

**S**oftware tests are valuable intellectual assets, especially in long-lived, multi-version, multi-platform commercial software. The highly-publicized Y2K software problem provides a very good sense of the problems that arise in such a domain as well as how long software tests should last. Software tests represent significant investment. Test developers are generally on their own to determine how to write better automated software tests. This leads to a number of problems, including: (1) Understandability: test cases and test oracles are typically buried in test code and hence difficult to rediscover; (2) Maintainability: automated tests are extremely sensitive to changes in the implementation. Both problems lead to difficulty in adjusting a legacy automated test because of the arbitrary nature of the current practice of test code development.

*“Write Once,  
Test by Anyone,  
Anytime, Anywhere,  
with Anything”*

**T**he ultimate goal for TestTalk is to support the following maxim: “Write Once, Test by Anyone, Anytime, Anywhere, with Anything”. By “write once”, we mean that test descriptions only have to be written once but can be used perpetually. New transformation rules evolve old tests to account for various changes. By “test by anyone”, we mean that TestTalk descriptions are so easy to understand that a tester can easily take over tests written by other testers or developers. By “anytime”, we mean that TestTalk tests survive over time through application evolution and revisions. By “anywhere”, we mean that TestTalk tests can be transported to another platform or operating system without modification. By “with anything”, we mean that switching the test automation environment does not nullify TestTalk tests.

**T**estTalk is a software test description language designed for specifying test cases and test oracles in a manner natural to the software testing process rather than the programming or development process. TestTalk helps testers focus on requirements and design aspects of software tests rather than the implementation details of test execution. By enabling practitioners to separate concerns between software test description and test execution, TestTalk provides the means for creating software tests that are readable, maintainable, and portable, yet executable.

### Highlights

- Software tests are described in domain-specific terms using abstractions natural to testers.
- Transformation rules are used to transform test descriptions into executable forms.
- When it is necessary to transport tests to another platform or another testing environment, all that is needed is a new set of transformation rules.

**W**e are building a toolset to support the TestTalk language. The current toolset consists of a prototype parser and translator, which recognizes test descriptions and transformation rules (both expressed in what we consider the core language). As illustrated in figure 1, TestTalk produces automated test programs for the application-under-test (AUT) for a specific platform and test automation tool by using the transformation rules in the translation of the test descriptions.

**T**estTalk facilitates writing better automated software tests. In particular TestTalk helps achieve the following:

- Software test descriptions are more readable and understandable by testers and developers alike;
- Software tests are maintainable and can be adjusted quickly to changes in the application under test without modification of each individual test;
- Software test descriptions can be reused without modification when transplanted to another platform;

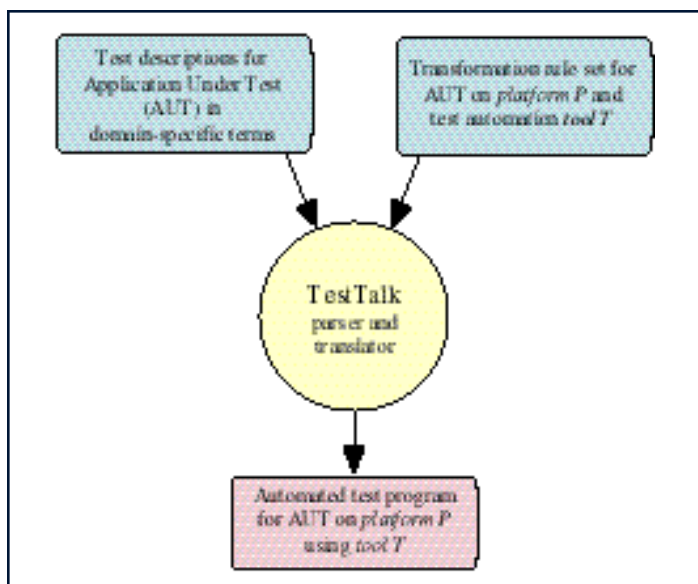


Figure 1

- Software test descriptions can be reused with a different test automation tool;
- Software test descriptions can be embedded in a requirements document or traditional test document and extracted from that document for testing;
- Software tests can be used to generate test data and to provide better support for test adequacy analysis.

A TestTalk description consists of the following definition sections (which can be provided in any order and can be scattered in different locations and files, as long as the TestTalk parser and translator can locate them):

- A *setting definition* section defines the “one-for-all” testing environment, which changes whenever the AUT advances to a new revision, platform or test tool;
- A *dialect definition* section defines a dialect used in scenarios, test cases, action lists, or oracle definitions;
- A *transformation rule set definition* section defines how scenarios, test cases, action lists, or oracles in a certain dialect should be transformed into executables or other useful formats for test automation;
- *Scenario / Action List definition* sections define steps to carry out tests;
- *Oracle definition* sections define test oracles using a dialect;
- A *test suite* definition section defines a set of test cases that are logically related.

