



# UserTest Quick Manual (FITTEST tool for automating the Rogue User)

Project no. 257574

**FITTEST**

**Future Internet Testing**

Specific Targeted Research Project

Information and Communication Technologies

Start date of project: 01-09-2010 Duration: 36 months

Organisation name of lead contractor for this document: UPVLC

Document Manager: Sebastian Bauersfeld

| **Project co-funded by the European Commission within the Seventh Framework** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

## 0. Requirements

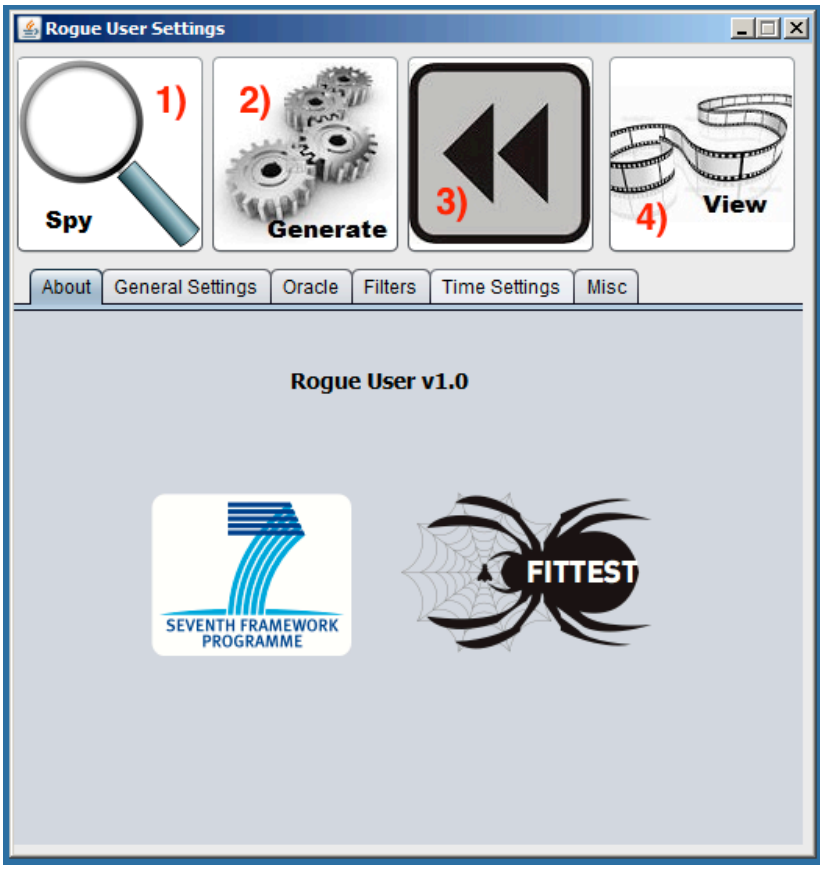
The current version of the Rogue User runs on Windows 7 64 bit. To ensure that the RU runs on your system you need to install the Java Development Kit (JDK) in version 1.6 or above.

## 1. Installing the Rogue User

The Rogue User comes in a file called `rogueuser.zip`, which contains all files necessary for its execution. Just unzip this file into a directory with write-access. There is no setup routine.

## 2. Starting the Rogue User

Within the main directory you will find several files, which are crucial for the execution of the RU. The most important one is `rogueuser.bat`. Double-click on it and the RU will start and display the main screen:




As shown in the picture above, the RU's main screen contains four buttons, which start the RU into its four main modes:

1. Start in **Spy-Mode**: This mode does not execute any actions. It will start the System under Test (SUT) and allows you to inspect the GUI. Simply use the mouse cursor to point on a widget and the Rogue User will display everything it knows about it. The Spy-Mode will also visualize the set of actions that the Rogue User recognizes, so that you can see which ones will be executed during a test.

