



SILVERMARK'S SMALLTALK TEST MENTOR

WE HAVE MORE THAN A DOZEN programmers developing a large automobile insurance application using VisualAge for Smalltalk. In our rapid development environment, we frequently release two or more versions a week. To support this fast-paced environment, we needed a flexible automated regression testing tool that would enable us to create and execute reliable, reusable, and maintainable automated testing scripts.

Smalltalk Test Mentor allowed us to achieve these goals, while many of the tools which reside outside the VisualAge environment could not. Using Smalltalk Test Mentor, we were able to develop test scenarios using reusable "scriptlets" so modifications need to be done in only one place, with many scripts based on test code automatically generated from recorded interactions with our application.

Smalltalk Test Mentor presents and organizes tests within a tree-view based editor. The tree shows the hierarchy of suites, scenarios, and steps. A test suite, which is really a Smalltalk class behind the scenes, is the highest level of the test hierarchy. Test suites contain scenarios that are composed of test steps. Test steps are the building blocks of Smalltalk Test Mentor. The tool is well stocked with a variety of test steps. Some of the test steps that SilverMark provides are: UI recordings, UI scripts, verification scripts, instance methods, class methods, workspace, collection of steps, file iteration, and unspecified. The unspecified step enables a test designer to design and document a test before a programmer or technical tester implements a test. Smalltalk Test Mentor also contains suite, scenario, and script steps. These steps

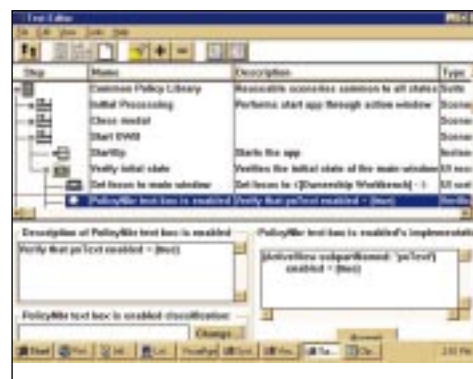


Figure 1 Test editor.

allow you to execute suites, scenarios, and test steps from within other test suites. Using these steps properly is the key to designing reusable test scenarios that require minimal maintenance.

Test results are presented using a traffic light paradigm within a tree view. Within this paradigm, a green light signifies the test step was successful, the yellow light indicates a failure, and the red light denotes that an exception/walkback has occurred. Smalltalk Test Mentor also displays method coverage metrics when you specify the applications to be covered. By doing so, the results browser shows method coverage statistics for each class in the specified applications. The browser also allows you to delve deeper and see each method that was or was not tested. The results browser also provides execution times for each step that rolls up to the test scenario. These times are useful for developing application performance tests. Test results can be saved in Envy or in a file and later retrieved for comparison using the results differences browser. The results differences browser allows you to set up test baselines and

Contact:
SilverMark, Incorporated
4068 Barrett Drive
Raleigh, NC 27609
<http://www.silvermark.com>
info@silvermark.com

(888)588-0668
(919) 870-7994
(919) 870-7885 (Fax)



Martin Belan is a Senior Programmer/Analyst at Progressive Insurance Co. in Cleveland, Ohio.
Martin_Belan@auto-insurance.com

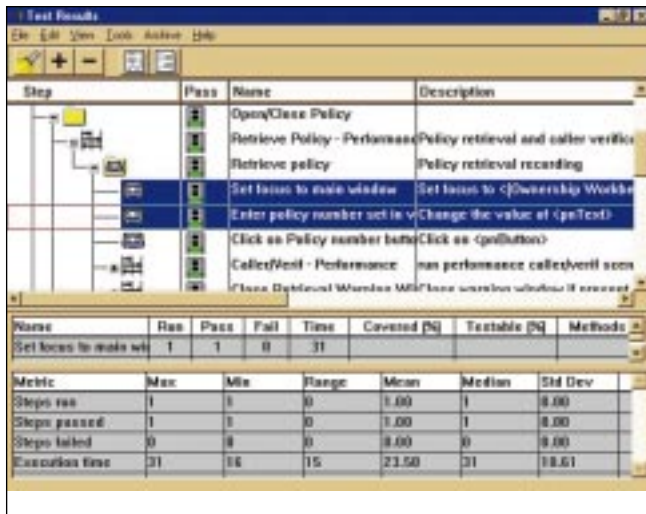


Figure 2 Test Results browser.

then generate a report when the current deviates from the baselines.

Smalltalk Test Mentor also includes a test runner view that allows you to control the test execution. In the test runner, you can specify which steps to run and pause execution of a test. The test can be paused prior to the execution of a step, after execution of a step, if a step fails, or if there is an exception. Once execution is paused, test steps can be inspected and the debugger can be opened to find the cause of an exception or to debug test steps.