The experiment illustrated in Figure 2 demonstrates the versatility of TestTalk, whereby only transformation rules must be changed when transporting to a new version of the AUT, a different computing platform, or a different test automation tool; the multitude of test descriptions can be reused as originally specified.

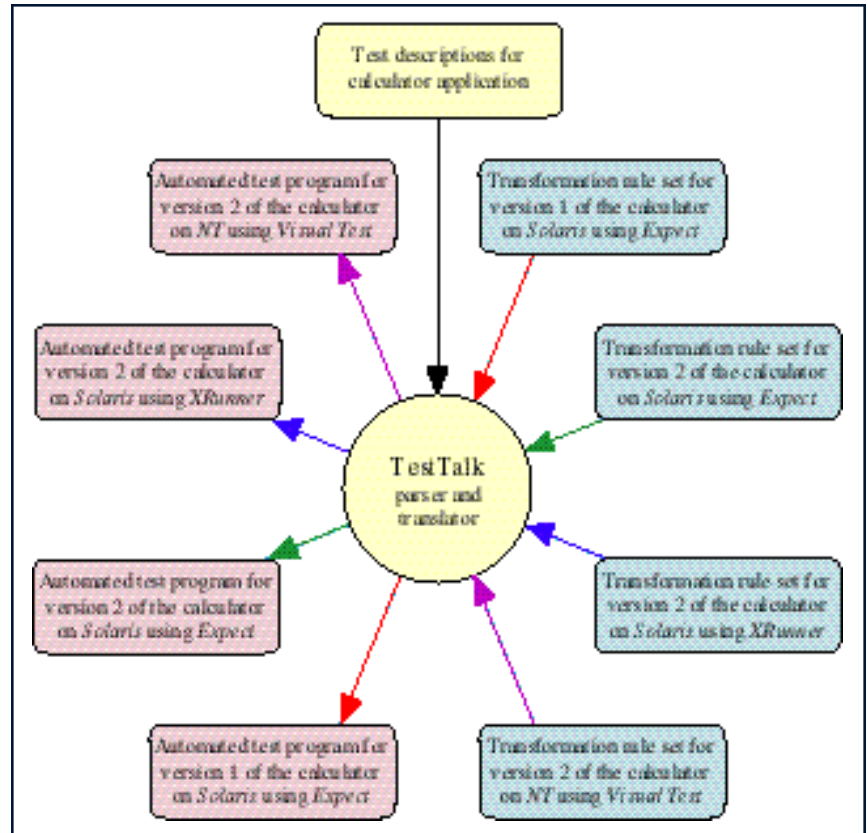


Figure 2

## References

1. Chang Liu, Debra J. Richardson, “TestTalk Language Reference (Version 0.1)”, Technical Report 99-07, Information & Computer Science, University of California, Irvine, February 1999.
2. Chang Liu, Debra J. Richardson, “TestTalk, A Test Description Language: Write Once, Test by Anyone, Anytime, Anywhere, with Anything”, Technical Report 99-08, Information & Computer Science, University of California, Irvine, February 1999.
3. Chang Liu, Debra J. Richardson, “Programming Languages Considered Harmful in Writing Automated Software Tests”, Technical Report 99-09, Information & Computer Science, University of California, Irvine, February 1999.



The **ROSATEA** tools are research prototypes, some of which have been successfully transitioned into use on real projects, but not yet as commercial systems. These tools were developed by the Research Organization for Specification- and Architecture-based Testing E (&) Analysis at the University of California at Irvine. The work was done in conjunction with the Perpetual Testing Projects sponsored by the DARPA’s EDCS program.



The **TestTalk** research and development effort is sponsored in part by: the Air Force Materiel Command, Air Force Research Laboratory, and the Defense Advanced Research Projects Agency under agreement number #F30602-97-2-0033. The views and conclusions contained herein are those of the researchers and should not be interpreted as representing the official position or policy, either expressed or implied, of the U.S. Government, AFMC, Rome Laboratory, DARPA, or the University of California, and no official endorsement should be inferred.

### Freely Available Software

Information about UC Irvine’s  
*Research Organization for  
 Specification- and Architecture-based  
 Testing E (&) Analysis (ROSATEA)*,  
 as well as its software, is available at:  
<http://www.ics.uci.edu/pub/rosatea/>

### Contact Information

Professor Debra J. Richardson  
 Information and Computer Science  
 University of California  
 Irvine, California 92697-3425

url: <http://www.ics.uci.edu/~djr>  
 email: [djr@ics.uci.edu](mailto:djr@ics.uci.edu)  
 voice: 949-824-7353  
 fax: 949-824-1715