2. Start in **Generation-Mode**: This mode will start the SUT and execute a full test on the SUT.

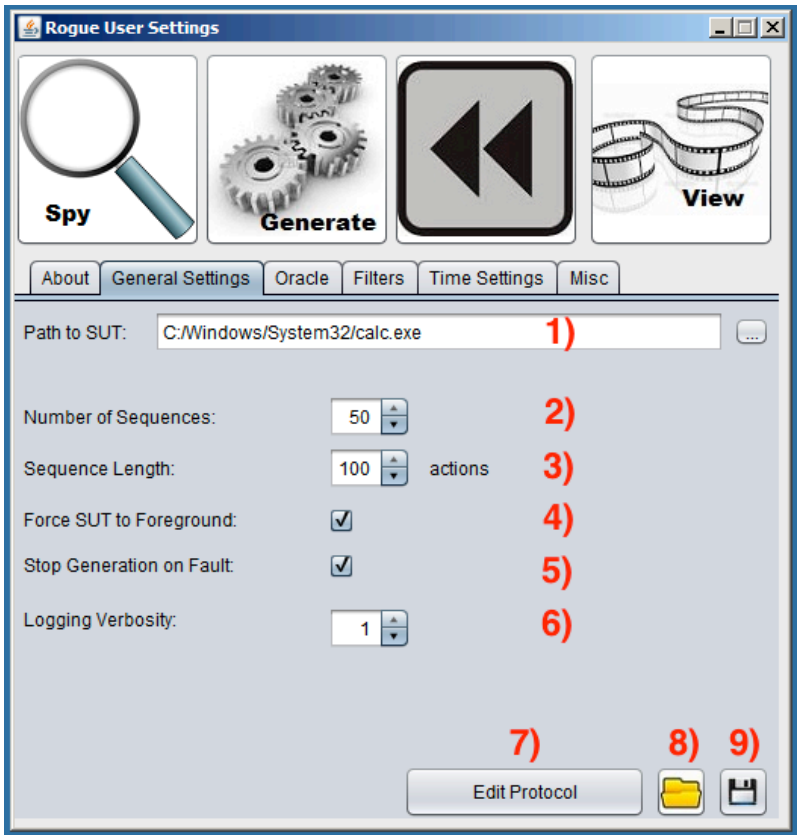
3. Start in **Replay-Mode**: This mode replays a previously recorded sequence. The Rogue User will ask you for the sequence to replay.

4. Start in **View-Mode**: The View-Mode allows you to inspect all steps of a previously recorded sequence. Contrary to the Replay-Mode, it will not execute any actions, but only show you the screenshots that were recorded during sequence generation. This is ideal if a sequence turns out not to be reproducible.

# 3. Setting up your tests

## 3.1 General Settings Tab

The Screenshot below shows the tab with the most important settings for the RU.



1. Path to the SUT: Pick the executable of the SUT or insert a custom command line.

2. Number of sequences to generate.

3. Sequence length: After having executed the given amount of actions, the RU will stop the SUT and proceed with the next sequence.

4. Force the SUT to the foreground: During test generation, the SUT's windows might get minimized or other processes might block its GUI. If you check this option, the RU will force the SUT to the foreground.

5. Stop sequence generation on fault: If the RU detects an error, it will immediately stop sequence generation.

6. Logging verbosity: The higher the value, the more information will be written to the RU's log-file. The log-files of each run can be found in the RU's output directory and contain information about the actions that were executed, the faults that were found and potential problems that occurred during the test.

7. Edit Rogue User Protocol: By clicking this button you will open the Protocol Editor for the RU. As we will see later on in this Manual, this editor allows you to override and extend the RU's basic functionality in order to implement complex action sets and sophisticated oracles.

8. Load settings file. If you have a specific setup that you saved into a file, you can load it here. This is ideal for switching between different settings for the same SUT or between the settings of different SUTs.

