

Running Automated Tests with Ranorex and imbus TestBench

1 About imbus TestBench

imbus TestBench is the working environment for test teams of all sizes, and provides everything that is required of a modern test tool like test planning, test design, test automation, test execution and reporting.

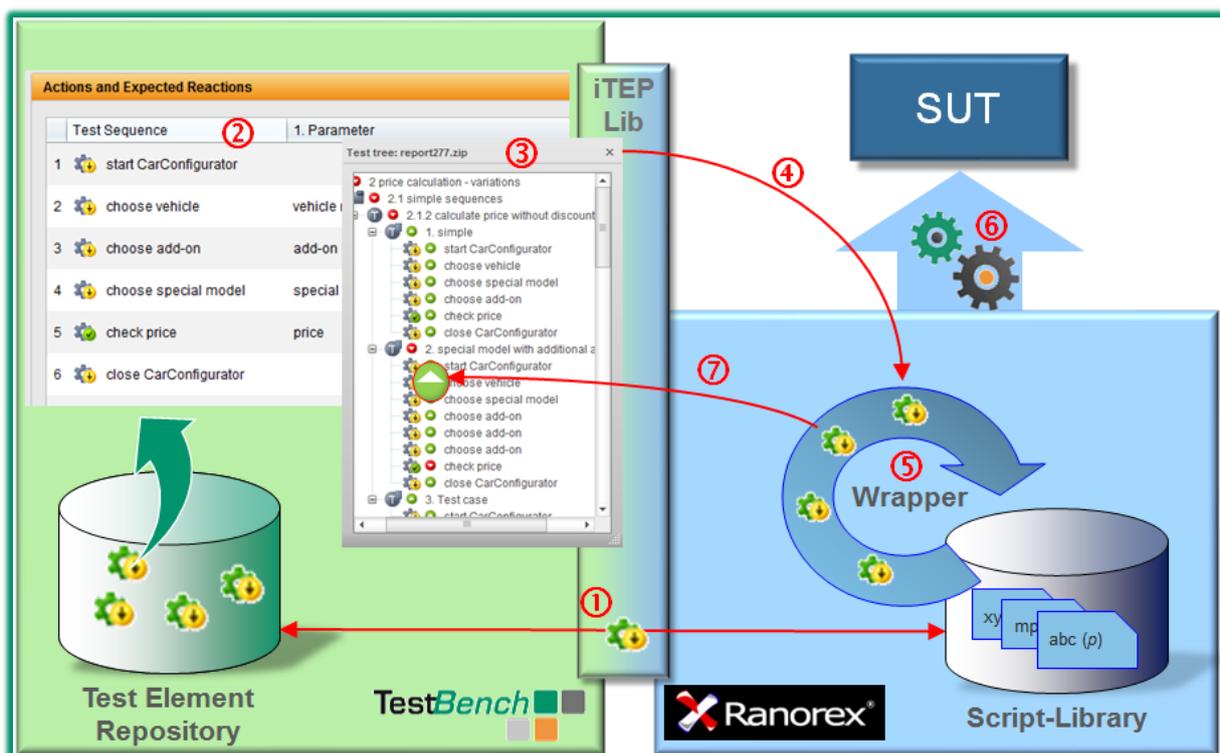
TestBench can be integrated seamlessly into your existing tool chain. Its key benefits include intuitive operation, high-performance test functions and a practical model for rights and roles, besides which it also supports structured testing according to the ISTQB® standard. It provides complete control and transparency in the content, status, progress and results of your software tests - at all times and across all development locations and all releases/versions of your software.

Different approaches for writing test specification are available. From the simplest test description, with the aid of a formatted test field, up to reusable text modules and the interaction method, the TestBench provides every test designer with the right specification method. At any time, it is possible to switch methods or to combine various methods with each other. For automated testing we are going to use the interaction method which is the combination of keyword- and data-driven testing.

2 Objective and Achievement

Aim of the following scenario it to use most benefit of Ranorex as well as TestBench. The purpose of TestBench as test management & design tool is the handling of the test activities, providing required data, managing of test suites and providing all test related information for reporting. Ranorex is responsible for all interactions with the system under test (SUT) like object mapping and executing instructions.

For this approach the interaction method (a combination of keyword- and data-driven testing) is used. By using keyword-driven testing, the test specification consists of test steps with a unique meaning. By reading the name of a keyword (test step), a tester or test robot knows exactly what to do. Those keywords and all related test data are stored in a repository within TestBench.



