A TESTRUNNER APPLICATION FOR OOREXXUNIT

Rainer Kegel

February, 2007

Table of Contents

Table of Figures

Table of Code

1 Introduction

Software Engineering: “The computer science discipline concerned with developing large applications. Software engineering covers not only the technical aspects of building software systems, but also management issues, such as directing programming teams, scheduling and budgeting.” [cf. W3Se07a]

In other words software engineering is the process of manufacturing software systems. [cf. W3Se07b, HaNe05]

A software system consists of

- the executable computer code (e.g. a word processor) and
- the supporting documents needed to manufacture, use and maintain the code (e.g. user manuals, requirement documents and designs).

As software engineering and especially software testing is getting more and more important – because software systems are to a great extent becoming larger, more complex, and vital (i.e. software for hospitals) – the software reliability is increasingly important and the staff resources particularly in quality management are short, the testing of software and especially the automation of testing is of particular importance. [cf. W3Se07b, HaNe05]

Because of the fact that programming languages are the fundamental building blocks of every software, there need to be especially accurate testing tools (i.e. if the programming language does not work properly, the created software will not work in the desired way as well).

Open Object RexxUnit (ooRexxUnit) is such a testing tool (framework) for the programming language Open Object Rexx (**O**pen **O**bject **Re**structured E**x**tended E**x**ecutor).

To facilitate the handling of so called testing frameworks, there are test runners (text-based or graphical) created for each framework and each programming language.

In this Master Thesis the new graphical test runner, developed for the Open Object RexxUnit framework, called TeRA (**Te**st**R**unner **A**pplication for ooRexxUnit) will be described in detail.

TeRA is used (in combination with the ooRexxUnit framework) to test all parts of the ooRexx programming language, with the aim to eliminate errors and failures and improve all actions of the programming language.

An example to visualize the meaning of TeRA:

Assumption: The array class is changed for any reason. With TeRA it is possible to test this class via ooRexxUnit and the corresponding test unit. After the changes the developers need to know whether the class still works the same, or not. For this purpose the developers can use the testing framework and the test runner. When the test run ends, it is possible to access the log files, where the whole information about the test run is stored. If the class does not work the same this information (log file) is necessary to easily identify the occurred problems. It is also possible to access the log files at a later date. This is important to reconstruct former changes in the tested software. Furthermore there is a function which provides the possibility to compare two test runs. This enables developers to identify changes, improvements and worsenings easily.

The project was defined as follows:

- Create a graphical user interface (GUI) to facilitate the execution of tests.
- Filter out the important information and store it (log files).
- Use browsers to display the results.
- Create clear structures (tree structure) for the arrangement of test units to facilitate the handling within the user interface.
- Find a possibility to compare the generated log files.

In the course of the project, there were some definitions added:

- Highlight failures and errors in the log files in terms of colour to raise clarity.
- Highlight changes in the compared log files to raise clarity.

- Use XML (Extensible Markup Language) for log files and compared log files to establish a basic structure for displaying.
- Use XSL (Extensible Stylesheet Language) to display the files.
- Use the XML parser to strip down the XML files, because the use of already existing building blocks raises the reusability and clearness of software and denotes a decline in programming effort.

In other words, the aim of this Master Thesis is to facilitate the execution of test units, edit the results in a clearly arranged way, and, as a final result, save a lot of time for the users (ooRexx developers) by providing quick navigation and clearly structured results without redundant information.

To understand the issue better, the following chapters specify every single part of TeRA. The arising problems are dealt with in the adequate chapters, as well as the problem solving.

This Master Thesis will start with a general chapter "*Software Testing*" (cf. chapter 2), which is an overview about the basics of software testing. After this the "*Unit Testing*" (cf. chapter 2.1 on page 14), a more specialised part of software testing will be dealt with. To understand the operating mode of unit testing frameworks, JUnit, a testing framework for the programming language Java will be the next issue (cp. chapters 2.1.1 and 2.1.1.1 on pages 14 and 16). The last part of the second chapter will be the different test runners (cf. chapter 2.1.2, page 17) for JUnit.

The next section will be the software tests (cp. chapter 3.1, "Unit Testing for Open Object Rexx", page 19) for the programming language Open Object Rexx (cf. chapter 7.2, "*Excursus: Open Object Rexx*", page 100).

This part of the Thesis will especially deal with the ooRexxUnit framework, a unit testing tool, similar to JUnit. The ooRexxUnit framework (cf. chapter 3.1.1, "*The ooRexxUnit Framework*", page 19) consists of the ooRexxUnit class (cf. chapter 3.1.1.1, "*The ooRexxUnit Class*", page 19) and different test units (cf. chapter 3.1.1.2, "*Test Units*", page 24), which will be parts of chapter 3 as well.

Chapter 4 (“*A TestRunner Application for OOREXXUNIT – TeRA*”, page 27) is then introducing the new graphical test runner for ooRexxUnit, called TeRA, which is the acronym for “***Te**st**R**unner **A**pplication for ooRexxUnit*”. As TeRA is the topic of this Thesis, chapter 4 will be the main part. At the beginning of this part, the architecture of TeRA (cp. chapter 4.1, “*TeRA-Architecture*”, page 28) will be explained with all references between the different components, which is quite important for the understanding of the whole test runner. Moreover, every single component of TeRA will be described in chapter 4.2 (“*Components of TeRA*”, page 31). There will be figures of TeRA architecture in nearly every chapter concerning the components, to get an overview of the whole test runner program all the time.

The “*Appendix*” (cp. chapter 1, page 94) is divided into two sections:

- The first section is a collection of basics (“*Excursus: Programming Language*”, “*Excursus: Rexx*”, “*Excursus: Common Public License*”, “*Excursus: HyperText Markup Language*”, “*Excursus: Cascading Style Sheets*”, “*Excursus: Extensible Markup Language*” and “*Excursus: Extensible Stylesheet Language*”), which are helpful, when reading this Thesis.

- The second part of chapter 7 is a collection of the source code of the different components of TeRA. The different chapters are not sorted after the structure of the components in chapter 4, but programs with the same purpose are arranged together (i.e. Cascading Style Sheets or Rexx programs are in the same chapter).

2 Software Testing

Software testing is a part of software engineering.