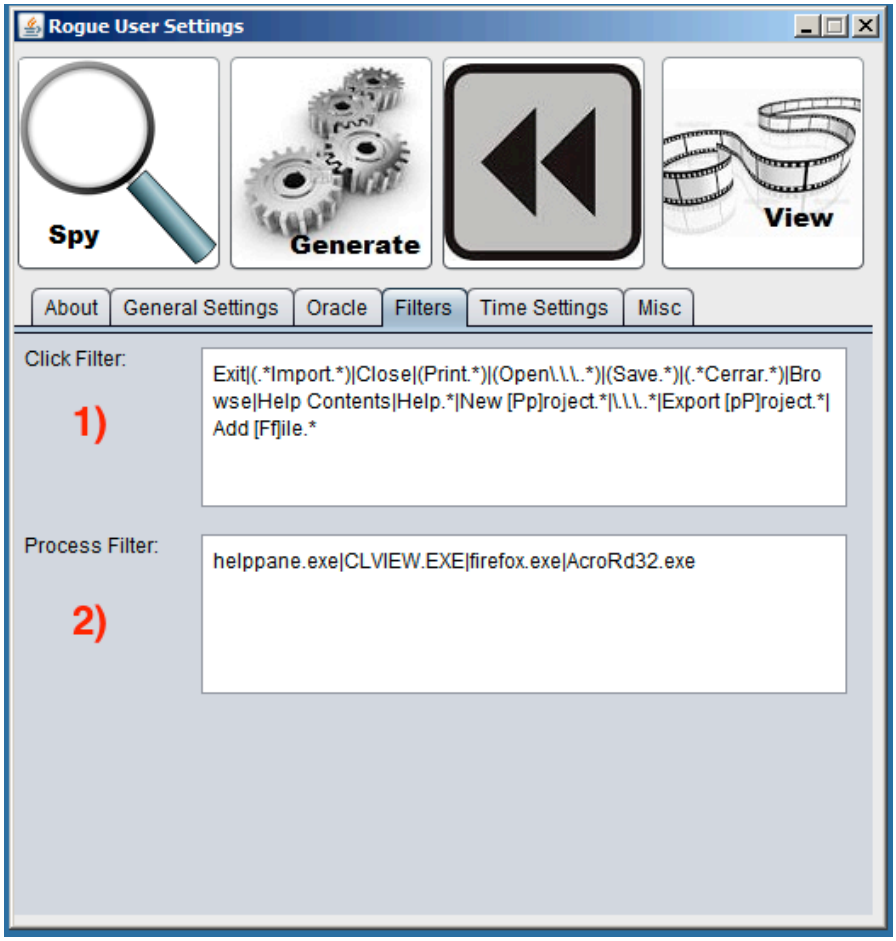
9. Save the current settings to a file.

Remark: The RU automatically saves all setting changes you make. Thus you do not explicitly have to save your settings to a file every time you make an adjustment.

### 3.2. Filter Tab

In this tab you will be able to do the following two things:

1. Tell the RU which actions not to execute (because they might be dangerous or undesirable), and
2. Tell the RU which processes to kill during test generation.



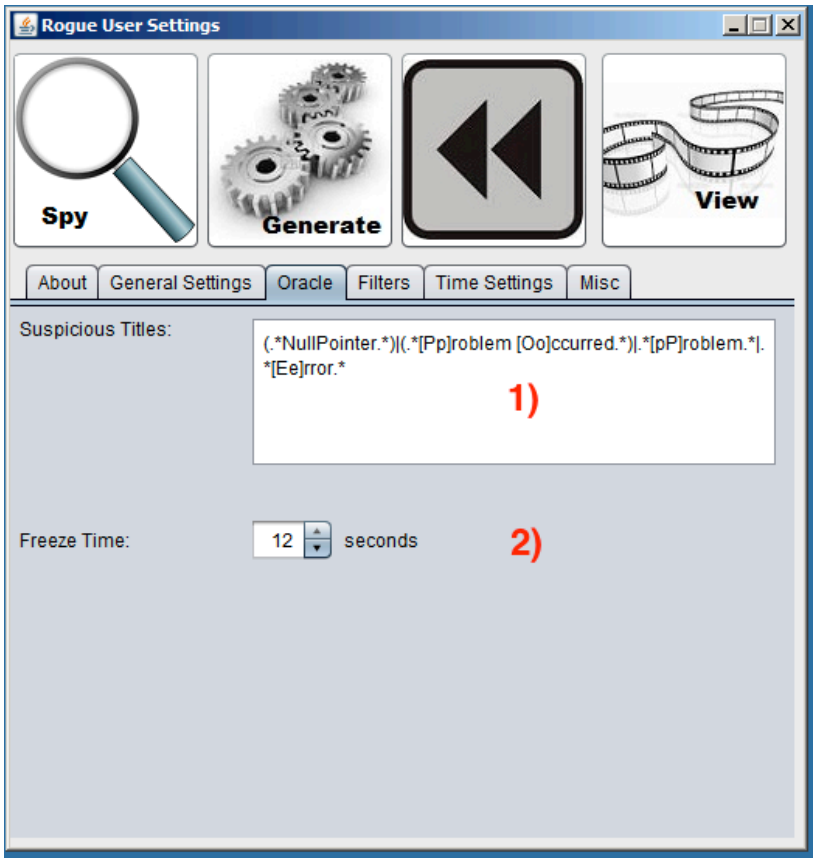

1. Click-filter: Certain actions that the Rogue Users wants to execute might be dangerous or undesirable, such as printing out documents, creating, moving or deleting files. The Rogue User will not execute clicks on any widget whose title matches the given regular expression. To see whether or not your expression works, simply start the RU in Spy-Mode, which will visualize the detected actions.

