significant implications on product success. That's why efficient and effective software asset development and management processes are now so critical. At the same time, many companies face major struggles with software development and quality. For example, a Standish Group report finds that only 32 percent of software projects were deemed successful in 2009. Nearly half, or 44 percent, of projects were challenged, meaning they faced cost overruns and schedule delays and/or fell short of requirements. The remaining 24 percent of projects were failures, meaning they were canceled before completion or never used.[1] The cost to the industry is potentially billions of dollars a year. And this is all happening under current levels of complexity.

With the need for smarter products to be instrumented and connect to and interact flawlessly with various systems and hardware, higher development efficiencies are critical to getting costs and failures under control so they don't continue growing. One of the key reasons for the poor performance is that most companies are still using ad hoc techniques that are not adapted to modern challenges to deliver software for products that are intelligent, interconnected and instrumented.

By nature, these products are complex, and for such complex artifacts it's difficult to deliver quality on time. Model-based development has facilitated faster creation of new software because it enables software engineers to more clearly understand and analyze requirements, define design specifications, and even automatically generate code for direct deployment on the target hardware. Despite gains in model-based development, however, traditional testing processes have become a significant bottleneck because testing processes have not been adequately adapted to keep up with new development realities. Since testing processes are still largely code based and often manual, testing teams typically struggle to keep up with development processes, which can now rapidly produce designs.

Model-based unit testing, which can be performed using IBM Rational® Rhapsody® software, provides a way to introduce unit testing earlier in the development process and continually throughout the life cycle. With a proactive, model-based unit testing approach, teams can find issues

earlier in the life cycle of complex products while helping to make the testing process more intuitive and efficient in support of higher quality on tight deadlines.

Unit testing using a model-based approach

To better understand the advantages of model-based unit testing and how it works, it's useful to first examine how model-based development helps organizations overcome complexity-related challenges as well as the general functions and quality-related benefits of software unit testing.

Many organizations have turned to a model-based approach to optimize developer productivity in the face of ever-increasing complexity. By raising the level of abstraction from code to semantically rich graphical models, modeling makes it easier for teams to understand how an application will work while helping to improve collaboration and reduce the time needed for development. Models provide a basis for understanding and analyzing requirements as well as for process automation and design reuse. Teams can perform simulations using models to verify design concepts early in the development life cycle and get a head start on creating a quality product. Moreover, teams can automatically transform models into full production code for direct deployment on hardware with the help of standard modeling languages such as the Unified Modeling Language (UML) or Systems Modeling Language (SysML). The key benefits of modeling include the ability to deliver more complex and intelligent designs in less time.

Software unit testing

Software unit testing, which is a method to verify whether a unit of software meets the functional or performance characteristics identified in its design specification, helps ensure that basic software blocks are of a high quality. In this approach, the smallest testable part or unit of an application or software component is tested. Figure 1 shows the integration of unit testing in the V model development process. Software unit testing can be performed after the implementation of the unit is completed as well as during or even before the construction of the unit. This approach is often called test-driven development.

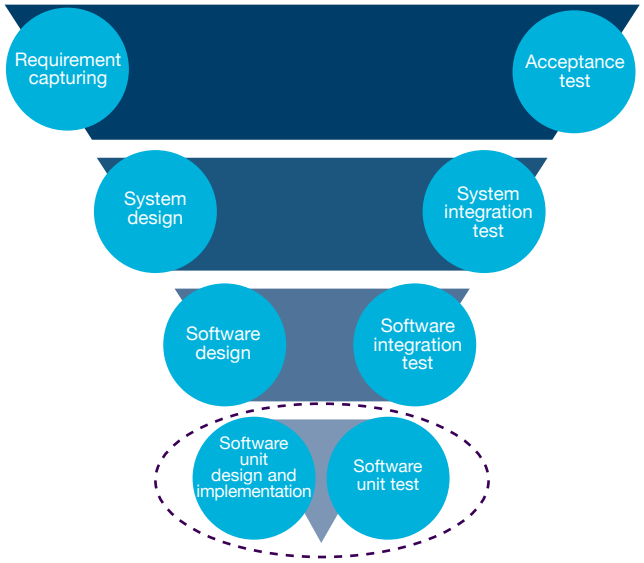

Figure 1: In the systems engineering V model, software unit testing is performed on individual components to help ensure the correctness of the tested software units. Software unit testing can be performed after the implementation of the unit is completed or even during the construction of the unit. This approach is often called test-driven development.

Software unit testing delivers several important benefits. For one, it can help increase product quality because each unit of software will adhere more closely to the interface and operational specifications for which it was designed. The resulting integration of software components is thus more deterministic, helping to shorten the integration test phase. Unit testing also supports standards for safety in critical system development (for example, DO-178B/C, ISO 26262 and IEC 61508). Perhaps the biggest benefit of software unit testing is that it helps facilitate cost reductions by enabling you to find defects in the coding phase, which can be significantly less expensive than finding them in later phases, as illustrated in table 1.

