Dance Away Your Testing Blues

GUI*dancer* to the Rescue

GUIdancer is an Eclipse-based tool for the automated testing of Graphical User Interfaces (GUI's), with a focus on programs written with Java and Swing. The beauty of GUIdancer lies in the fact that you do not require programming expertise at any stage of the testing process. With the latest release of the plug-in, testers using GUIdancer can now easily specify tests for applications containing non-standard components without even the slightest hint of programming expertise.

Put On Your Dancing Shoes

GUI*dancer* has been developed to make automated GUI-testing easier. Since reusable, maintainable tests can be specified, executed and analyzed by testers without programming, the costs associated with the testing process is greatly reduced. Test cases in GUI*dancer* are not programmed or coded; Rather, they are specified. The test developer requires no knowledge of programmatic structures such as loops and control statements because these structures do not exist in GUI*dancer*. Tests cases are specified in an intuitive, almost natural language. For any step in any test, it must be decided:

- Which component should be tested,
- Which action to test on the component, and
- What parameters or data to apply to the action.

For example, a test step may be to "click a button once", or to "enter 'hello' into a text field". Instead of recording these steps in a running application, or writing a piece of code that will do this, a GUI*dancer* user specifies exactly these three details, which combine to form a 'test step'. Test steps and reusable test cases can thus be easily created using this intuitive process, which requires no programming skills at all. These test cases can be combined and recombined to

GUIdancer Benefits At a Glance

- Requires no programming knowledge and hence advantageous to project managers and testers
- Users can craft "pools of simple tests that are reusable across projects
- Changes in the Application Under Test (AUT) require no changes to the test cases

make tests as complex as necessary. No scripts need to be generated, modified, and separately maintained. Basically, if you can describe a series of steps in terms of what actions to perform and where, you can write a GUI*dancer* test case.

The real benefit of GUI*dancer* is that even without programming knowledge, a test developer can reap the rewards of such powerful concepts as reusable test cases, multilanguage testing, and combination of simple test cases to produce more powerful, complex test cases.

Get Into Rhythm

In addition to a high degree of reusability, the program offers flexibility and resilience to changes in the application under test. Changing the place, properties or type of component does not affect the (re)location of the object in the application under test. Tests can easily be kept up-to-date with the developing software without the need for re-specification. GUI*dancer* also keeps specification and execution initially separate, so that specification can begin even before the application under test is ready.

Since the specification does not involve the production or maintenance of program code (scripts, XML, and so on), programming knowledge is not a prerequisite. Additionally, the specification can be done manually or using the observation mode, both of which produce the same output structure.

Show Off Your Moves

One of the primary features of any testing tool is test specification. GUI*dancer* shares this feature with many testing tools. Fewer testing tools have the ability to map Application Under Test (AUT) components in such a way as to insulate test cases from changes to the AUT. Even fewer free test developers from the burden of separately maintaining test scripts and related test specifications. GUI*dancer* offers the above functionality, plus a testing model that enables, encourages, and rewards reuse and combination of test cases.

GUI*dancer* is extensible. Software developers can follow a well-documented process, allowing GUI*dancer* to support new custom actions and even graphical components.

One of the key features of GUI*dancer*, and one that is rarely offered by other testing tools, is the ability to start testing early in the development process. As soon as the requirements for the AUT are available, relevant test cases can be specified. Of course, with GUI*dancer*, these specifications are executable. Unlike other automated testing tools, there's no need to translate the test steps into a scripting language. These tests can then be executed in the same instant that the relevant portion of the AUT is developed, which means that defects can be spotted earlier in the development process. Defects are much less expensive to correct when they are found earlier rather than later.

On the other hand, we realize that not everything always goes according to plan. Sometimes, certain tests are not specified until after the AUT is implemented. In this situation, test developers must rush to specify the test case, and may not produce a test case that is modular and reusable. As such, GUI*dancer* supports refactoring of test cases, allowing complex, monolithic test cases to be decomposed into simpler, reusable units.

Some of the other features include:

- **Robustness:** once specified, tests are joined to the AUT by mapping the necessary component from the AUT and assigning it to the user-defined component specified during test creation. The mapping process is independent of object and position, and relocation of components does not depend on the place or properties of the object remaining the same. Even changes to component type often do not affect test execution. The tester can define the exact search terms for objects in the AUT. Changes that do affect test execution can usually be integrated into the test by re-mapping the object. The need for re-specification is rare.