2. Processes to kill: Some SUTs start other processes during test sequence generation. These might popup in the foreground and block the SUTs GUI. They might also consume excessive memory, etc. The RU will kill any process whose name matches the given regular expression.

### 3.3. Specifying simple Oracles

In order to detect faults, you need to tell the RU what to look for. In the “Oracle” tab you can specify a simple oracle, which analyzes each state of the GUI and reports errors.

The screenshot below shows the Rogue User's oracle settings:

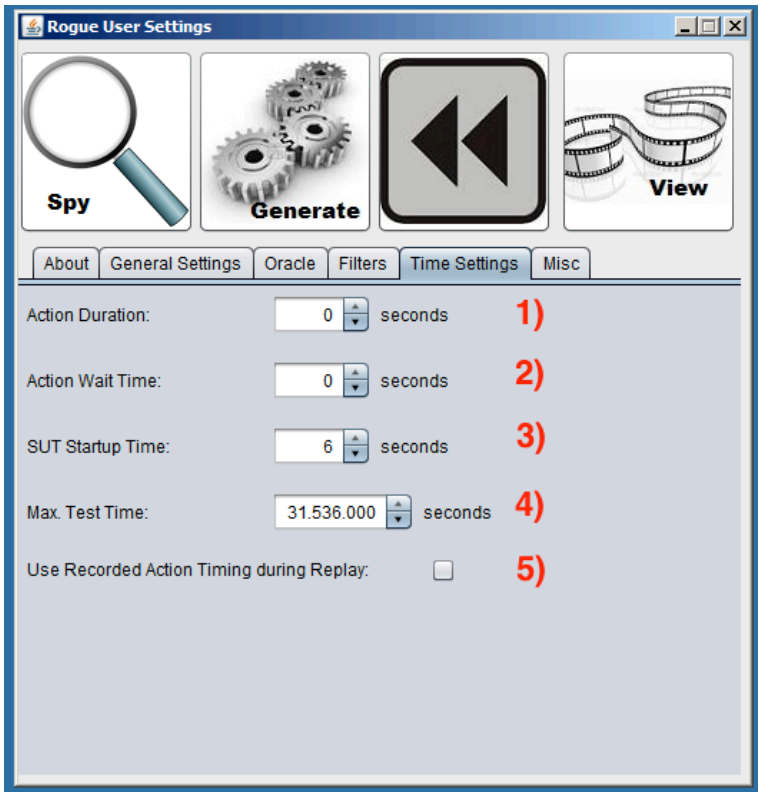



1. **Suspicious Titles**: In this text box you can enter a regular expression that describes those messages that you consider to be related to possible errors. The RU will apply this expression to each title of each widget on the screen. If it matches any widget's title, the RU will report an error and save the sequence for later inspection. For example: Imagine you are looking for a critical message box with the title "A NullPointerException Exception has been thrown". You could simply add the expression ".*NullPointerException.*" which will match any title that contains the word "NullPointerException" (the ".*" are placeholders for arbitrary characters). To learn more about Regular Expressions see http://en.wikipedia.org/wiki/Regular_Expression

2. **Freeze Time**: The RU is able to detect crashes automatically, because it realizes when the SUT is not running anymore. However, if the SUT does not really crash, but just freezes (is unresponsive) for a long time, then the RU does not know whether it is just carrying out heavy computations or hangs. If the SUT is unresponsive for more than the given amount of seconds, the RU will consider it to be crashed and mark the current sequence as erroneous.

### 3.4. Time Settings Tab

The time Settings tabs, allows the user to configure the following:




1. **Action Duration**: The higher this value, the longer the execution of actions will take. Mouse movements and typing become slower, so that it is easier to follow what the Rogue User is doing. This can be useful during Replay-Mode, in order to replay a recorded sequence with less speed to better understand a fault.