The software testing process is often divided into phases, which can be represented by a sequential software development model. One of the best-known sequential software development models is the waterfall model (figure 1). [cf. W3Se07b, HaNe05]

The first phase of the testing process is unit testing or module testing of software developed by a single programmer. [cf. W3Se07b, HaNe05]

The second phase is integration testing. There, units are combined and tested as a group. System testing is done on the entire system (with test cases developed from the system requirements). [cf. W3Se07b, HaNe05]

Acceptance testing of the system is the last phase and is done by its intended users. [cf. W3Se07b, HaNe05]

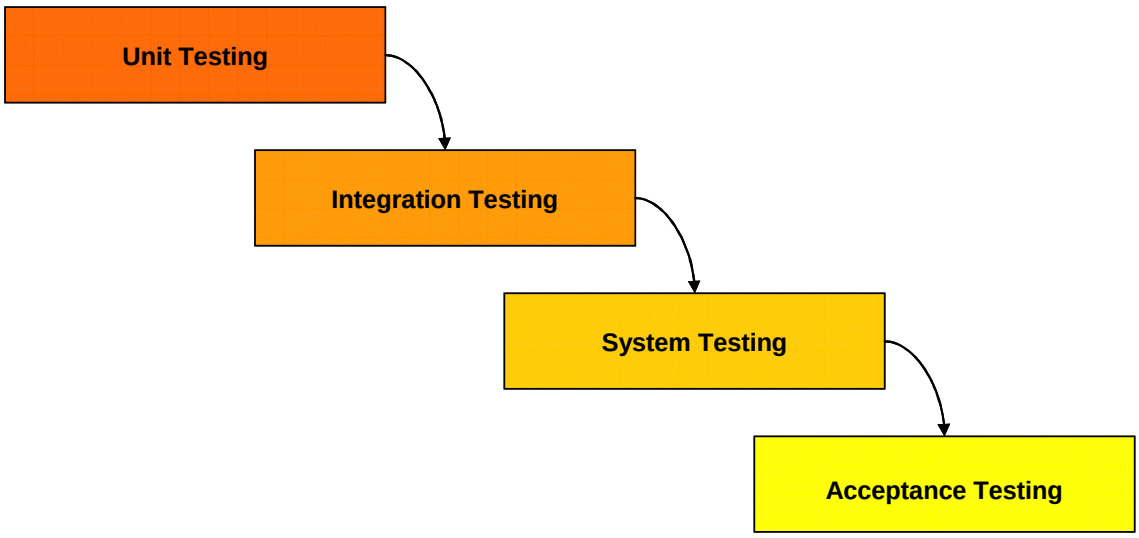

Figure 1: Software Testing (Waterfall Model).

The test case is the basic unit of testing.

Test cases consist of a

- test case type (aspect of the system that the test case is supposed to exercise),

- test conditions (consist of the input values for the test),
- the environmental state of the system used in the test and
- the expected behaviour of the system for the inputs and environmental factors given.

If software has to be changed, i.e. to fix bugs or enhance the software, there may be errors introduced. To ensure that no errors are raised, all test cases have to be rerun after each change (known as regression testing). [cf. W3Se07b, HaNe05]

Although software testing is one of the most important tasks within the software engineering, it is neglected very often.

There are a lot of reasons why software testing should be carried out in any case:

- Testing is an important part of quality management. Bad testing can have negative effects on the reliability of software products. This may lead to higher development and maintenance or even to the collapse of a project. [cf. HaNe05]
- Software systems are getting more complex. [cf. Ehmk07]
- The reliability of software systems is getting more important (e.g. software for hospitals or integrated software in the automotive engineering). [cf. Ehmk07]
- Lack of labour resources. [cf. Ehmk07]
- Testing processes cause a great deal of the development costs, therefore it is necessary to have efficient testing processes. [cf. HaNe05]
- A good testing process forces people involved to be disciplined and encourages them to deal with possible problems already at an earlier stage. [cf. HaNe05]
- With good testing processes, problems can be identified very early. [cf. HaNe05]

2.1 Unit Testing

Unit testing is a procedure used to validate that individual modules or software components (Units) are working properly.

Unit tests are tests for specific units of the program, where each test case should ideally be independent from the others. Typically, unit testing is done by the developers and not by end-users. [cf. W3Ut07]

There are quite a lot of testing frameworks for unit testing: [cf. W3Un07]

- JUnit: for the programming language[1] Java,
- NUnit: for all .Net languages,
- Cantata++: for C, C++ and EC++,
- ObjcUnit: for Objective-C on MacOS X,
- PalmUnit: for Palm devices[2],
- PearlUnit: for the programming language Pearl,
- Ruby Test::Unit: for the programming language Ruby,
- Unit++: for the programming language C++ and
- ooRexxUnit: for the programming language Open Object Rexx.

2.1.1 Java Unit (JUnit)

If a program feature lacks an automated test, it should be assumed that it does not work. This is a lot safer than the conventional assumption, that if a developer assures that a program feature works, it is really working. [cf. W3Jc07]

[1] Compare chapter 7.1 (“Programming languages”, page 94).
[2] The Palm Corporation produces a number of Personal Digital Assistants (PDAs) which run the Palm operating system.

Developers are not done when they write and debug the code. They have to write tests to demonstrate the correct functioning of the programs as well. However, sometimes there is not enough time for testing. [cf. W3Jc07]

Hence, the first goal in software testing is to create a framework using familiar tools, where developers will actually write tests in. One requirement for this framework is that no more work than writing a new test has to be done (eliminate duplicated effort). [cf. W3Jc07]

The second goal of testing is the creation of tests that retain their value over time. Someone other than the original author has to be able to execute the tests and interpret the results. Furthermore it should be possible to combine tests from various authors and run them together without fear of interference. [cf. W3Jc07]

JUnit is a simple, light-weight, open-source Java test execution framework. It is obtainable from www.junit.org and is included in the Enhanced JUnit product distribution. It provides a user interface (test runner) for selecting and launching tests, and APIs for asserting conditions. [cf. W3Jc07, W3Si07]

2.1.1.1 Architecture of JUnit

Figure 2 shows the architecture of JUnit.

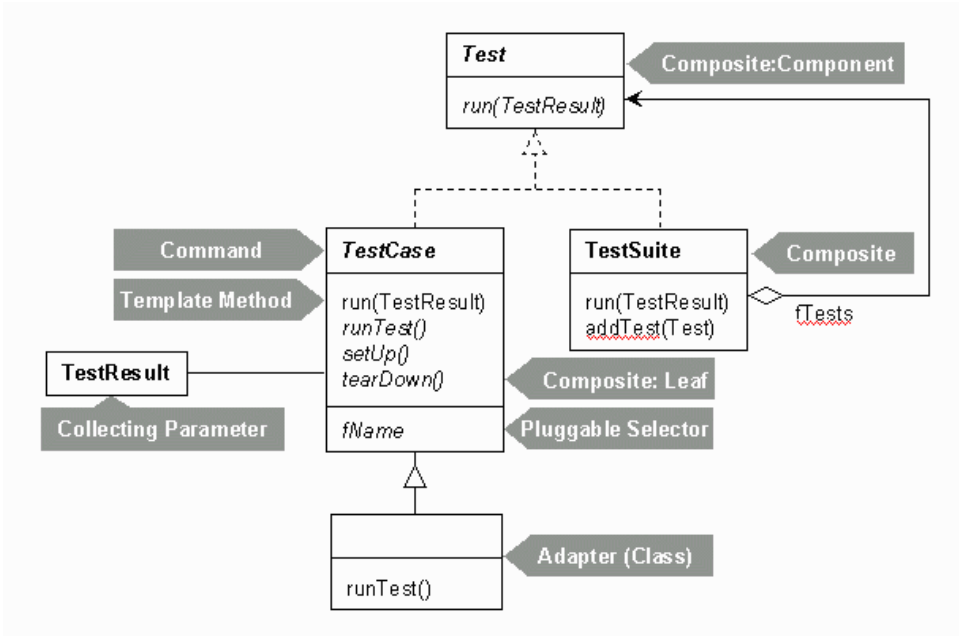

Figure 2[1]: JUnit Architecture[2].