Running Automated Tests with Ranorex and imbus TestBench



The workflow steps from the picture above are (each step is explained in more detail afterwards):

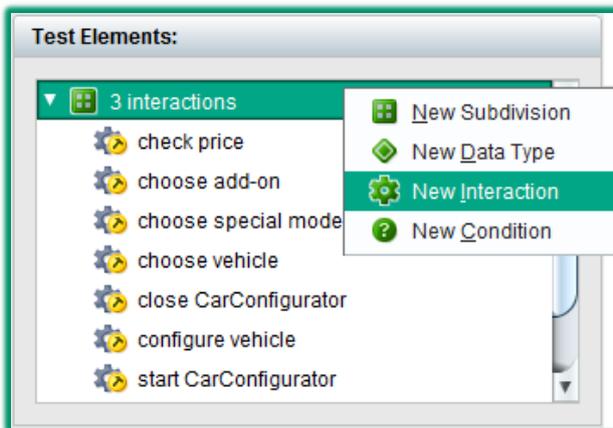
1. Each interaction (test step) in TestBench has a corresponding component (procedure) in Ranorex
2. Tests are put together with reusable interactions of the TestBench library (logical description) and extended with test data (concrete test case)
3. The planning of the test execution (test management) takes place in the TestBench – what should be tested and which data should be used
4. The test execution is started out of TestBench and the imbus Test Execution Plugin (iTEP) generates a “test screenplay”
5. The wrapper (a script in the test automation tool) fetches the corresponding scripts from the Ranorex library in accordance with the instructions of the “test screenplay”
6. Ranorex executes the scripts on the SUT and checks, if the execution was successful or not
7. The results (and comments) of each individual step are returned to TestBench

2.1 Components in TestBench and Ranorex

Each interaction in TestBench triggers an action method (procedure) in Ranorex. For doing so, a mapping between both systems is required. The easiest way to establish such a link is just using the same naming convention.

2.1.1 Representation in TestBench

Interactions and test data (called data-type) are managed in a central repository and used in a test case via drag & drop.



New interactions can be added easily by the context menu like in the screenshot above.

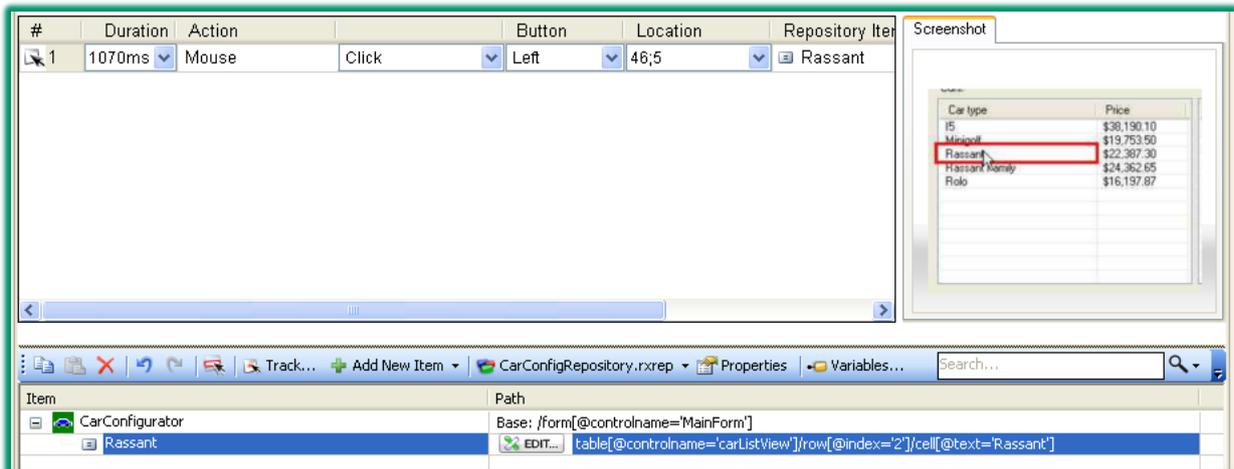
Running Automated Tests with Ranorex and imbus TestBench

2.1.2 Representation in Ranorex

Each atomic TestBench interaction needs a method to be called by Ranorex. The fastest way is by