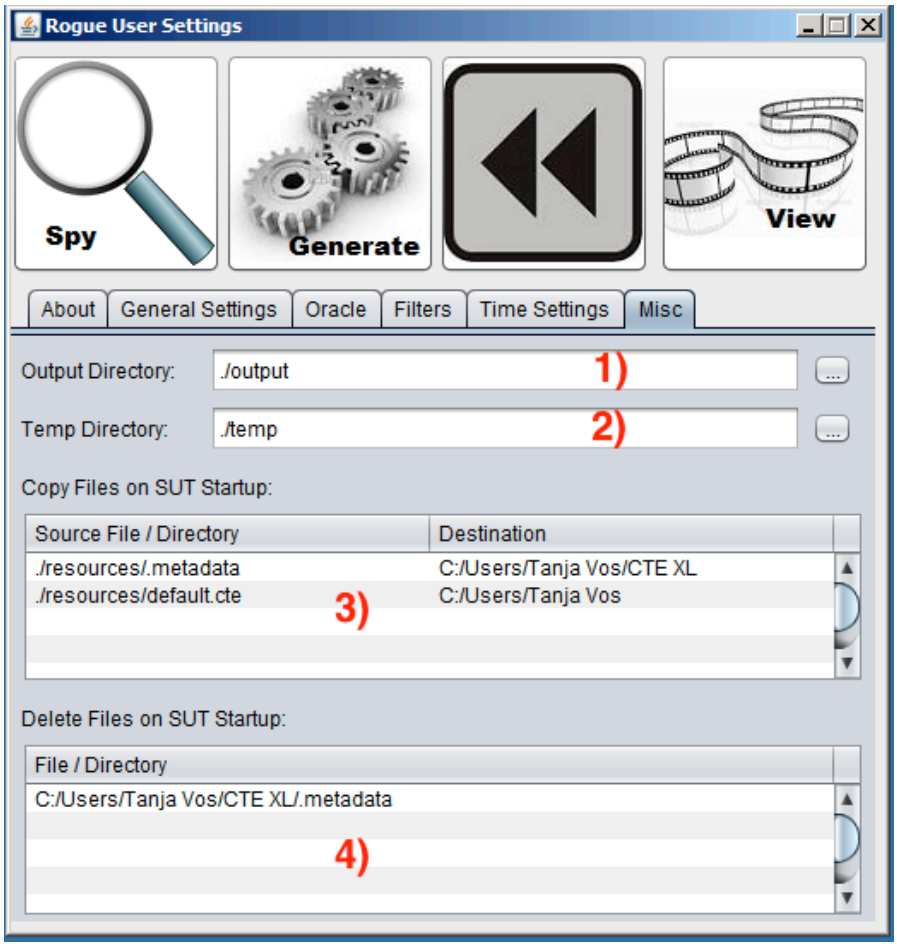
2. **Time to wait after execution of an action**: This is the time that the Rogue User pauses after having executed an action in Generation-Mode. Usually, this value is set to 0. However, sometimes it can make sense to give the GUI of the SUT more time to react, before executing the next action. If this value is set to a value > 0, it can greatly enhance reproducibility of sequences at the expense of longer testing times.

3. **SUT startup time**: This is the time that the Rogue User waits for the SUT to load. Large and complex SUTs might need more time than small ones. Only after this time has expired, the Rogue User will start sequence generation.

4. **Maximum test time** (seconds): The RU will cease to generate any sequences after this time has elapsed. This is useful for specifying a test time out, e.g. “1 hour”, “one day”, “one week”.

5. **Use Recorded Action Timing during Replay**: This option only affects Replay-Mode. If checked, the RU will use the action duration and action wait time that was used during sequence generation. If you uncheck the option, you can specify your own values.

## 3.5. Miscellaneous Tab




This tab contains some settings that are less frequently used.

1. Output directory: This determines the directory where the Rogue user outputs recorded sequences, log-files and state snapshots.

