IBM® Rational® TestConductor

Model Based Testing auf Basis des UML Testing Profile

WILLERT.
pioneers in embedded software engineering

Vorteile Modellbasierten Testens

UML-Modelle haben gegenüber C-Code einen, im wahrsten Sinne des Wortes, *mächtigen* Vorteil. Sie basieren auf einer Metastruktur, die unter anderem auch Notationselemente für Requirements, Linkbeziehungen, Test Harness, SUT *(System Under Test)* und Test Results beinhaltet, welche durch das UTP *(UML Testing Profile™)* von der OMG *(Object Management Group)* standardisiert sind.

Diese Notationselemente sind in Programmiersprachen *(ANSI C, C++, …)* nicht vorgesehen worden, vereinfachen jedoch die Definition und die automatisierte Durchführung von Tests sowie das Management von Testarchitekturen erheblich.

Vielfach macht man sich die Vorteile standardisierter Metastrukturen nicht in ihrer ganzen Wirkweise bewusst. Sie liegen in der vereinfachten Automatisierung wiederkehrender Aufgaben, in der Wiederverwendung von Arbeitsergebnissen und der Reproduzierbarkeit in Q-Prozessen. Durch Model Based Testing *(in unserer Betrachtung auf Basis von IBM Rational Rhapsody und IBM Rational TestConductor)* ergeben sich folgende Vorteile, auf die wir im weiteren Verlauf noch näher eingehen:

Automatische Generierung von:

- Testarchitekturen aus UML-Modellen
- Testarchitekturen aus Requirements für Unit-, Integration- und Systemtest
- TestCode aus modellierten Testszenarien *(Sequence-Diagrams, FlowCharts oder StateCharts)*

Automatische Erstellung von Traceability und Abgleich von Änderungen von/zu Requirements

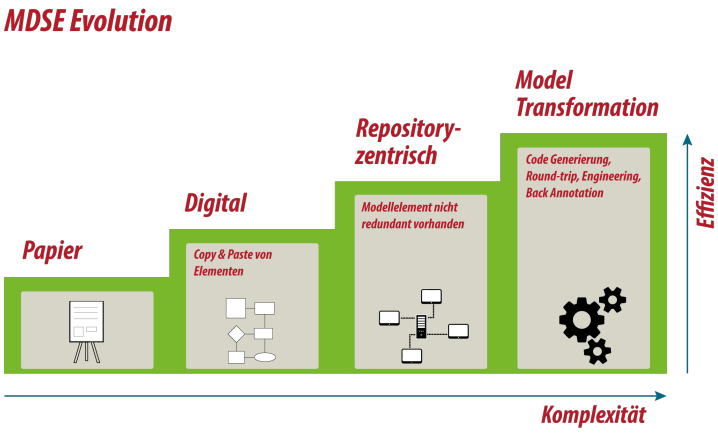
- Automatische Erstellung der Traceability zum Ausgangsobjekt *(vollständige Traceability von Requirements, über Modell-Elemente und Code zu korrespondierenden Testfällen)*
- Requirements Change und Impact Analyse *(Req-Synchronisation in RM-Tools)*
- Auto Reports über TestCoverage in Bezug auf Modell-Elemente und Requirements

Automatische Erzeugung von Dokumentation *(Reports über Coverage, Ausführung und Ergebnis)*

Laufzeit- und Code effizient *(keine zusätzliche Instrumentierung im Modell und dem daraus generierten Code)*

Die nebenstehende Grafik verdeutlicht den Produktivitätsgewinn durch Einsatz von MDSE, die Notwendigkeit für den Einsatz und die Entwicklung im MDSE selbst.

Bei der Überlegung des logisch folgenden nächsten Evolutionsschrittes fällt auf, dass zwischen Modellierung und Test eine große Produktivitätslücke besteht. Während in der modellbasierten Software-Entwicklung bereits Modellsimulation praktiziert wird, um dadurch in einer frühen Phase Designfehler zu finden, werden die Tests heute codezentrisch, manuell erstellt und meist auf der Target-HW durchgeführt *(und damit Fehler spät erkannt und vergleichsweise teuer behoben!)*



IBM Rational TestConductor

IBM Rational TestConductor ist das perfekte Werkzeug, um die aufgezeigte Produktivitätslücke zu schließen und damit Effizienz von Entwicklungsprojekten deutlich zu steigern. IBM Rational TestConductor ist auf Basis der Notation des UTP *(UML Testing Profile™)* nahtlos in IBM Rational Rhapsody integriert und co-existiert hervorragend mit anderen bekannten und wichtigen Profilen wie SysML, AUTOSAR etc.
UTP ist durch die OMG™ *(http://www.omg.org/spec/UTP/)* standardisiert.