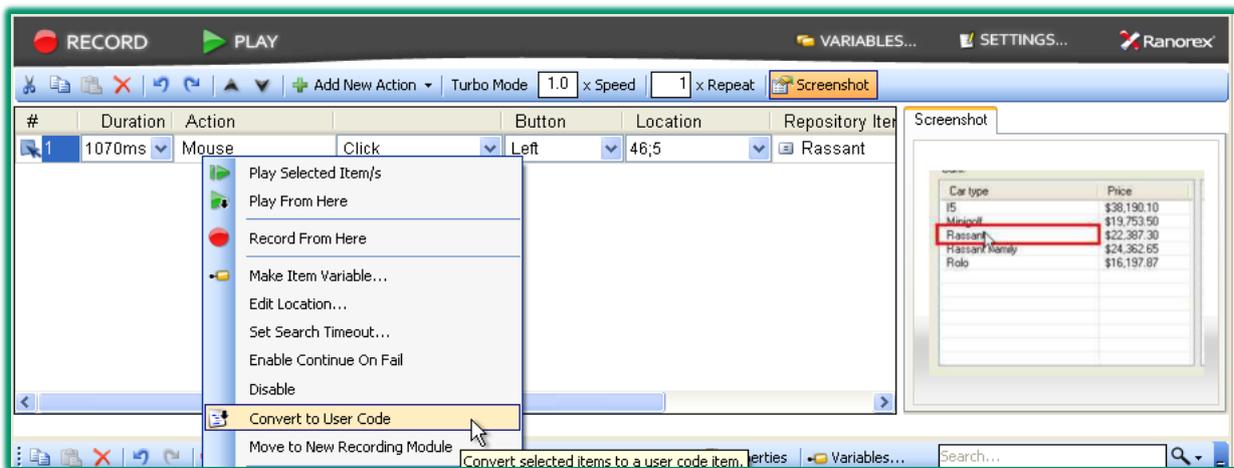
using “Record” to get all GUI objects and actions:



#	Duration	Action	Button	Location	Repository Item	
1	1070ms	Mouse	Click	Left	46;5	Rassant

Car type	Price
i5	\$38,190.10
MiniCoil	\$19,753.50
Rassant	\$22,387.30
Rassant Money	\$24,362.65
Role	\$16,197.87

By converting the recording into “User Code” it gets available by the wrapper:



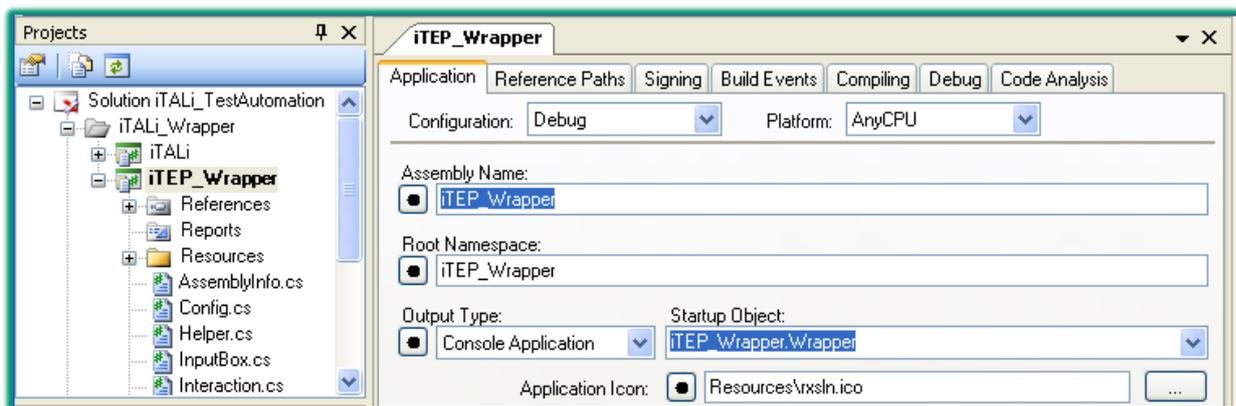
Via “View User Code” the code can be analyzed and adapted. This is required, when a test step is going to be executed with different test data. At the example above it should be possible, to replace a specific data (e.g. “Rassant”) from the recording with the data provided by the wrapper. Therefore a parameter is going to be added to the generated User Code.

```
public void choose_vehicle (String vehicle) ...
```

Running Automated Tests with Ranorex and imbus TestBench

2.1.3 Linking the API's

The link between TestBench and Ranorex is established via iTEP wrapper (see chapter 2.5) and is represented as a project within Ranorex. This project is available from imbus and can be customized to the specific needs of the project. As the wrapper is the starting point of test automation it has to be the StartUp Project too:



2.2 Defining a Test Case

Now it is possible to put together a test sequence via drag & drop by using the keywords ...