One of our primary concerns about using a regression testing tool was maintenance. We have a very large dynamic application that has to support numerous US states, each with its own special rules and regulations, with more than one hundred windows. Our challenge was to provide adequate test coverage while keeping maintenance to an acceptable level. We accomplished this by creating scenarios using Smalltalk Test Mentor's UI recording and verification interface and then factoring out the recorded scenario into miniature reusable scenarios. For example, we only have one data retrieval scenario in our test library. Each test scenario that executes the retrieval scenario sets up data in variables that the data retrieval scenario uses when it executes. Later, we took this one step further by using a file iteration step to repeatedly execute the data retrieval scenario over a variety of insurance policy test data.

Creating scenarios using the UI recording and verification steps is easy and does not require a technical person.

However, the real power of Smalltalk Test Mentor is that it exists within the Smalltalk development environment and has access to the Smalltalk development image. We have created test suites that serve as libraries of reusable test scenarios that we combine to create our regression tests.

We have striven to make these test scenarios intelligent so

they can be reused across the application in multiple application states. This will help minimize rework and maintenance of the test scenarios. For example, instead of recording verification scenarios for each of our application's wizard pages to verify the presence and state of the wizard buttons, we coded a verification scenario using an instance method step that verifies the wizard buttons for any wizard window in the application. The scenario knows the Back button should not be enabled on the first page of the wizard and that the Finish button instead of the Next button should be present on the last page of the wizard. The wizard forms are reused in several different wizards in the application, so each page's verification script only needs to call the wizard button verification script.

Smalltalk coding is not necessary to get reuse from Smalltalk Test Mentor. One of the first reusable test scriptlets we wrote was for the verification of the OK and Cancel buttons. One of the standards in our application is that every modal window must have an OK and Cancel button, and that the mnemonics and default button are the same on each. This verification script is simply a UI recording that can be called from any modal window's verification script, so if the standards for these buttons change, the test scenarios will need to be changed in only one place. Of course, to be able to get this kind of reuse from Smalltalk Test Mentor, your programmers need to follow naming standards and should be getting reuse within your application. In the preceding example, if the OK and Cancel buttons were named differently on every window, it would be much more difficult to create a reusable verification scenario.

We have also found ways to use Smalltalk Test Mentor other than regression testing. We have begun to develop scenarios that automatically determine which US state-specific policy attributes to test for by having Smalltalk Test Mentor interface with a class that understands which specifics are applicable to each state. Combining the generic regression scripts with the state specific dynamic scripts gives us the flexibility of running a policy from any state through the same scenario with Smalltalk Test Mentor deciding what needs to be tested.

We have also used Smalltalk Test Mentor to unit test domain base classes. This was accomplished by reusing the policy retrieval script that was created for regression testing and then executing a workspace which set up the objects the same way our persistence classes would. With the business objects instantiated, I was then able to finish development and test the user interface using a live policy and real data. Using Smalltalk Test Mentor to test the domain base classes and the user interface, we were able to have several developers working in parallel on the business objects, user interface, persistence, and mainframe programs. There were fewer errors during final integration testing because

the business objects and user interface were previously tested using Smalltalk Test Mentor.

Even though Smalltalk Test Mentor is a young product, its current version is very useable. The SilverMark staff has been very responsive in correcting bugs and implementing enhancements we have requested. They have also been very helpful via Internet and phone support in debugging script problems and in brainstorming ways to develop scenarios for problem areas.

Being a young product, Test Mentor does have some room for improvement. Many of the improvements can be categorized as scenario maintenance and Silvermark intends to improve some of these items in their next release. The copy function in the test editor is unorthodox and does not follow Windows GUI standards. When copy is selected, a copy of the step or steps is pasted directly underneath the copied step. The step then needs to be drag-and-dropped to the appropriate location. Also, scenario steps cannot be copied across suites. These limitations are not show-stoppers but they can be a nuisance. Another useful enhancement would be the ability to browse senders of a scenario. We have been able to create reusable suites of scenarios using Test Mentor; however, currently there is no easy way finding references to them.

Another welcome enhancement would be the ability to filter steps in the results browser window. Currently, all steps are shown regardless of whether they pass or fail. Traversing through a large tree of steps can be tedious and time consuming. Adding a couple quick filters such as: show only failed steps or named steps, would be very helpful. We were able to work around this problem by extending Smalltalk Test Mentor and customizing a report generated from the results browser, a definite advantage of choosing a tool that works within the VisualAge IDE.

Silvermark has plans to implement enhancements in the next version for the copy and browse references features discussed earlier. Undo/redo functionality, wizards to aid in scenario creation, and more reporting options with better formatting are also planned.

All in all, Smalltalk Test Mentor has exceeded my expectations for a Smalltalk testing tool. Universal testing tools that live outside the Smalltalk development image just don't have the information available to them to develop powerful regression tests for VisualAge for Smalltalk applications. In most testing tools, test scripts are either written in a C-based language or in a scripting language of their own. Test Mentor scripts are written in Smalltalk, so your Smalltalk developers can write test scripts in the same language that they are developing in. By using Smalltalk Test Mentor, we have been able to develop flexible, reusable regression tests for our VisualAge application as well as explore other uses for this tool. ♦