Automatische Generierung von Testarchitekturen aus UML-Modellen

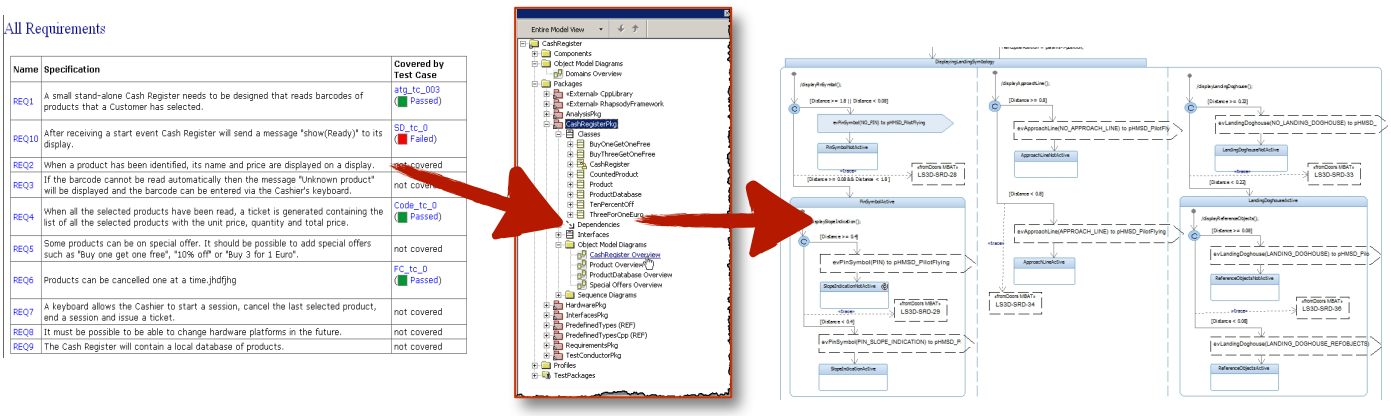
IBM Rational TestConductor ist eine UML-konforme Testumgebung für Realtime Embedded Systeme, die aus Rhapsody-Modellen automatisch Testarchitekturen erzeugt. TestConductor generiert auch TestCode aus, in FlowCharts oder Sequence Diagrammen, modellierten Testfällen. Aus den Architekturelementen *(Testkomponenten)* können wiederum mit einem Klick die für die Ausführung benötigten Testfälle generiert werden.

Model Based Testing kann auch auf externen C/C++ Code angewendet werden, indem man die Schnittstellen in Rhapsody bekannt macht ***(z.B. ganz einfach über die Reverse Engineering Funktion in Rhapsody)*****. Der Code muss nicht in das Modell übernommen werden und kann extern bleiben.**

Automatische Generierung von Testarchitekturen aus Requirements

Vorausgesetzt wird lediglich, dass die Requirements im Rhapsody Modell vorhanden sind. Mit dem WILLERT ReqXChanger können Requirements auf Basis des ReqIF Standards aus beliebigen RM-Tools *(z. B. IBM Rational DOORS; DOORS NG; Polarion)* nach IBM Rhapsody importiert und synchronisiert werden.

Darauf basierend können aus den Spezifikationen oder einzelnen Kapiteln, mit einem Klick automatisch Testarchitekturen generiert werden.