Table 1: The increasing costs of fixing a defect

Defect removal activity	Expected defects distribution (valid and invalid, best in class)	Cost multiplier (US$120)
Requirements review	4 percent	1
High-level design review	7 percent	4
Detailed requirements review	9 percent	2
Detailed design review	6 percent	7
Unit test	12 percent	10
System test	23 percent	16 (expensive)
User acceptance test	36 percent	70 (very expensive)
Production	3 percent	140 (outrageously expensive)

SOURCE: IBM Global Business Services®

Table 1: Defects that are fixed early in the development cycle are much less expensive to fix than those that are fixed later.

Model-driven unit testing: test models the way you design them

Model-driven unit testing capabilities available in Rational Rhapsody software build on best practices from model-based development and existing unit testing processes to create a testing approach that can help you better meet quality requirements in today's complex smarter products. Model-based testing tools enable you to perform testing at the model level using an industry-standard graphical notation such as UML, which aids in communicating the purpose of the test cases. It also enables testers to create scenarios starting with the requirements phase at the earliest levels of development, helping to ensure that tests address customer expectations. The model-driven testing approach is especially valuable in instances where teams are already using model-based development because the same language can be used for design and testing.

In contrast to code-based unit testing, model-driven unit testing employs models to graphically capture test cases and results, enabling you to design and execute unit tests and test architectures instead of only hand coding them. Thus, all benefits that apply for model-based development, including visibility, simplification and automation, also apply for model-based testing. When you use a model-based approach for acceptance testing, system integration testing and software integration testing, you can use the same process, mechanism and tools to complete the lowest-level unit tests.

Rational tools that support model-driven unit testing

The innovative capabilities that enable model-driven unit testing are delivered through IBM Rational Rhapsody TestConductor Add On software to Rational Rhapsody software, which is a visual development environment for systems engineers and software developers. Using industry-standard languages, including UML and SysML, Rational Rhapsody software helps teams collaborate to understand and elaborate requirements and abstract complexity visually. Teams can also validate functionality earlier in the development cycle and use automation to help speed the delivery of innovative products. Extending Rational Rhapsody software with IBM Rational Rhapsody TestConductor Add On software and IBM Rational Rhapsody Automatic Test Generation Add On software can help teams close the productivity gap between development and testing while supporting the delivery of higher-quality products.

Rational Rhapsody TestConductor Add On software

Several key capabilities are delivered through Rational Rhapsody TestConductor Add On software, including capabilities for defining tests, executing tests and analyzing test results. The tool can automatically generate test architectures in a graphical UML model format for portions of the test design that need to be tested. Test cases can be defined in the context of test architectures, as code, or as flowcharts or scenarios. Rational Rhapsody TestConductor Add On software executes test cases by launching the generated application on the host or target platform, with progress displayed in run time. Users can also step through test execution and perform scenario-based testing. In the end, users have comprehensive analysis options, including tracing code failures back to errors and inserting assertions to validate the execution of test cases. Overall, using Rational Rhapsody TestConductor Add On software can help you save time in identifying, understanding and correcting defects—in support of lower costs and higher quality.

Rational Rhapsody Automatic Test Generation Add On software

Rational Rhapsody Automatic Test Generation Add On software can analyze a design and generate sequence diagrams that help ensure that all aspects of the design are tested, including every state and transition. The solution can capture test cases in multiple formats, which can then be executed using Rational Rhapsody TestConductor Add On software. Because test generation is automated—and thus faster than manual approaches—more testing is possible throughout the development process, helping to optimize quality.