Actions and Expected Reactions			
	Test Sequence	1. Parameter	Comment
1	start CarConfigurator		
2	configure vehicle	configuration	
3	check price	configuration.price	
4	close CarConfigurator		

... and adding different test data:

Test Cases	
	configuration
1	simple
2	Rassant Jazz Variation 1
3	Rassant Family Variation 1

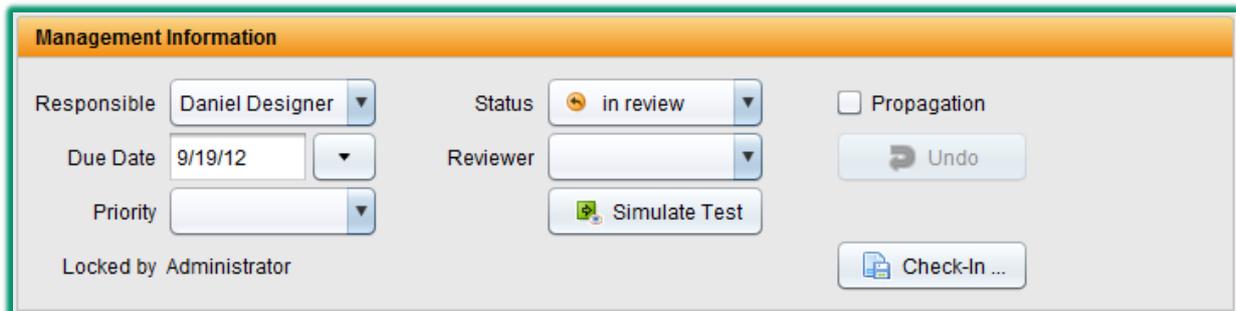
(At this example compound data types and interactions are used already. This sophisticated feature enables the usage of e.g. business objects instead of flat data values only. A *configuration* includes all

Running Automated Tests with Ranorex and imbus TestBench

required data about a car like its type, special model, add-on's and of course its specific price. The test step *configure vehicle* itself contains more detailed test steps like shown in the preview at chapter 2.4.)

2.3 Planning of the Test Execution

With TestBench, it's easy to create detailed test plans. Important planning and status (such as responsible person, priority or due date attributes) can be evaluated quickly, and any classification level, as required.



Management Information

Responsible: Daniel Designer | Status: in review | Propagation

Due Date: 9/19/12 | Reviewer: | Undo

Priority: | Simulate Test

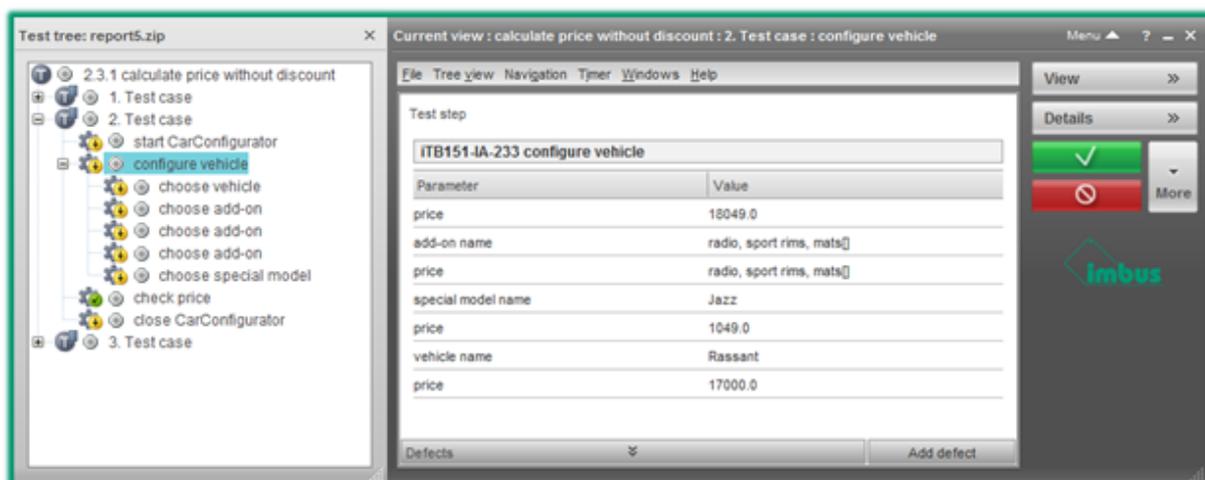
Locked by Administrator | Check-In ...

