# Automated Graphical User Interface Test Tools

*A brief introductory description of automated GUI testing, what it is and the available tools that we evaluated*

Automated GUI Test Tools use image matching technology to manipulate GUI applications on the “System Under Test” (SUT). The tools "see" the screen of the SUT, and can recognize, for example, when a particular window, icon or text appears onscreen. Because the tool only looks at the "screen" of the SUT, the technology used in the application to be tested does not matter. The tool requires no direct ties to the SUT source code. This is advantageous because it allows the SUT to be tested as a black box without the need for special instrumentation.

Many of the available tools utilize a two-computer system consisting of a host controller machine, where scripts are authored and executed, and a SUT machine which runs a “Virtual Network Computing” (VNC) server. The test tool connects to the VNC server via a built-in viewer connection. Thus, the SUT could be any system that has a VNC server for it.

## Test Levels

*Describe what level test this effort is targeting.2/3/4/5?*

Automated GUI Test tools can support all levels of testing that require the ability to “record and playback” GUI interaction. However, they are most useful for system and component level tests where the SUT utilizes a GUI to execute system actions. Applicability to software test levels is summarized in the following table.

|  |  |  |
| --- | --- | --- |
| Test Level | Test Level  Description | Applicability to  Automated GUI Test Tools |
| 1 | Unit Testing – tests that verify the functionality of a specific section of source code at the function/class level. | Low – generally performed via code driven testing methods. May be utilized to automate GUI based unit test drivers. |
| 2 | Integration Testing – tests that seek to verify the interfaces between components against a software design. | Low – generally performed via code driven testing methods. May be utilized to automate GUI based component test drivers. |
| 3 | Component interface Testing – tests that validate the handling of data passed between various units, or subsystem components. | Moderate – useful for testing GUI driven components themselves or driving the system GUI to test other components. |
| 4 | System Testing – tests which verify that a fully integrated system meets its requirements. | High – very useful to automate GUI driven user tasks in order to exhaustively and sometimes regressively test system requirements. |
| 5 | Acceptance Testing – tests performed for or by the customer as part of the delivery hand off process. | Moderate – can be useful if the customer is involved in validating the automated tests, but generally this level of testing is performed manually |

Table 1 Applicability of Operator Action Based Automated Testing

## Tool Capabilities

*What’s needed? Briefly describe the things that are needed for automated testing at a high level with more details below (GUI manipulation, Image Capture/Comparison, Optical Character Recognition (OCR), etc.)*

In order to adequately test graphically oriented software, a tool must capture the actions and reactions of the operator and the user interface. This requires capabilities to capture screen images and operator actions, play them back on an automated or ad hoc basis, and validate the results.

Since the tool will be part of the project’s development system, it must support team collaboration and provide the ability to manage and version control tool artifacts (model files, script files, image files, etc.).

The following tool capabilities are needed:

* GUI Manipulation – ability to interact with the SUT GUI by generating mouse and keyboard events as necessary to manipulate the interface elements. Provide the means to control the timing and playback of sequential interactions.
* Image Capture/Comparison – ability to capture the state of the SUT GUI and scan it for the existence of desired element images (buttons, window titles, toolbar icons, etc.). Image content is subject to underlying system scaling and resolution differences. Image comparisons need to be tolerant to these differences.
* Optical Character Recognition – ability to scan captured SUT GUI images and extract textual content for comparison to expected values, or for the purpose of obtaining displayed calculated result values.
* Integrated Development Environment (IDE) – ability to create tests; start, pause, debug, and stop tests; and manipulate tool artifacts in a coordinated and developer friendly manner.
* Team Collaboration – support multiple user roles for test development, test execution, and test result management for issue tracking.
* Artifact Management – provide features to manage and version control the tool artifacts or support the use of external source management tools.