

Title of Paper

A Component-based Approach to Test Automation

Presenter

Felice Cerullo / ST Incard (I)

Instructional Level

Introductory Intermediate Advanced

Target Group

Test Practitioners and Engineers (incl. Test Automators), Test Managers

Keywords

- Test Automation
 - Model-based Testing
 - Regression Testing
 - Test Management and Strategies
-

Abstract

Major challenges for test automation are *maintainability* and *reusability*.

One of the biggest problems in using automated test suites is keeping them functional as the software product changes. A software product typically changes many times during its life cycle and in most cases, this prevents old tests from running. Any changes, for example to the User Interfaces, very often require extensive changes to the automated test cases/scripts, thus jeopardizing, even severely, the test automation's viability. Another challenge is to design cases/scripts that can be (re-)used in different test projects (i.e. for testing different applications), rather than "reinventing the wheel" each time.

The actual (most used) test automation techniques and tools, based on the capture-and-replay, data-driven or keyword/action-driven paradigms, does not allow an effective reuse of the test cases/scripts, because they are too much monolithic and dependant from the particular Application-Under-Testing (AUT). These techniques still fail in case of frequent product changes or in reusing the same tests from one product to another.

By analysing a wide range of software products, we noted that very often, many different, heterogeneous (i.e. operating in different *contexts*), software products are made up by a set of common functionalities with more or less the same *behaviours*. From this analysis emerges that, to obtain reliable test automation it is essential to effectively model *behaviours* and *contexts* of usage. A viable solution is to design more abstracted test cases/scripts that verify behaviours and are highly configurable for running in different contexts. This allows separating the test cases/scripts from the specific AUT, thus allowing keeping test suites functional as the software product changes or in testing different products.

In order to address these questions about automation, maintenance and reusability, this paper pro-

poses a new approach called *Component-Based Testing*. This automation paradigm is founded on *Test Components*:

A Test Component is a reusable, context-independent and composable test unit, providing test services through its contract-based interfaces

By this approach new test cases/scripts are created by primarily assembling and configuring existing, pre-built, Test Components (rather than (re-)writing the test suites each time from scratch).

Test Components are conceived as self-consistent, building blocks that allow testing of well-defined behaviours in a wide range of contexts.

Test Components offers Test Services. A Test Service is a compound of atomic Abstract Test Cases (ATC), which concur to test a given behaviour in valid, limit and invalid scenarios. Test Components and ATCs are independent form the specific AUT: the possible usage contexts are modelled in a set of Context Variables (separated from the Test Services, hence from the ATCs) that must be set at runtime with the particular AUT/Context data.

All public Test Services are defined in the Test Component's interface; this grants that the same Test Component may have more than one interchangeable implementation. The interfaces, hence the provided and required Test Services, are stated in a *Contract*, following the design-by-contract paradigm. The Contract governs the interaction of the Test Component with the rest of the world and describes what the Test Component does, and how it behaves.

By this method, in order to create a new test suite, a test automator has to identify, from Contracts, the Test Components and Test Services needed to accomplish the particular test suite requirements/goals. Once selected, the Test Components has to be assembled and the Context Variables has to be set.

To turn in practice this approach, we have developed *Huracan*, an open source framework, for testing Java applications. Huracan is based on IBM Rational Functional Tester®. This agent-based framework defines the rules for building Test Components and offers a variety of services for assembling and running such components. Test Component Contracts are described in XML files. The Contract's syntax and semantics allows to model and set *pre-conditions* (including Context data), *test data* and *post-conditions*. In Huracan, Test Components are compounds of *Test Tasks*. A Test Task is an implementation of an ATC. Tasks concur to realise a Test Service. Tasks are "micro-component", because they have similar characteristics but limited capabilities. Tasks has Contracts like components, but cannot be accessed directly, they can be invoked only if correctly "plugged" to a Test Component, which is the owner of the Task.

Huracan has been validated in several test projects for testing both stand-alone and distributed Java applications. For this purpose 12 Test Components and 61 Tasks has been developed. The main results obtained were:

- reduction of testing time/cost: up to 50% (respect similar past test projects)
- higher test cases/scripts reusability: up to 80%
- increased test effectiveness [Number of Executed Tests / Bugs Found]: up to 45%
- reduction of the test documentation: up to 40%

CBT and Huracan are now successfully used by a first set of companies interested in developing this automation approach.

In the future we hope to have a huge set of Test Components developed by the tester community and made available to everyone. Therefore, a test automator will be able to find on the web the Test Components he needs, to create his own test suite. This scenario will cut considerably the testing

time and costs, improving also the test reliability.

We are currently working on developing more “intelligent” Test Components, by adding some “reasoning” modules to Huracan in order to better recognize and automatically manage product changes and contexts. The first results of this activity will be available at the beginning of 2008.

REFERENCE:

- Bachmann, B., Bass, L., Buhman, C., Santiago, C., Long, F., Robert, J., Seacord, R., Wallnau, K. *Volume II: Technical Concepts of Component-Based Software Engineering, 2nd Edition*, TECHNICAL REPORT CMU/SEI-2000-TR-008 ESC-TR-2000-007
- Meyer, B. *Applying Design by Contract*. Computer, IEEE, October 1992, pages 40– 51.
- Cerullo, F. *Component-Based Testing*, in Proceedings of EuroSTAR 2005.
- Virgilio, R. *Design, Implement and Reuse: A Component-based Approach for Testing Automation*, in Proceedings ICSTest 2006
- Cuomo, V. *Component-Based Test Automation*, in Proceedings of STARWEST 2007
- Component-based Testing: www.componentbasedtesting.org
- IBM Rational Functional Tester: <http://www-306.ibm.com/software/awdtools/tester/functional/>

Biography

Felice Cerullo is a Software Testing and QA Engineer in the R&D Division of ST Incard, a company of the STMicroelectronics Group. Felice has almost ten years of industrial and R&D experience in software engineering particularly in testing and automation. He manages software-testing projects of component-based systems. His research interests include testing methodologies, software metrics, testing component-based software, and CASE. He is speaker and participant at several conferences on software testing and software quality.

Contact information of Presenter

Full Postal Address	81025, ZI Marcianise SUD (CE), ITALY
Affiliation	ST Incard Srl – a company of ST Microelectronics group
E-mail Address	felice.cerullo@st.com
Phone Number	+39.0823.630.270
Fax Number	+39.0823.630.250