With user-defined elements like *Tags* and *User Defined Fields*, you are flexible and have the chance to depict your individual test process.

All those information can be used for managing test suites (e.g. regression test suite) very fast and easy.

2.4 Starting the Test Execution

The test execution can be conveniently started by using the iTEP Export which converts the internal information into a "test screenplay" with step-by-step instructions. The screenshot below illustrates the single test steps with their data.



Test tree: report5.zip | Current view: calculate price without discount : 2. Test case : configure vehicle

Test step: ITB151-4A-233 configure vehicle

Parameter	Value
price	18049.0
add-on name	radio, sport rims, mats[]
price	radio, sport rims, mats[]
special model name	Jazz
price	1049.0
vehicle name	Rasant
price	17000.0

Defects: Add defect

2.5 Wrapper

The wrapper connects the API's of TestBench (represented by iTEP.dll) and Ranorex. It handles the test screenplay, instructs Ranorex to execute test steps with specific data and saves the result before storing them in TestBench for reporting and archiving.

Being more specific, it reads the screenplay and calls for each interaction the corresponding method (see 2.6) of Ranorex with the test data as parameter. The outcome like SUCCESS or FAILURE (see 2.6) of each method will be stored in the execution log and can also be extended for example by a

Running Automated Tests with Ranorex and imbus TestBench

comment why the test step failed. In case of an error it is possible to define the behavior of the screenplay, like continuing with next test step or jump to the next test case.

An example implementation can be requested at testbench@imbus.de.

2.6 Execution of Test Steps and Comparison of Results

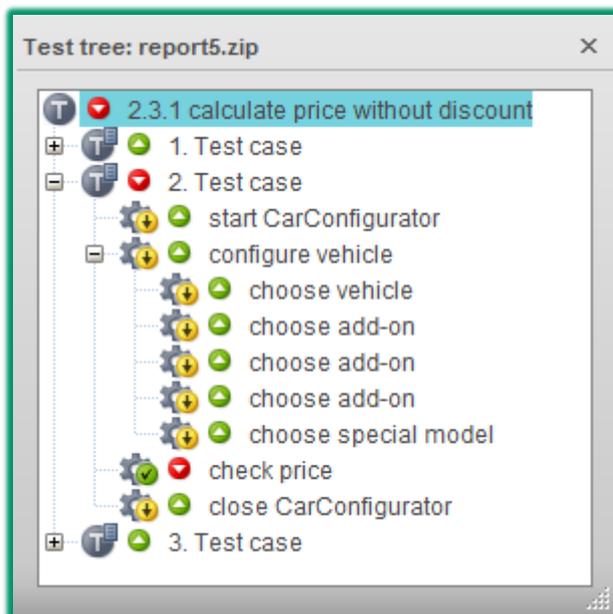


Like the most other test projects within Ranorex the automated test starts by clicking on

Now Ranorex works through the “test screenplay” and stores the test verdict of each single test step (and also other information if needed).

2.7 Return of Test Results

The result of the automated test can be checked by using iTORX.



If everything is fine, the test results are going to be imported into TestBench, versioned and ready for reporting.

3 Conclusion

The imbus TestBench helps you to keep an overview of the testing tasks, even in large, automated test suites. It enables the test designer to write test specification for automated testing without any programming knowledge as a result by just using existing keywords.

Using keywords reduces the complexity of the test automation because simple interactions can be used instead of implementing whole test procedures. Those small code snippets are much easier to maintain than large test sequences and are more flexible because they do not contain any hardcoded test data.

TestBench is the single point of change in case of modifying existing test cases or data without needing to adapt the test automation implementation.

Every specified test can also be executed manually. As either automated test results out of Ranorex as well as test results of manual testing is stored in TestBench, it is possible to have an overall overview of the test process. All information is stored in one data source which also takes care of versioning the information and their context.

With interfaces of requirement and defect management systems it supports the whole testing process. By linking requirements, it automatically provides the full trace starting from the requirement to test

■ Running Automated Tests with Ranorex and imbus TestBench

specification, test results and even bugs (if found). Therefore all testing related information are available for efficient test management and individual reporting.