All Requirements

Name	Specification	Covered by Test Case
REQ1	A small stand-alone Cash Register needs to be designed that reads barcodes of products that a Customer has selected.	atg_tc_003 (Passed)
REQ10	After receiving a start event Cash Register will send a message "show(Ready)" to its display.	SD_tc_0 (Failed)
REQ2	When a product has been identified, its name and price are displayed on a display.	not covered
REQ3	If the barcode cannot be read automatically then the message "Unknown product" will be displayed and the barcode can be entered via the Cashier's keyboard.	not covered
REQ4	When all the selected products have been read, a ticket is generated containing the list of all the selected products with the unit price, quantity and total price.	Code_tc_0 (Passed)
REQ5	Some products can be on special offer. It should be possible to add special offers such as "Buy one get one free", "10% off" or "Buy 3 for 1 Euro".	not covered
REQ6	Products can be cancelled one at a time.jhdfjhg	FC_tc_0 (Passed)
REQ7	A keyboard allows the Cashier to start a session, cancel the last selected product, end a session and issue a ticket.	not covered
REQ8	It must be possible to be able to change hardware platforms in the future.	not covered
REQ9	The Cash Register will contain a local database of products.	not covered

Dabei wird auch automatisch die Verlinkung zwischen Quell- und Zielobjekt erstellt. Stichwort: Traceability!

Automatische Erstellung der Traceability zum Ausgangsobjekt

Bei zunehmend mehr Projekten muss auf Grund regulatorischer Anforderungen oder interner Qualitäts-Maßnahmen eine Traceability von den Anforderungen zum Quellcode und die Testabdeckung/Ausführung nachgewiesen werden.

Die Requirements werden von den zuständigen System- bzw. Software-Entwicklern nach dem Importieren mit entsprechenden Modell-Elementen in Rhapsody verlinkt und es entsteht dadurch ein, nach modernen QS-Anforderungen definiertes, vollständiges System. Über das Modell-Element existiert die Verbindung zum Requirement und zum generierten Code. Die Linkbeziehungen ***(Voraussetzung für Traceability)*** zu den Testfällen erstellt IBM Rational TestConductor automatisch.

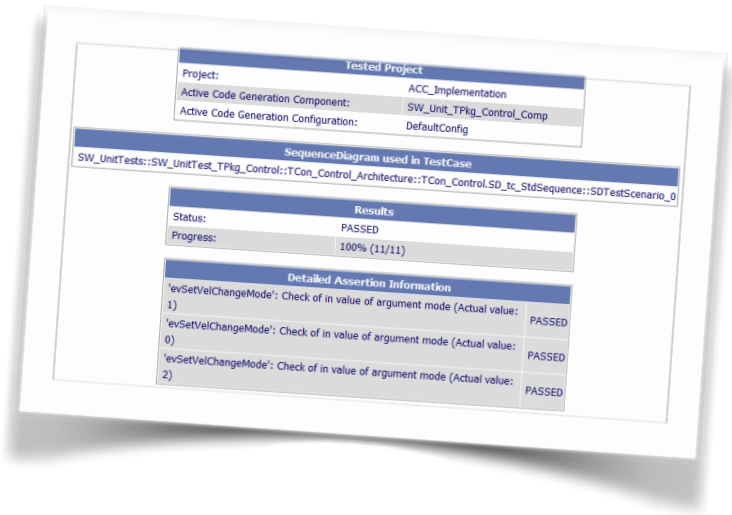
Mit TestConductor werden aus Requirements Unit-, Integration- und Systemtests generiert - und darüber hinaus auch Impact Analysen erzeugt und Suspect Links verfolgt.

Automatische Durchführung der Tests

IBM Rational TestConductor kann eine beliebige Kombination von Testfällen auf dem Host oder der realen Zielplattform ausführen. Informationen über den Fortschritt werden in Echtzeit angezeigt.

Um das Design oder Testfälle zu debuggen, kann man in der Ausführung schrittweise vorgehen, neue Testszenarien erstellen oder sogar neue Sequence-Diagramme automatisch generieren, um das Verhalten in der Ausführung zu beobachten.

Die Ergebnisse werden aufgezeichnet, im Modell abgelegt und können als Report ausgegeben werden. Wichtig: die Ergebnisse können mit den Passed/Fail-Informationen auch wieder in das RM-Tool zurück synchronisiert werden. So kann auf Basis der Traceability schnell und sicher geprüft werden, ob Testfall und/oder Requirement geändert werden muss.