[1] Source: *http://junit.sourceforge.net/doc/cookstour/Image6.gif*.
[2] The class TestCase specialises TestSuite.

2.1.2 Test Runner for JUnit

Test runner applications provide frameworks for performing unit tests.

Unit tests should be easy to execute, due to their immense importance to the development process of software. By providing an easy to use graphical user interface, test runner applications make creating and executing test suites more convenient and flexible. [cf. W3Tr07]

There are test runners for JUnit, which can be classified as follows:

- text-based (using the command line, i.e. the textui interface, figure 3) or
- graphical (graphical user interface[1] i.e. the swingui – figure 4 - or the awtui - figure 5 - interfaces).

Figure 3: Text Based User Interface. [cf. CySc02]

[1] "A graphical user interface is a particular case of user interface for interacting with a computer which employs graphical images and widgets in addition to text to represent the information and actions available to the user. Usually the actions are performed through direct manipulation of the graphical elements." [cf. W3Gu07]

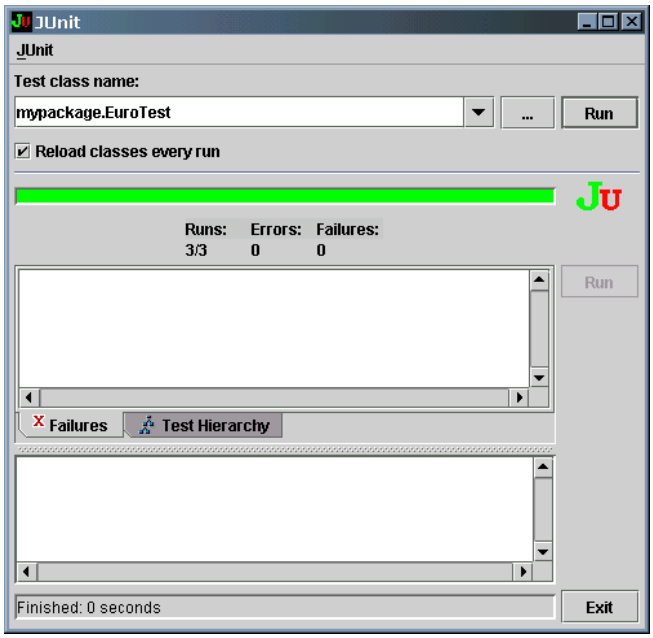

Figure 4: The SWINGUI User Interface. [cf. CySc02]

Figure 5 shows the graphical test runner awtui for JUnit.

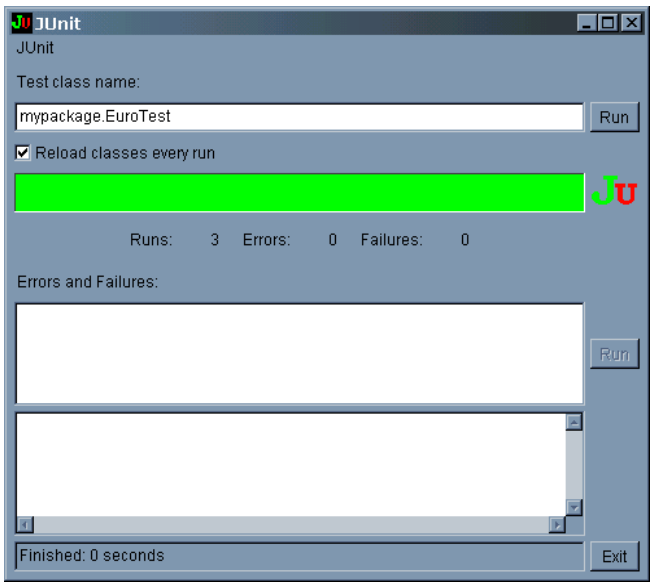

Figure 5: The AWTUI User Interface. [cf. CySc02]

There are several IDEs[1] implemented in JUnit. [cf. W3Ju07, W3Si07]

[1] "An integrated development environment (IDE), also known as integrated design environment and integrated debugging environment, is a type of computer software that assists computer programmers in developing software." [cf. W3Id07]

3 Testing Open Object Rexx

As well as for the programming language Java there is a unit testing framework for Open Object Rexx (compare chapter 7.2 “*Excursus: Rexx*”, page 100), the ooRexxUnit framework.

“ooRexxUnit is an Open Object Rexx implementation of the JUnit testing framework. It allows for creating and running ooRexx test cases, which assert whether application specifications are met. One usually creates such test cases according to the specifications parallel to the development of the application and employs them every time the application is developed further or changed because of maintenance purposes, e.g. because bugs need to get fixed.” [cf. Flat06]

3.1 Unit Testing for Open Object Rexx

To test the different parts (units) of ooRexx, there have to be so called test units created corresponding to the part that should be tested, e.g. classes, routines or built-in-functions. [cp. chapter 3.1.1.2 - “*Test Units*”, page 24]

3.1.1 The ooRexxUnit Framework

The ooRexxUnit framework consists of the ooRexxUnit class (chapter 3.1.1.1 – “The *ooRexxUnit Class*”, page 19) and the test units (chapter 3.1.1.2 – “*Test Units*”, page 24).

3.1.1.1 The ooRexxUnit Class

The ooRexxUnit class (ooRexxUnit.CLS) is the most important part of the TeRA test runner. It is the framework which is used by TeRA to compare the class methods[1].

[1] [cp. Flat06].

The ooRexxUnit.CLS consists of four core classes which tasks are to set up rules for comparisons, assemble test cases, test suites and to provide the test results. These four parts are the Assert class, TestCase class, TestSuite class and TestResult class. The whole source code can be found in chapter 7.8.3.1 (“*ooRexxUnit.CLS*”, page 160).

3.1.1.1.1 Architecture of ooRexxUnit

The TestCase class is a subclass of Assert and TestSuite is a subclass of TestCase.

Figure 6 shows the architecture of ooRexxUnit.CLS.