- **Maintainability:** Using the selection of editors and views, test data can easily be modified to keep in line with developments in the software.

- **Reusability:** test elements specified with GUI*dancer* have a large capacity for reusability. As well as an extensive referencing system, GUI*dancer* offers the tester the ability to map separate objects in the application under test to the same test case. This considerably reduces the time spent specifying the same action for different components. Combined with the ability to add and overwrite test data at places where test elements are reused, these features enable large complex tests to be specified using a small number of

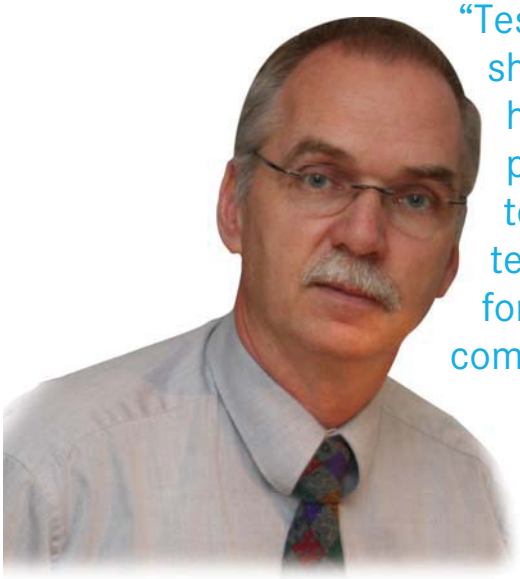
test cases. GUI*dancer* also supports multi-lingual testing, enabling data specified for one test case to be translated and added to the same test case. This eliminates the need to specify multiple test cases for different languages.

- **Error Handling and Reporting:** errors during test execution do not necessarily have to stop the testing process. GUI*dancer's* Event Handling options let testers specify actions and even other test cases to be executed in case of errors during testing. Test results and any errors can be clearly seen in a results view, can be saved as an HTML or XML file, and can be used in other software tools.

- **Additional Benefits:** GUI*dancer's* database system provides individual and multi-user storage options, resulting in easy collaboration on projects, to promote efficiency in project creation and maintenance.
  - o The client-server architecture used by GUI*dancer* allows the client and the server to be installed on different computers in the network, or even on different platforms from each other.
  - o Based on the Eclipse Rich Client Platform, GUI*dancer* is platform-independent. It has a modern, state-of-the-art user interface, which is configurable, intuitive and easy to use. The program works as a standalone application or as an Eclipse plugin, keeping the same functionality in both options.
  - o Additional or user-defined GUI objects can also be integrated into the tool using the API provided.
  - o GUI*dancer* is compatible with many version management systems, allowing projects to be stored as text files in any version control system.
  - o The focus on Java applications, which use Swing as a GUI, does not limit GUI*dancer* to Swing by design. Additional types of GUI's are planned for future releases.

GUI*dancer* 1.1: Add Extensibility to the Jig

After the success of the GUI*dancer* 1.1 preview version, BREDEX GmbH has released the full version 1.1 of the GUI testing tool. The latest version is available for Windows and UNIX systems, and incorporates a host of new features, most importantly, an interface to extend the tool's repertoire to other Java/Swing components and actions. "It is less than a year since GUI*dancer* was released, and the developments are showing no sign of slowing" said Hans-J. Brede, Managing Director. "Carrying on from the preview version, we've been improving and adding features from our release plan and from suggestions from our customers. The most important feature of this release, however, is the extensibility interface. GUI*dancer* has a large repertoire of components and actions, but there are often variations in their behavior in applications. We believe that testers shouldn't have to program to create tests, even for modified components".