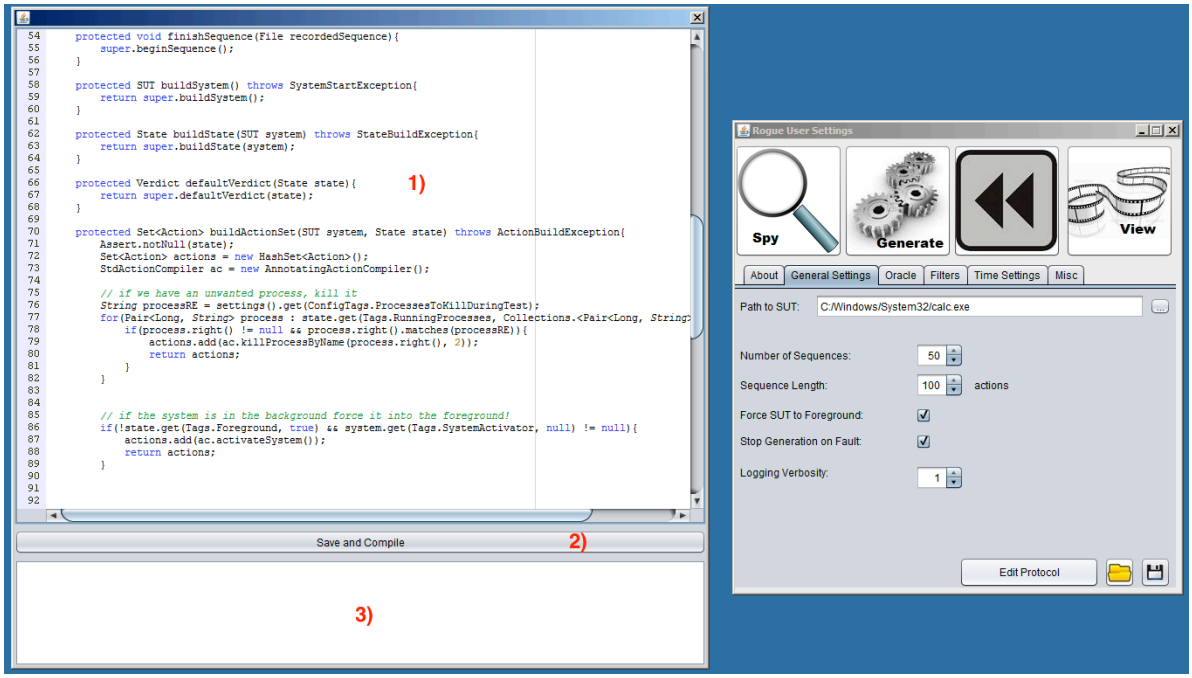
2. Temporary Directory: The RU will use this directory to store temporary files during the execution of sequences.

3. Files to copy before SUT start. When you start the SUT, sometimes it can be useful to restore certain configuration files to their default, so that the SUT always starts in the same state. Therefore you can define pairs of paths (copy from / to). The RU will copy each specified file from the given source location to the given destination. Simply click the text-area and a file dialog will pop up.

4. Files to delete before SUT start: Certain SUTs generate configuration files, temporary files and files that save the system's state. This might be problematic during sequence replay, when you want a system to always start in the same state. Therefore, you can specify these files, to be deleted before

the SUT gets started. If you click the text-area, a file dialog will pop up which allows selecting files and directories to be deleted.

## 3.6. Specifying Advanced Oracles




All the settings presented so far help you to setup tests for small and less complicated SUTs. However, at some point you might feel the need to implement more complex actions or setup a more sophisticated oracle. The RU allows you to edit its protocol, i.e. the source code that determines its behavior. The above screenshot shows the Protocol Editor, which you can enter by navigating to the "General Settings" tab and clicking the "Edit Protocol" button. It shows:

1. The Source Code of the Protocol

2. The "Save and Compile" Button, which compiles the protocol and saves it to be used during the next test.

3. The Error Console, which informs you about potential errors during compilation.

The source code used in the protocol is plain Java.

# 4. Running the tool

## 4.1 Action Visualization in Spy-Mode

When in Spy-Mode, the RU displays the detected actions (Shift + 1 to toggle visualization). Each action type has a specific appearance, as described in the following table:

| | |
|---|---|
| Green dot | Left click |
| Yellow circle | Right click |
| Red circle | Left double click |
| Blue Arrow | Drag & Drop Operation |
| Blue Text | Click into text field and type the displayed text |

## 4.2 Test sequence generation and execution

Before running a complete test in the 'Generate'-Mode you have to pay attention to a few things:

1. Define the set of actions that you want the Rogue User to execute on your SUT: Although, you want to setup a thorough test that fully stresses your SUT, you might want to spare out certain actions, such as printing documents, terminating the SUT (which might be detected as a crash by the oracle) or minimizing it etc. You might also want to only test a specific subset of all actions because you suspect that faults in specific dialogs of the GUI are more likely to be triggered. You can use the *ClickFilter* in the *Filters* tab to exclude actions on particular widgets (see section 3.2 for further details).
2. Define startup time, action duration or wait time after actions. Those time settings might be important for your test, since they influence the reproducibility of sequences. If you start sequence generation to early (before the SUT has been fully loaded) or execute actions too fast / do not give the GUI enough time to react, your generated sequences will still find faults. However, these faults might be more exotic (a human user might not be able to trigger them at all) and are usually very hard to reproduce since the timing aspect plays an important factor. (See section 3.4 to learn how to set these settings)
3. Make sure that the SUT always starts in the same initial state. This is very important to guarantee reproducibility. The large majority of SUTs remembers specific settings or saves the position of its windows as they have been during the last session. If you do not restore the SUT's settings to their defaults, a previously recorded sequence might not be replayed properly, simply because the SUT starts in a different states during sequence generation and sequence

replay (e.g. starts already with the last edited document opened). You can use the settings in the *misc* tab to delete or restore the SUT's settings files.

4. Define your oracle: The Rogue User automatically detects certain faults, such as crashes. However, you might want to look for critical error messages or low responsiveness. Sections 3.3 and 3.6 describe how to set up oracles that help you find certain types of faults.
5. Stopping criteria: Depending on how long you want the Rogue User to run, you have to adjust your stopping criteria. You might want to runt he RU for 5 hours (then you can use the Maximum Time setting in the *General Settings* tab) or have it generate 1000 sequences, etc.
6. It can be difficult to verify whether the RU will do what you told him to and thus you have to test your settings. Therefore you may use the "View" mode in order to inspect the set of actions that the RU will execute later on (hit Shift + 1 to see the generated actions) or you can switch to the "Generate-Debug" mode using Shift + Left Arrow / Right Arrow.

Generally, it will take some time until you have everything set up for a full test. Make sure that you experiment with your settings in the View Mode and look for potential problems that might arise. Take a look at the appendix, which lists helpful keyboard shortcuts (e.g. stopping test generation etc.) that you can use once a test is running.

## 5. Viewing and replaying the results

During test generation in the *Generate Mode*, the RU will save erroneous / suspicious sequences into the output directory. Once the RU has finished a test, you might want to inspect those generated sequences to better understand faults or problems. There are basically two ways to do that:

1. Replay the sequence: Of course this is the preferred way, since it shows you directly what the RU has generated. When you start the RU you can click on the *Replay Mode* Button and the RU will ask you for the file to replay. Just select a sequence file of your choice and hit ok and the RU will try to replay it. You can even slow down the sequence in case it was recorded at a high speed (just lower the value for action duration). However, certain sequences might not be reproducible, because the SUT is not in the correct starting state (have you considered deleting / restoring settings files?) or the sequence was recorded to fast (decrease the value of action duration and action wait time during sequence generation). In that case you can…
2. View Screenshots of the Generated Sequence: Just hit the button for the *View Mode* (in the RU's start screen). Again the RU will ask you for a file of a recorded sequence. Browse to the location of the file and hit ok. Now you will enter the RU's *View Mode*, which will show you screenshots of each state that the SUT has been in during sequence generation. Just keep hitting "Next" to proceed to the next step. The *View Mode* will always work and it allows you to see what happened on the screen during sequence generation, which can be very helpful in case you are unable to replay a previously recorded sequence.

# APPENDIX

## Keyboard Shortcuts

Within the various modes, the RU accepts several shortcuts:

| Shortcut | Effect | Modes |
|---|---|---|
| Shift + Arrow Down | Quit Monkey | Generate, Spy, Replay |
| Shift + Arrow Up | Save snapshot of current state to output directory | Generate, Spy, Replay |
| Shift + Arrow Left / Arrow Right | Switch Mode | Generate, Spy, Replay |
| Shift + 1 | Toggle Action Visualization | Spy |
| Shift + 2 | Toggle visualization of Widget under Cursor | Spy |
| Shift + 3 | Toggle widget information | Spy |
| | | |

## Directories

| | |
|---|---|
| ./temp | Temporary files such as the currently recorded sequence |
| ./output | The RU outputs log files, state snapshots and generated sequences into this directory |
| ./resources | If you have files for your SUT that need to be restored through copying, you can put them in here. |