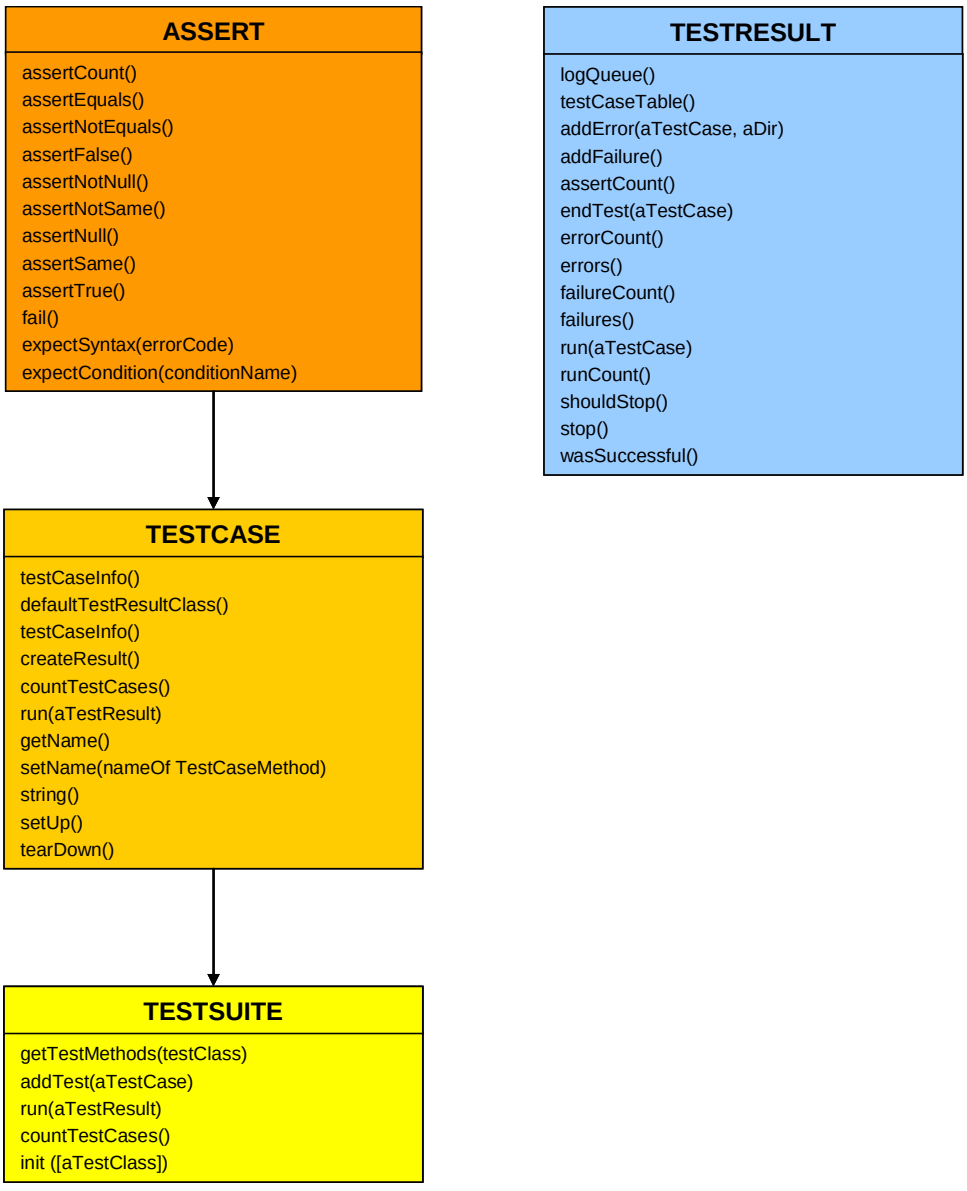

Figure 6: The ooRexxUnit Architecture.

All methods of the Assert class are available to the instances of the TestCase and TestSuite class and in turn all methods of the TestCase class are available to all instances of TestSuite. [cf. Flat06]

3.1.1.1.2 The Assert Class

The Assert class defines all assertion methods, which are used for testing assertions. [cf. Flat06]

Generally, if the assertion fails, the fail method will be invoked.

- **assertCount**: counts successful assertions.
- **assertEquals(failMsg, val1, val2)**: compares val1 and val2 for equality (=).
- **assertNotEquals(failMsg, val1, val2)**: compares val1 and val2 for not being equal.
- **assertFalse(failMsg, val)**: checks if the result of an equitation is .false (the string “0”).
- **assertNotNull(failMsg, val)**: expects any value, but the .nil object.
- **assertNotSame(failMsg, val1, val2)**: compares val1 and val2 for not being identical (using the == operator and comparing its result with .false).
- **assertNull(failMsg, val)**: expects the .nil object as the result of the operation.
- **assertSame(failMsg, val1, val2)**: compares val1 and val2 for identity using the == operator.
- **assertTrue(failMsg, val)**: checks, if the result of an equitation is .true (the string “1”).
- **fail(failMsg)**: used to raise an error, because user exception needs to be propagated.
- **expectSyntax(errorCode)**: expects a syntax condition with the given errorCode.

- **expectCondition(conditionName)**: expects a condition with the given conditionName.

3.1.1.1.3 The TestCase Class

The TestCase class is a subclass of the Assert class and provides the methods to set up and tear down a testing environment. Furthermore it contains methods to start the test case (run), get (getName) and set (setName) the name of the test case and to create the test result object (createResult). [cf. Flat06]

- **testCaseInfo** (class attribute): directory object which allows the storing of information about all test cases.
- **defaultTestResultClass** (class attribute): allows storing of the class object which is used by the instance method createResult.
- **testCaseInfo** (instance attribute): storing information about a particular test case.
- **createResult**: returns a TestResult object (an instance of the class object stored in the class attribute defaultTestResultClass).
- **countTestCases**: counts the number of test cases.
- **run([aTestResult])**[1]: is concerned with the tasks of running test cases.
- **getName**: returns the name of the test case method that gets run.
- **setName(nameOf TestCaseMethod)**: sets the test case object to name a different test method to run.
- **string**: creates a string representation of the test case object.
- **setUp**: Null Operation. “If a test case needs to set up a specific environment before its tests are run, the test class needs to override it by implementing itself a method named setUp.”

[1] Expressions between brackets (“[“ and “]”) are optional.

- **tearDown**: Null Operation. “If a test case needs to tear down a specifically set up environment or clean up after a test case ran, then the test class needs to override it by implementing itself a method named tearDown.”

3.1.1.1.4 The TestSuite Class

The TestSuite class is a subclass of the TestCase class. It allows to set up a Suite, a collection of test cases which should be executed. [cf. Flat06]

- **getTestMethods(testClass)**: retrieves all test methods of the testClass and sorts them alphabetically.
- **addTest(aTestCase)**: adds the test case object to the test suite.
- **run([aTestResult])**[1]: runs all the test cases in the test suite and returns the TestResult object (where results of running test cases are logged).
- **countTestCases**: counts number of test cases in the test suite.
- **init ([aTestClass])**: if the argument aTestClass (optional) is given, then all methods of that test class starting with the string “test” are regarded to be test case methods.