"Testers shouldn't have to program to create tests, even for modified components"

Hans-J. Brede, Managing Director, BREDEX GmbH

Many applications use components, which deviate from standard uses or implementations. GUI*dancer* offers a range of actions on various different components. However, there are always going to be instances where a component has been modified to act in a slightly different way, which can be critical for the GUI test. Because of this, the GUI*dancer* team has provided an extension interface with detailed instructions describing how to test custom Java/Swing components. The extension of GUI*dancer* to accommodate custom actions and components is a well-documented process with two general steps. First, the developer must create Java tester classes that implement the desired action on the custom component. These classes reside on the GUI*dancer* server alongside the AUT. Then the developer modifies an XML configuration file for the GUI*dancer* client. This makes the new action/

component visible and accessible within the client GUI. The documentation for this process is, of course, provided with GUI*dancer*, and contains general instructions as well as code examples.

Extensibility isn't the only new feature in GUI*dancer* 1.1. One of the complaints with earlier versions of GUI*dancer* has been the lack of Ant integration. This version also offers a command line interface that executes tests without the GUI*dancer* GUI. This feature allows tests to be run within, for example, Ant scripts or batch files. GUI testing can now be integrated directly into the build process.

GUI*dancer* 1.1 also provides testers with the ability to execute incomplete tests as they are being specified. This simplifies testing of complex systems, as the tester can quickly identify, for example, which component hasn't yet been mapped or which parameter requires a value.

GUI*dancer* 1.1 also offers a set of additional new features and enhancements. It has added support for AUTs that use