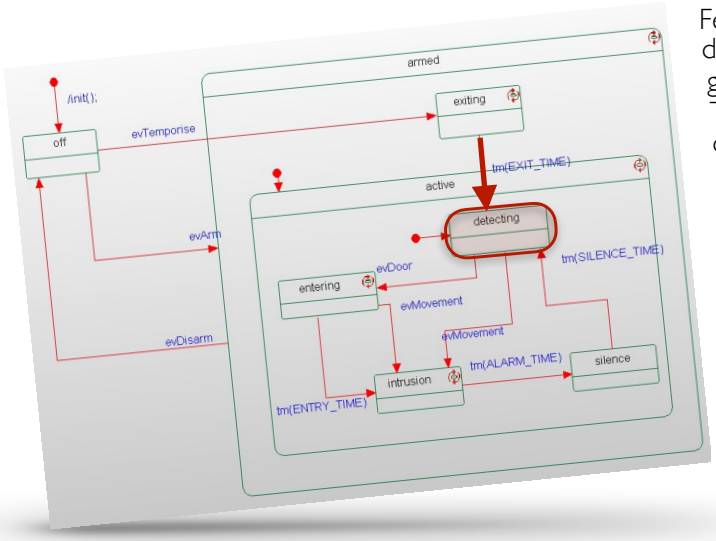
Tested Project	
Project:	ACC_Implementation
Active Code Generation Component:	SW_Unit_TPkg_Control_Comp
Active Code Generation Configuration:	DefaultConfig

SequenceDiagram used in TestCase
SW_UnitTests::SW_UnitTest_TPkg_Control::TCon_Control_Architecture::TCon_Control.SD_tc_StdSequence::SDTestScenario_0

Results	
Status:	PASSED
Progress:	100% (11/11)

Detailed Assertion Information	
'evSetVelChangeMode': Check of in value of argument mode (Actual value: 1)	PASSED
'evSetVelChangeMode': Check of in value of argument mode (Actual value: 0)	PASSED
'evSetVelChangeMode': Check of in value of argument mode (Actual value: 2)	PASSED

Analyse der Testergebnisse



Fehler, die im Test auftreten, können im Code bis auf die Fehlerzeile, respektive im FlowChart auf das grafische Element verfolgt werden. Es ist möglich, Testfälle zur Validierung „on-the-fly" zu ändern und das Verhalten in der Ausführung zu beobachten.

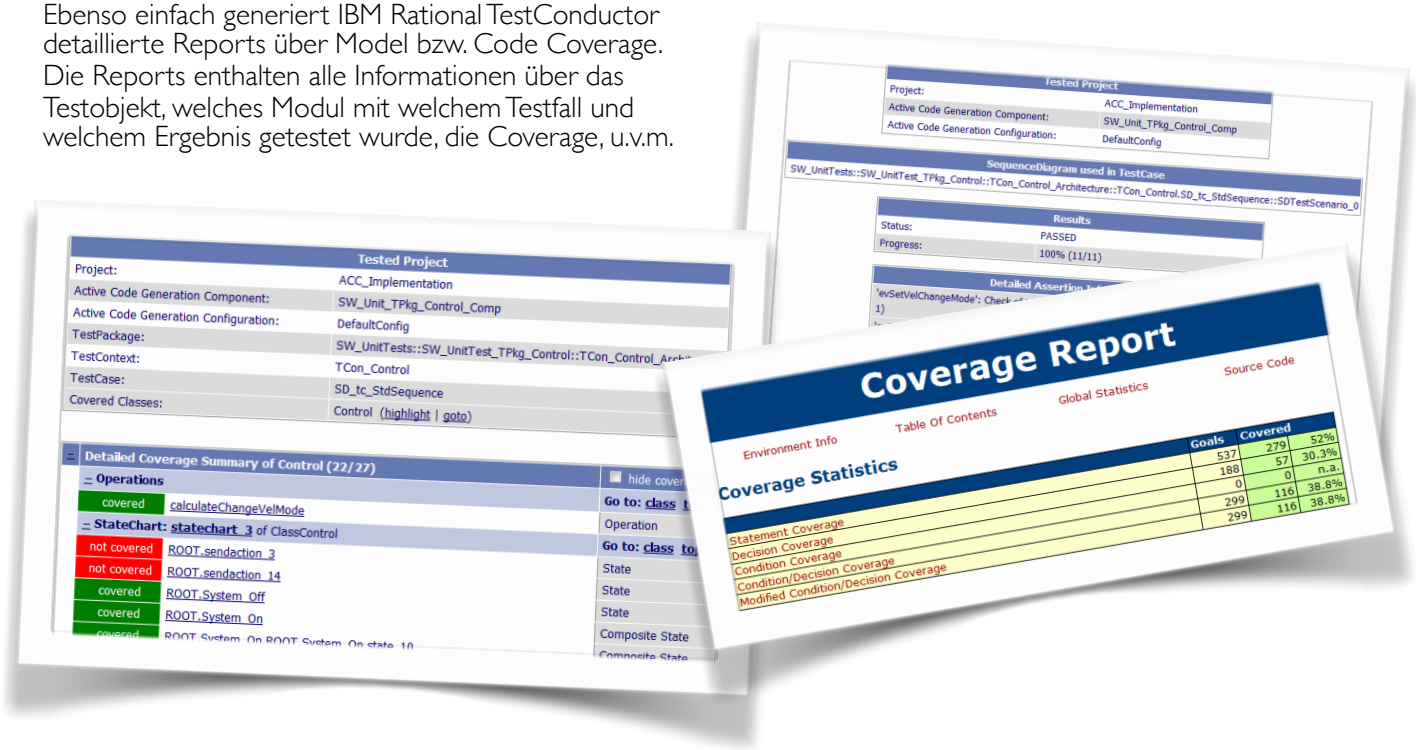
Wenn die Ausführung eines szenariobasierten Testfalls fehlschlägt kann ein Sequence-Diagram, mit allen aufgezeichneten Meldungen ***(inklusive der fehlerverursachenden)*** und der erwarteten Sequenz, generiert werden. Dies hilft, das Problem besser zu verstehen und macht es einfacher, das Design zu korrigieren.