3.1.1.1.5 TestResult Class

The TestResult class is responsible for saving the results of the running test cases to an instance of the TestResult class. Test cases can run successfully (assertions hold), can fail (assertions do not hold) or an unexpected error may occur (e.g. a syntax error). [cf. Flat06]

- **logQueue**: returns a queue containing directory objects created by the run method.
- **testCaseTable**: returns a table, whose associated item is a queue (indices are the individual test case items).

[1] Expressions between brackets (“[“ and “]”) are optional.

- **addError(aTestCase, aDir)**: queues the directory object (condition) to the logQueue, the errors queue and adds a string to the testCaseTable queue.
- **addFailure(aTestCase, aDir)**: queues the directory object (condition) to the logQueue, the failures queue and adds an appropriate encoded string to the testCaseTable queue.
- **assertCount**: counts number of successful assertions.
- **endTest(aTestCase)**: encodes the date and time into a string (added to logQueue and testCaseTable).
- **errorCount**: returns the number of errors.
- **errors**: returns error queue.
- **failureCount**: returns the number of failures.
- **failures**: returns failure queue.
- **run(aTestCase)**: runs the given test case and returns a test result object.
- **runCount**: gets number of run test cases.
- **shouldStop**: returns a boolean value to indicate whether the running test cases should stop (.true stops, .false does not stop the running test cases).
- **stop**: sets shouldStop to return .true.
- **wasSuccessful**: returns .true if no failures or errors have been encountered, otherwise it returns .false.

3.1.1.2 Test Units

Test units are another important part of the ooRexxUnit framework, where e.g. all methods of a class can be tested. Test units either contain one or more test classes, each of which may have test methods defined that represent individual test cases. To test a class of ooRexx, a matching test unit has to be created. The file extension of test unit programs is .testUnit. Within the test unit files, there are assertions which test each single method of the ooRexx class.

An example of a test unit can be found in chapter 7.8.4.1 (“*ooRexx.base.class.mutableBuffer.testUnit*”, page 182)

As already mentioned, the assertion methods are situated within the Assert class in the ooRexxUnit.CLS. As Assert is a super class of the TestCase class, all of its methods are available to the instances of TestCase with `self~` (e.g. `self~assertSame()`).

```
::method "test_APPEND"

  TestBufferAPP=.mutableBuffer~new
  TestBufferAPP~insert(“TEST4”)
  TestBufferAPP~insert(“TEST3”)
  TestBufferAPP~insert(“TEST2”)
  TestBufferAPP~insert(“TEST1”)
  TestBufferAPP~append(“TEST5”)
  TestBufferAPP~append(“TEST5”)
    self~assertSame("subTest1",     "TEST1TEST2TEST3TEST4TEST5", TestBufferAPP~string)

  TestBufferAPP~append(“TEST6”)
    self~assertSame("subTest2", "TEST1TEST2TEST3TEST4TEST5TEST6", TestBufferAPP~string)

  TestBufferAPP~append(“TEST7”)
    self~assertSame("subTest3",     "TEST1TEST2TEST3TEST4TEST5TEST6TEST7", -
    TestBufferAPP~string)

  TestBufferAPP~append(“TEST8”)
    self~assertSame("subTest4",     "TEST1TEST2TEST3TEST4TEST5TEST6TEST7TEST8", -
    TestBufferAPP~string)
```

Code 1: Test for Method Append of the MutableBuffer Class.

Code 1 shows a part of the oorexx.base.class.mutableBuffer.testUnit, a test unit created for the testing of the MutableBuffer class. The aim of the first lines (`TestBufferAPP~insert(“TEST+Number”)`) is to set up a test buffer. With `~insert()` e.g. different values and strings can be added to the buffer. AssertSame is a method of the ooRexxUnit.CLS, Assert class. This test checks whether the string “`TEST1TEST2TEST3TEST4TEST5`” is identical to the content of the mutable Buffer (`TestBufferAPP~string`).

The method’s names have to start with the string “test” (as seen in code 1: `::method “test_APPEND”`), because the method getTestMethods of the TestSuite class returns an ascendingly sorted array object, which contains all the names of methods, whose names start with the string “test”. [cf. Flat06}

```
::requires oorexxunit.cls
```

Code 2: Make Classes and Routines of ooRexxUnit.CLS available.

The directive (code 2) causes the interpreter to call the ooRexxUnit framework program, which will make all its public classes and routines available to the test unit program. [cf. Flat06]

3.1.2 Test Runner (for ooRexxUnit)

As seen with JUnit, there is a text based test runner application for ooRexxUnit, as well:

- runallTests.REX, which tests all test units found on the hard disk (figure 7).

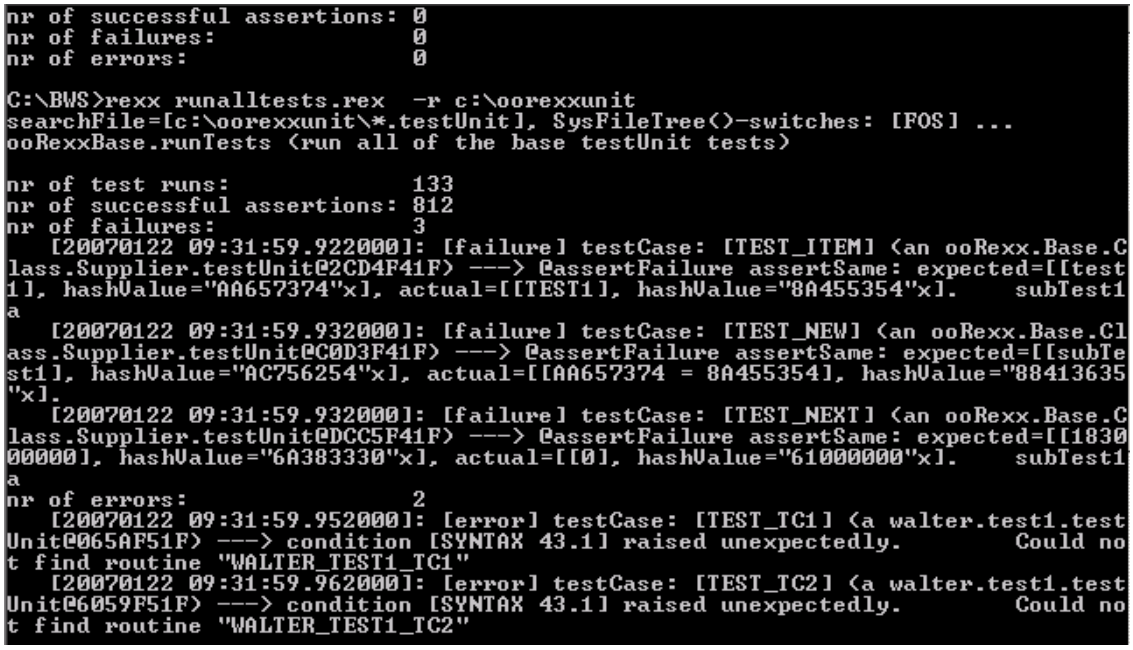

Figure 7: RUNALLTESTS.REX, Test Runner for ooRexxUnit.

