### *TestTalk: Software Test Description Language: http://www.ics.uci.edu/~djr/edcs/PerpTest.html*

University of California, Irvine - Debra Richardson

Software 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.

TestTalk 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.

The 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.

We 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). The TestTalk toolset produces automated test programs for the application-under-test for a specific platform and test automation tool by using the transformation rules in the translation of the test descriptions.

### *UML/Analyzer - A System for Defining and Analyzing the Conceptual Integrity of UML Models: http://sunset.usc.edu/*

University of Southern California, Center for Software Engineering (USC/CSE) – Alexander Egyed

Software development is about modeling a real problem, solving the model problem, and interpreting the model solution in the real world. In doing so, a major emphasis is placed on mismatch identification and reconciliation within and among system views (such as diagrams). UML/Analyzer describes and identifies causes of architectural and design mismatches across UML views as well as outside views represented in UML (e.g., C2 style architectures).

- It is integrated with Rational Rose (market leader for OO modeling)
- It implements a generic view integration framework
- It incorporates UML's Object Constraint Language (OCL)

UML/Analyzer supports the definition of mismatch rules and model constraints. It also defines what information can be exchanged and how it can be exchanged. With that, architects can identify and resolve inconsistencies between views automatically:

- Mapping: Identifies related pieces of information and thereby describes what information is overlapping and can be exchanged.
- Transformation: Extracts and converts model elements of views in such a manner that they can be interpreted and used by other views (how to exchange information).
- Differentiation: Traverses the model to identify (potential) mismatches within its elements. Mismatch identification rules can frequently be complemented by mismatch resolution rules.

UML/Analyzer is integrated with Rational Rose and is used to create and modify views (synthesis). Rational Rose models are converted through an automated process into UML-A where they are analyzed via UML/Analyzer. Model constraints and mismatch rules are verified via a parser component. The conceptual integrity of the model is then validated through the model checker component. The model checker makes use of mapping, transformation, and differentiation. Generated modeling information as well as identified model mismatches can be fed back into Rational Rose for visualization.

- UML/Analyzer identifies inconsistencies and incompletenesses,
- Model currently supports class, object, sequence, collaboration, state, and various architectural diagrams (e.g., layered and C2)
- Model constraints, mismatch rules, and transformation rules can be modified without programming.

### *WebDAV: http://www.ics.uci.edu/pub/edcs/*

University of California, Irvine – Richard Taylor/David Redmiles

WebDAV is an extension of HTTP that provides a standard infrastructure for asynchronous collaborative authoring of a wide variety of content across the Internet. WebDAV has been approved by the IETF and is being actively developed by a number of Software vendors, including Microsoft, IBM, Xerox, Novell, DataChannel, and CyberTeams. This demo will feature a WebDAV client (WebDAV Explorer) which will show how the WebDAV protocol facilitates collaborative use of distributed files.