In Verbindung mit Rhapsody kann das geänderte Design in einem animierten StateChart in der Ausführung Schritt für Schritt geprüft werden.

Test Reports und Coverage Analyse

Alle testrelevanten Informationen werden während der Testausführung aufgezeichnet und beim Beenden eines Testlaufs im HTML-Format im Modell abgelegt. Dadurch kann auf Knopfdruck ein Report der Testergebnisse erzeugt und ausgegeben werden. Als Standardausgabeformate stehen das HTML- + PDF- Format zur Verfügung. Optional kann auch nach MS-Word oder Excel exportiert werden.

Ebenso einfach generiert IBM Rational TestConductor detaillierte Reports über Model bzw. Code Coverage. Die Reports enthalten alle Informationen über das Testobjekt, welches Modul mit welchem Testfall und welchem Ergebnis getestet wurde, die Coverage, u.v.m.



Resümee

In klassischen Projektabläufen mit Coding durch Entwicklungs-Teams und Weitergabe an Test-Teams beansprucht das Testen bis zu zwei Drittel der Gesamtprojektlaufzeit. Zusätzlich wird der Quellcode mit TestCode und Testskripten versehen und die Traceability wird zeitaufwändig und fehlerträchtig in externen Tabellen gepflegt. Und genau hier steckt das Potenzial von Model Driven Testing.

Kunden die modellbasiert testen, bestätigen dass Sie Projektlaufzeiten signifikant kürzen und die Codegröße um bis zu 50% verringern konnten. Ganz nebenbei hat sich bei vielen die Zusammenarbeit zwischen Entwicklungs- und Test-Teams verbessert. Die Investition in Werkzeuge und Mitarbeiterschulungen ist gegenüber dem möglichen Einsparungspotential sehr gering.

Trainingsangebote

Workshoptermine zum TestConductor finden Sie tagesaktuell auf unserer Webseite.

Kostenfreie Webinare bieten wir periodisch und auf Anfrage kundenspezifisch an

Testversion

Eine Testlizenz stellen wir auf Anfrage gerne zur Verfügung. Der TestConductor kann bei der Rhapsody Installation als zu installierende Komponente optional ausgewählt werden.

Willert Software Tools GmbH
Hannoversche Str. 21
31675 Bückeburg
+49 5722 967860
info@willert.de
www.willert.de