The disadvantages of the text based test runners are the missing of clearness, the nonexistent interactivity and the unavailable possibility to highlight important parts.

The following chapters will deal with the new graphical test runner for Open Object RexxUnit, called TeRA. [cf. chapter 4 - "*A TestRunner Application for OOREXXUNIT (TeRA)*", page 27]

4 A TestRunner Application for ooRexxUnit – TeRA

TestRunner Application for ooRexxUnit (TeRA) is a graphical test runner application for ooRexxUnit. TeRA was programmed just using the programming language Open Object Rexx.

To increase the usability of TeRA for the users XML, XSL and HTML were used in the course of creating the test runner application. Therewith it is possible to change most parts of TeRA, just with knowledge in XML, XSL and HTML. This notably applies to the XML parser and all result pages (log files and compared log files). With the use of these languages the reusability and further development will be easier.

The TeRA test runner application opens up the possibility to run, easily navigate through, select and deselect and sort test units. A further advantage over the text based test runner is the fact, that it is very easy to run single or a choice of test units as well as all test units, just by clicking the corresponding checkboxes (or buttons).

Another advantage is the structured and highlighted result pages (i.e. log files and compared log files).

With TeRA it is also possible to compare log files to identify possible improvements or worsenings. Differences between two compared log files are highlighted to easily spot the changes.

Therewith it is not necessary to analyse the whole result pages, but only analysing the highlighted parts.

A disadvantage of TeRA might be that nearly all information generated by the ooRexxUnit framework is displayed within the result pages, but not all. If the not displayed information is needed it can only be found in the file INTERMEDIARY.TXT. Actually, this “hidden” information should not be relevant.

4.1 TeRA Architecture

The architecture of TeRA is quite large and complex. Figure 8 visualizes the references between the different components of TeRA.

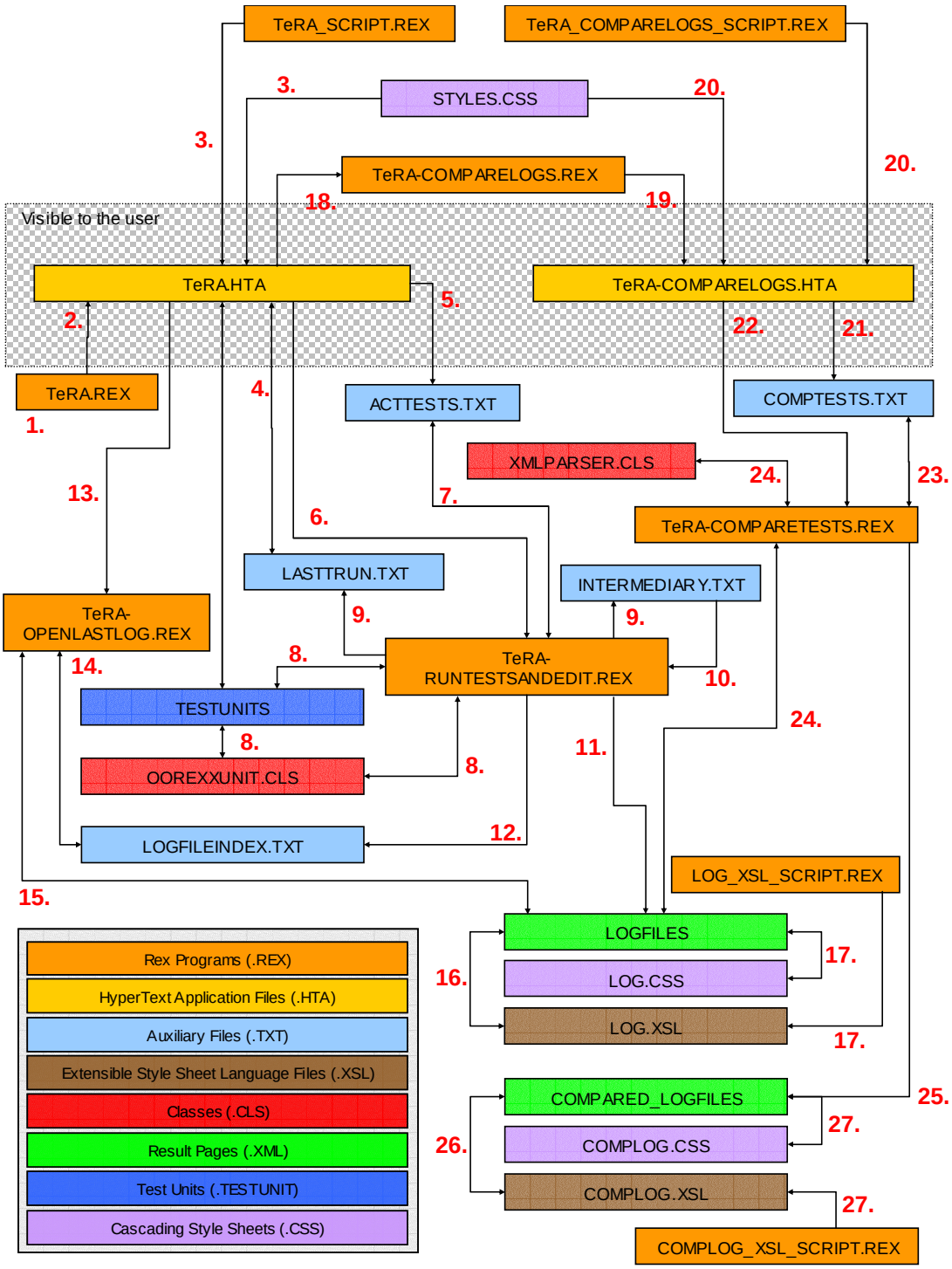

Figure 8: The TeRA Architecture.

The TeRA test runner consists of eight different kinds of files:

- Rex programs (.REX): to create HTA files and process data, as well as containing routines.
- HyperText Application (.HTA) files: to act as graphical user interfaces.
- Cascading Style Sheets (.CSS): to structure the HTA files.
- Text files (.TXT): auxiliary files (no XML files).
- Result pages (log files and compared log files - .XML): to store the results of a test run or the results of a comparison of two test runs.
- Extensible Style Sheet Language files (.XSL): to structure the result pages and contain the corresponding routines.
- Classes (.CLS):
  XMLPARSER.CLS: to parse the log files.
  OOREXXUNIT.CLS: to get all important methods for the test runs.
- Test Units (.TESTUNIT): for testing e.g. each class or built-in-function of the programming language ooRexx.

