# Tool Radar: GUI*dancer*

(translated with permission from Javamagazin 3.2008)

Verion 2.0 of GUI*dancer*, BREDEX GmbH's Eclipse-based software, was released in December 2007. GUI*dancer*'s aim is to make the automated testing of Java (Swing, SWT/RCP) and HTML interfaces more efficient. In particular, testers can start to write GUI-tests from requirements, before an application under test (AUT) is even available. As well as ensuring that testing begins as early as possible, the code-free method of test creation means that tests grow alongside the software being developed.

As soon as GUI*dancer* has been installed with a valid license, test creation can begin. Test Steps, the smallest unit in GUI*dancer*, are created from interactive menus using three pieces of information: the GUI-component, the action and any parameters. A wide range of supported components, and actions to execute on them can be selected. Data (parameter values) are then entered for the action. Within seconds, the first specifications are ready. There is no additional automation process in GUI*dancer*, so these specifications are also the implementation for the test.

These Test Steps are used to form reusable modules (Test Cases). Reusability is a key concept of GUI*dancer*. Because tests are composed of repeated actions (clicking buttons, checking text, selecting values), a tester should take some time to think about what sort of steps will be needed often. By considering which modules to create and use, tests can be specified in a generic way so that a small amount of modular Test Cases can test as much as possible. GUI*dancer* supports reusability by allowing testers to leave the parameter values and GUI-component to be tested open until a later point. The data and the components to test can be added and edited later, and can even be given different values in different Test Cases.

The test specification and the AUT are joined via Object Mapping. Real GUI-components are collected from the AUT and are then easily mapped to the user-defined components from the Test Cases using drag and drop. The object map which is thus created is separate from the test flow. This reduces the maintenance effort – if changes need to be made, they can be made centrally. Using the abstract components in the test specification also improves the robustness and flexibility of tests. Abstract components can be mapped to various different real components in an AUT.

It's true that no programming is needed to create and maintain even large and complex tests. Testers can begin specifying tests without having to wait for any input from programmers (such as the implementation of keywords).

A good practice in testing, like in software development, is to think about which parts are reusable and can be modularised. GUI*dancer* offers strong support for modular testing – a new feature in version 2.0 is the reuse of whole projects so that previously written modules can be used as a test library.

GUI*dancer*'s client-server architecture means that tests are platform-independent and can run on various computers in the network. The client runs as a standalone application or as an Eclipse plugin.

A thirty-day demo license can be requested for a fully functional evaluation of GUI*dancer*. Example projects, tutorials, Test Case libraries and extensive documentation are all installed with the software. A floating license costs €3,900 for one GUI-toolkit (e.g. Swing) and €300 for each additional toolkit.