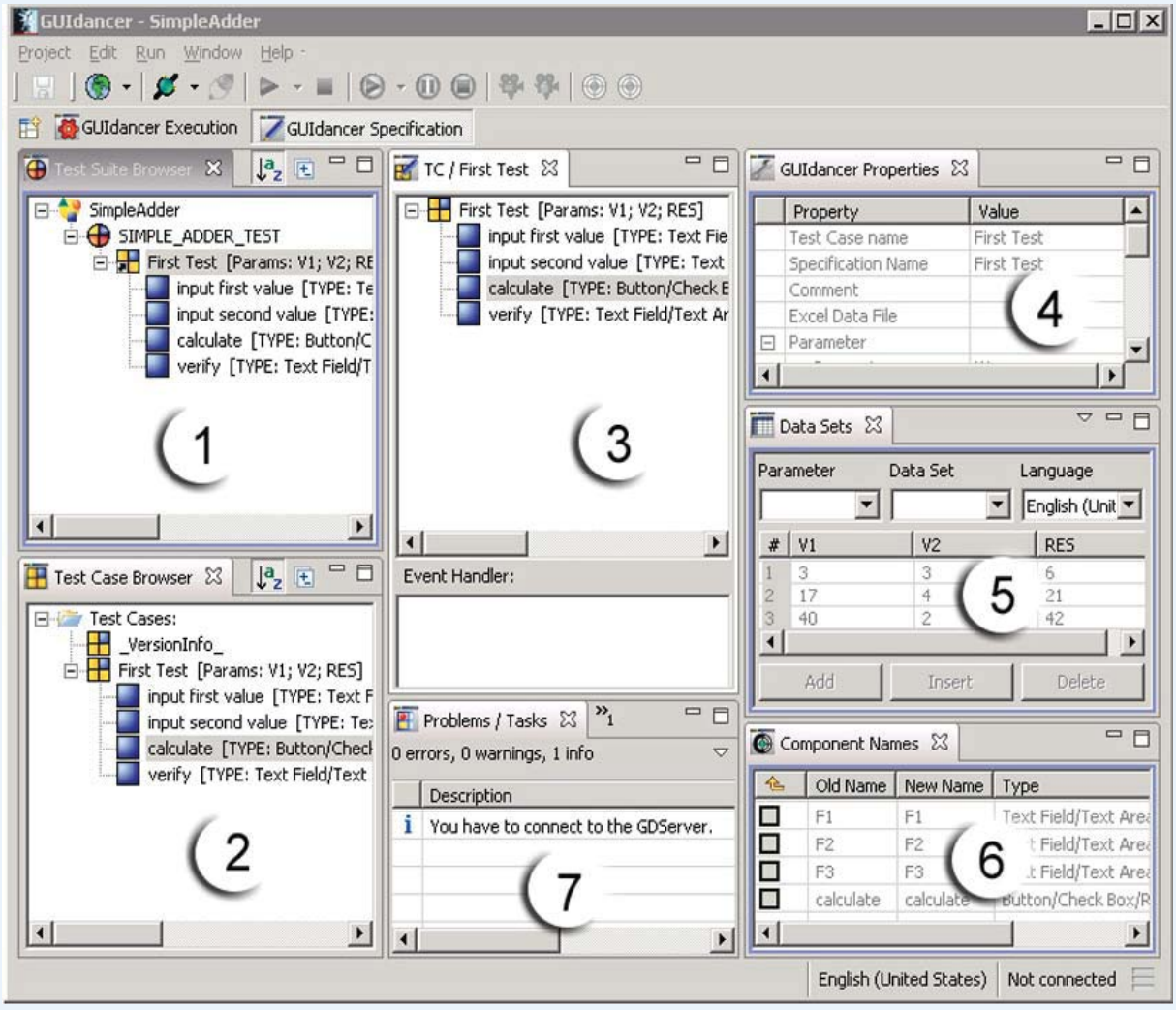


Fig. I: Specification Perspective

custom class loading. It also sports new test actions, such as restarting the AUT at any point in the test. GUI*dancer* also supports dynamic data: Test steps can retrieve and store any data displayed in the AUT and use this data later in the test. GUI*dancer* 1.1 also sports an improved user interface, with sharper, cleaner icons and visual indication as to which test cases are currently reused.

GUIdancer User Interface

There are four types of area in the GUI*dancer* user interface:

Perspectives: There are two perspectives, specification and execution, each with a different function. A perspective is a collection of views, editors and browsers on the screen. The specification perspective (Fig 01) provides browsers, editors and views to let you create tests. It contains:

- The Test Suite Browser (1)
- The Test Case Browser (2)
- The editor area (3)
- The Properties View (4)