The following achievement should be used with figure 8.

To start the test runner, the program TeRA.REX (**1.**) has to be started. TeRA.REX generates the first user interface – TeRA.HTA (**2.**). When TeRA.HTA is loaded, the routines of TeRA_SCRIPT.REX, are loaded into the user interface (**3.**) the routine `TUCode` (part of TeRA_SCRIPT.REX) checks the file LASTTRUN.TXT (for failure and error highlighting) (**4.**) and TeRA.HTA gets the styling rules out of STYLES.CSS (**3.**). Within TeRA.HTA test units can be selected and the test run can be started by clicking the button `run tests`. The selected test units are stored into ACTTESTS.TXT (**5.**).

When the button `run tests` is clicked TeRA-RUNTESTSANDEDIT.REX is started (**6.**). TeRA-RUNTESTSANDEDIT.REX reads the information in ACTTESTS.TXT (**7.**), uses ooRexxUnit.CLS to execute the test run (**8.**), stores the results to INTERMEDIARY.TXT (**9.**) and writes the information (which test units raised an error or failure) to LASTTRUN.TXT (**9.**). Then the information in INTERMEDIARY.TXT is analysed (**10.**) and the important data is stored to the

log file (XML file). Then TeRA-RUNTESTSANDEDIT.REX adds an entry to the file LOGFILEINDEX.TXT (list of generated log file names) (**12.**).

When the button `show result` is clicked (in TeRA.HTA) (**13.**), the program TeRA-OPENLASTLOG.REX extracts the last entry in LOGFILEINDEX.TXT (name of last generated log file) (**14.**) and opens the log file with the corresponding name (**15.**).

If opening a log file, the routines of LOG_XSL_SCRIPT.REX are loaded into the HTML (**17.**), which is created using the file LOG.XSL (**16.**) with the styling rules of LOG.CSS (**17.**).

When the button `compare tests` is clicked (**18.**), TeRA-COMPRELOGS.REX is started, which generates the second user interface TeRA-COMPARELOGS.HTA (**19.**).

The routines of TeRA_COMPARELOGS.REX are loaded into TeRA-COMPARELOGS.HTA (**20.**) and the styling rules from STYLES.CSS (**20.**) are used.

To compare log files, two of them have to be selected in TeRA-COMPARELOGS.HTA and when clicking the button `compare logs` the names of the selected log files are stored to COMPTESTS.TXT (**21.**). The program TeRA-COMPARELOGS.REX is started (**22.**).

TeRA-COMPARELOGS.REX reads the information in COMPTESTS.TXT (**23.**) and uses the methods of XMLPARSER.CLS to compare the two log files (**24.**). While comparing the log files, a compared log file is created (XML file) (**25.**). When opening the compared log file, the routines of COMPLOG_XSL_SCRIPT.REX are loaded into the HTML (**27.**), which is created using the file COMPLOG.XSL (**26.**) with the styling rules of COMPLOG.CSS (**27.**).

All single parts of the whole architecture will be well explained in the following chapters.

4.2 Components of TeRA

As already mentioned in chapter 4.1 (“TeRA Architecture”, page 28), TeRA consists of several components:

- TeRA.REX,
- TeRA.HTA,
- STYLES.CSS,
- TeRA_SCRIPT.REX,
- TeRA-OPENLASTLOG.REX,
- TeRA-RUNTESTSANDEDIT.REX,
- ooRexxUnit.CLS,
- TeRA-COMPARELOGS.REX,
- TeRA-COMPARELOGS.HTA,
- TeRA_COMPARELOGS_SCRIPT.REX,
- TeRA-COMPARETESTS.REX,
- XMLPARSER.CLS,
- LOG.CSS,
- LOG.XSL,
- LOG_XSL_SCRIPT.REX,
- COMPLOG.CSS,
- COMPLOG.XSL and
- COMPLOG_XSL_SCRIPT.REX.

All parts of TeRA are under the Common Public License version 1.0. [cf. chapter 7.3, “*Excursus: Common Public License – CPL*”, page 102]

4.2.1 TeRA.REX

TeRA.REX is the starting program of the TeRA test runner (cf. chapter 7.8.1.1, "*TeRA.REX*", page 110). Figure 9 shows the references of TeRA.REX to other components.