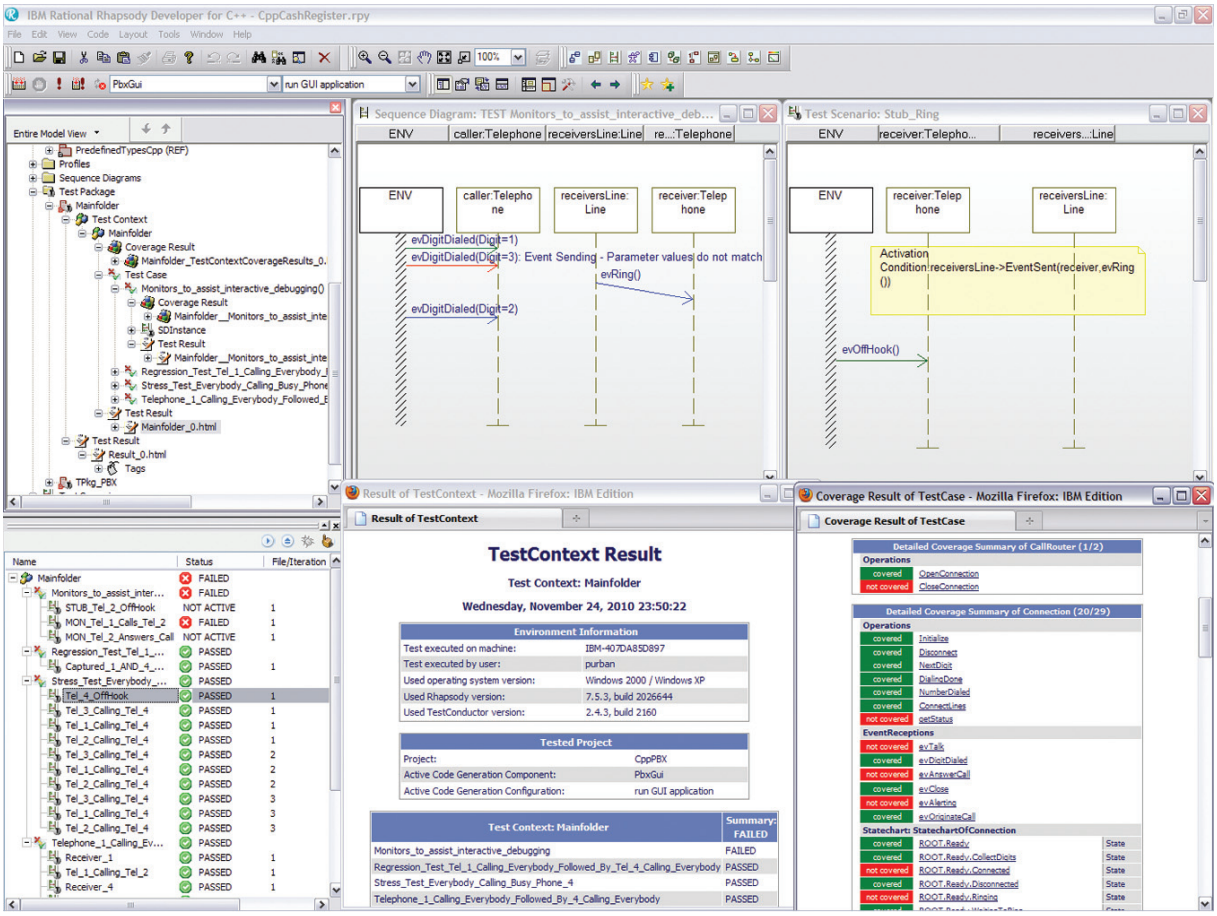

Figure 2: IBM Rational Rhapsody TestConductor Add On software enables users to create test cases using sequence diagrams, statecharts, flow charts or code. It then executes tests and monitors the results of the test cases, including pass/fail and model coverage reports.

Key benefits

The major benefits of model-driven unit testing include the following:

- It enables constant synchronization of test data because all test elements (such as test architecture and test cases) are directly linked to model artifacts, model/code data and corresponding test data. For example, changes in the model unit are directly visible in the modeled tests for the corresponding unit.
- It reduces testing complexity and increases maintainability by modeling test architectures and test behavior rather than just scripting and programming.
- It helps increase model (and final product) quality and helps reduce costs by applying test cases on the model level to support error identification in the modeling/design phase and by better ensuring the correctness of model components.
- It facilitates reuse of modeled test cases to test the actual code whether automatically generated from the design model or manually coded. Thus, models can be used to help ensure the correctness of the final implementations and code.
- It increases automation of test harness generation. Because model-based testing has direct access to model data, it can leverage the model data to automate most test activities that are traditionally performed manually.
- It enables automatic computation of test requirement coverage because test cases can be linked directly to modeled requirements (which may be imported from tools such as IBM Rational DOORS® software).

A modern testing approach for smarter products

Given the velocity of change in products toward ever more advanced features and greater customization capabilities, product development approaches necessarily must evolve. Model-based development provides a way for development teams to manage the complexity in today's and tomorrow's products, but without more efficient and effective testing processes, the potential gains are limited. Model-driven unit testing using Rational Rhapsody solutions is one of the best answers for making testing a valuable part of the development process from the requirements phase rather than a bottleneck as you rush to get quality products out the door. Adopting a model-driven unit testing approach can help drive more intuitive testing methods that help key players in the development process collaborate more effectively and work faster. But delivery speed is only one part of the equation. Quality is equally critical, and model-driven unit testing provides a way to more effectively meet customer and organizational expectations. In other words, it's a better approach to supporting smarter product development.

For more information

To learn more about IBM Rational Rhapsody testing solutions, contact your IBM sales representative or IBM Business Partner, or visit: **ibm.com**/software/awdtools/rhapsody/

Additionally, financing solutions from IBM Global Financing can enable effective cash management, protection from technology obsolescence, improved total cost of ownership and return on investment. Also, our Global Asset Recovery Services help address environmental concerns with new, more energy-efficient solutions. For more information on IBM Global Financing, visit: **ibm.com**/financing

© Copyright IBM Corporation 2010

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
December 2010
All Rights Reserved

IBM, the IBM logo, ibm.com, Rational and Rhapsody are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at **ibm.com**/legal/copytrade.shtml

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided "as is" without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

[1] The Standish Group, *CHAOS Summary 2009*, April 2009.

Please Recycle

Rational. software

RAW14245-USEN-00