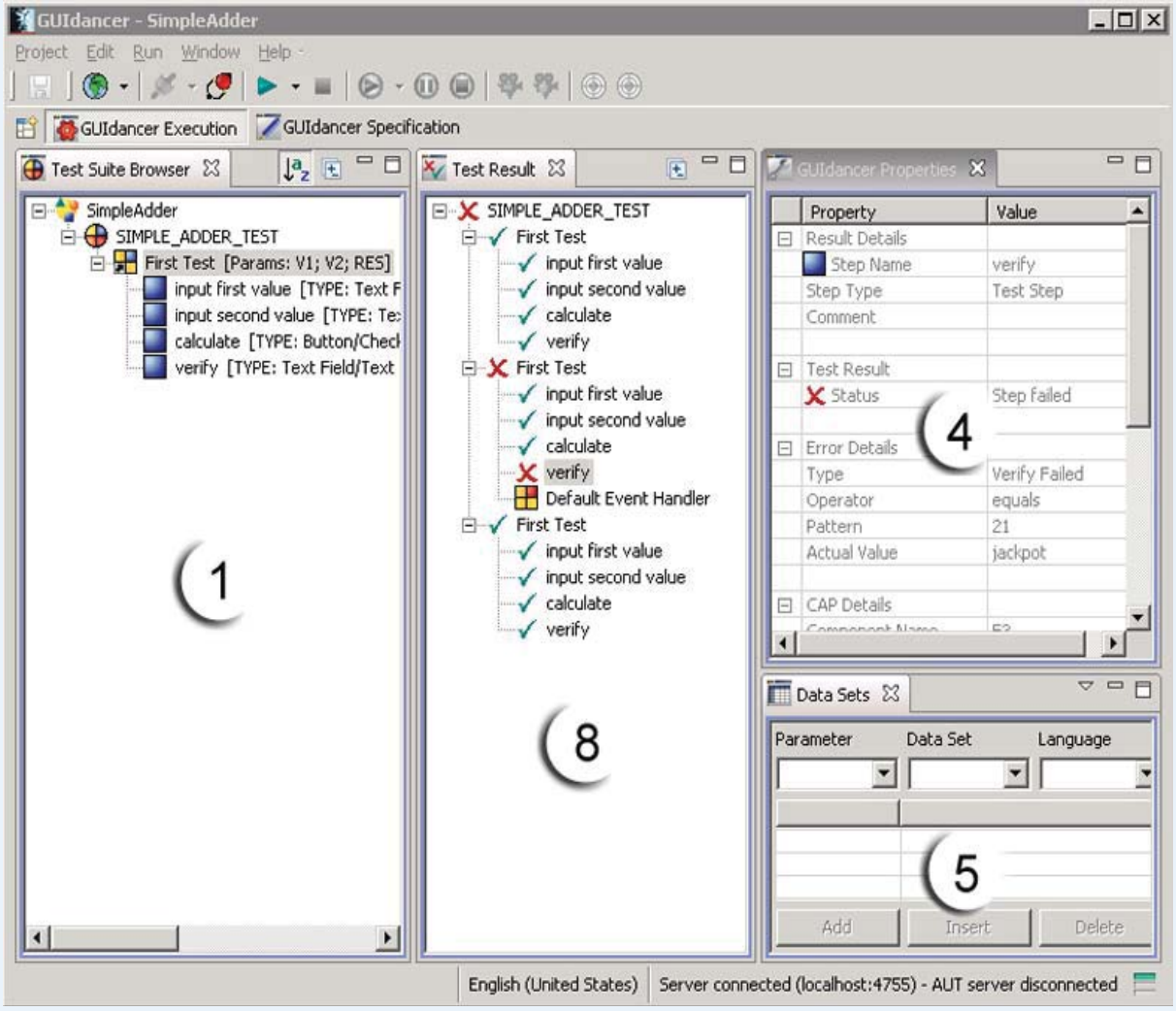


Fig. 2: Execution Perspective

- The Data Sets View (5)
- The Component Names View (6)
- The Problem View (7)
- The search result view (behind the Problem View)

The two browsers let you see your Test Cases (Test Case Browser) and your Test Suites (Test Suite Browser). If you double-click an item in one of these browsers, an editor for this item opens. Using the editor and its support views (numbers 4-6) you can edit the contents and properties of Test Suites, Test Cases and Test Steps. The content of the support view changes according to the currently selected item.

In the Execution perspective, you can see the following (Fig 02):

- Test Suite Browser (1)
- Test Result View (8)
- Properties View (4)
- Data Sets View (5)

You cannot edit in the execution perspective, but you can see your Test Suite and its results.

- **Browsers:** browsers let you see the Test Cases and Test Suites you have created in their hierarchical structures.
- **Editors:** editors let you modify, add and delete items in Test Cases and Test Suites. You can open editors by double-clicking on a node (e.g. Test Suite or Test Case) in a browser.
- **Views:** views show details about the currently selected item. If you single-click an item in a browser, the details in the views are read-only. If you select an item from within an editor, you can edit some of the values in the views.

GUI*dancer* Road Ahead

The GUI*dancer* team is still in the planning phase for the next major release. "As such, we can't offer a specific date for the next major release, other than that it will be within the upcoming year. We have several exciting features in the works for this release", says Hans-J. Brede. One of the features to look forward to is better support for test case metadata. This would allow each test case to contain such additional information as, for example, who created it or when it was last executed. This metadata could also include links to additional documentation relevant to the test case. The end result is a sharper overview of tests and the ability to more easily manage these tests.

"Also, following the same philosophy that we have embarked on, that reusable tests can be specified without programming, we are planning to widen our range of testable GUI toolkits. Possible candidates for support are SWT and web based applications", says Brede. He also adds that they are considering allowing testers to specify external scripts that can be run automatically before and after test execution. This is similar to the setUp and tearDown concept of JUnit, and allows the tester to, for example, add test data to a database right before a test is executed and then remove this data when the test is finished. This affords the tester better control over the test environment.

Conclusion

In the end analysis, GUI*dancer* uses a sophisticated, yet easy-to-use specification method which eliminates all programming from test creation for programs written with Java and Swing. A variety of powerful features allow the extensive reuse of editable test elements, resulting in tests that are quick to create, highly adaptable, and which require only minimal maintenance. In short, GUI*dancer* is flexible and stable, and all without programming. However, at 1,136.80 € for a named-user license (for one named user on one computer) and 4,500.00 € for a multiple-user license (for multiple users on one computer) you may also want to look at some of the other swing testing tools out there to crank out the raw stuff.

Resources & References

[1] http://www.bredex.de/en/guidancer/
[2] http://www.bredex.de/en/guidancer/downloads.htm