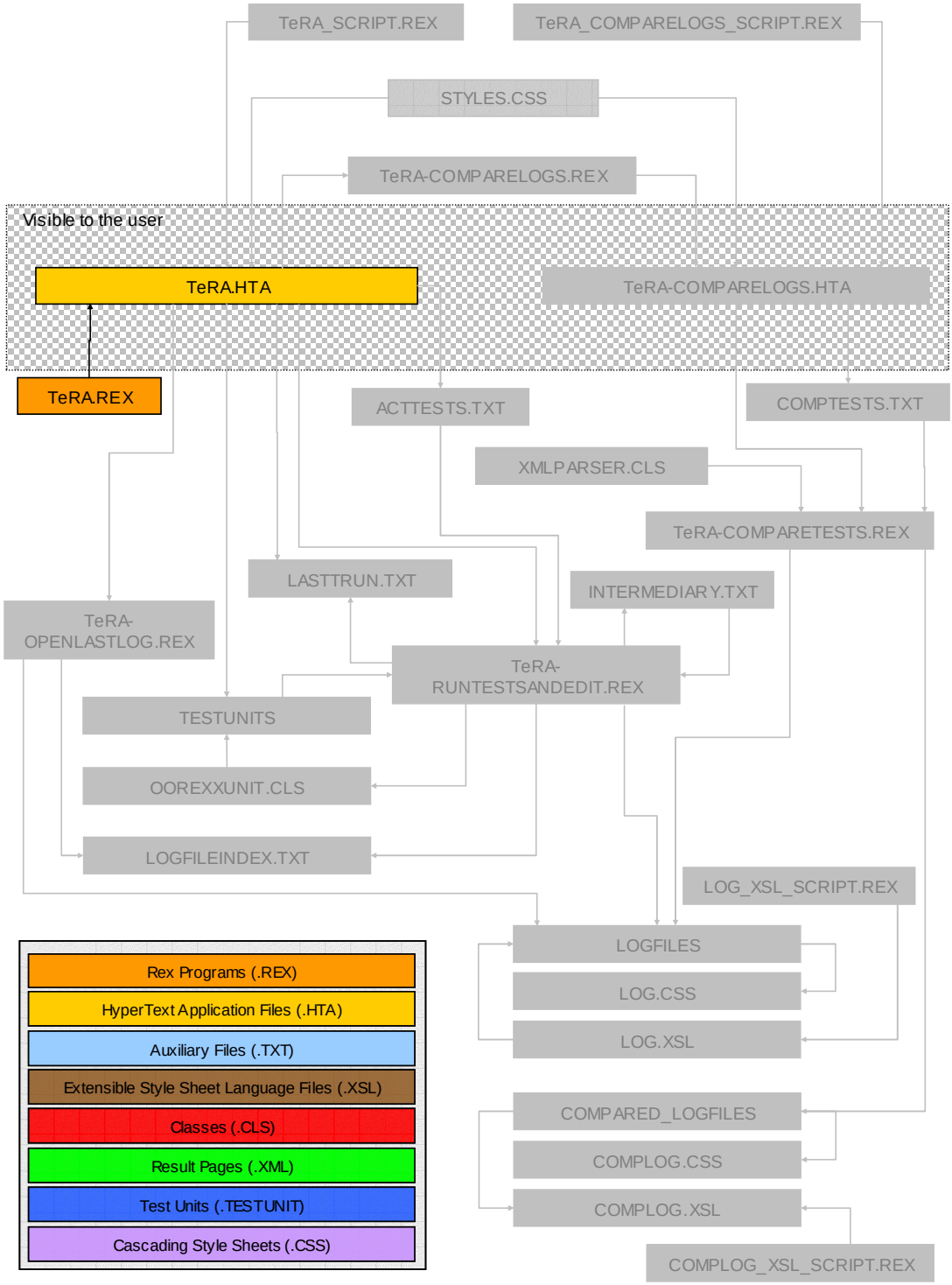

Figure 9: TeRA.REX, Reference to Other Components.

TeRA.REX generates a HTA[1] (hypertext application) file, which is used to select, run, show and navigate through test and test runs.

The first task of TeRA.REX is providing the structure for the generated HTA file TeRA.HTA.

Furthermore TeRA.REX sets up the structure for the test unit checkbox tree, which is used to select and navigate through test units.

Figure 10 shows the HTA file TeRA.HTA.

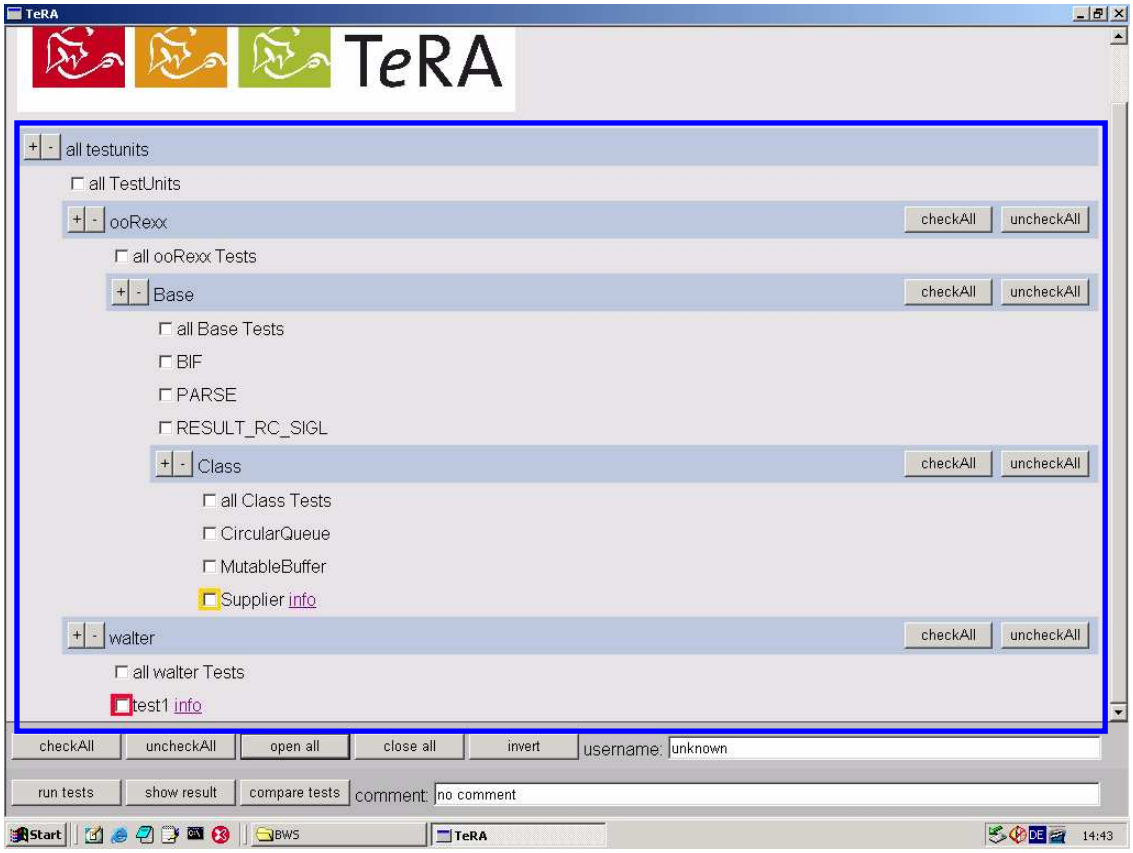

Figure 10: TeRA.HTA (Test Unit Checkbox Tree).

The generation of the checkbox tree is done by searching for all test units, which can be found on the hard disk using the extension ".testUnit" as searching criteria.

If there are no test units, the program stops and no HTA file is created.

[1] File type similar to html files, but without security restrictions.

If test units can be found, TeRA.REX returns the test unit paths (e.g. oorexx\Base\Class\oorexx.base.class.mutableBuffer.testUnit). Subsequent to this, the path is cut into three parts – the path itself, the file name and the extension. The path and the file extension are of no interest in the further proceedings. The file name is then fractionalized further (using the "." indicating the different parts – e.g. oorexx.base.class.mutableBuffer is cut into 4 parts: oorexx, base, class and mutableBuffer). This is very important, because these parts will determine the structure of the test unit tree in the HTA file (the last part is the name of the test unit).

Assumption: oorexx.base.class.mutableBuffer is the first test unit found on the hard disk, when running TeRA.REX. The program then creates 4 levels of folders:

- **allTestUnits**: level0, is predefined (can not be changed)
- **oorexx**: level1, all oorexx tests are situated within this folder
- **base**: level2, all base tests
- **class**: level3: all class tests

There is no folder created for mutableBuffer. This is due to the fact that this is the name of the class (mutableBuffer) which will be tested with this test unit. Therefore mutableBuffer will be added to the class folder.

Assumption: The second test unit found is e.g. oorexx.base.class.Supplier. In this case the folders oorexx, base and class already exist and Supplier is just added to the class folder.

The last task of TeRA.REX is the allocation of the buttons, which are used to run the various routines (chapter 4.2.1.1.2 "*TeRA_SCRIPT.REX*", page 39).

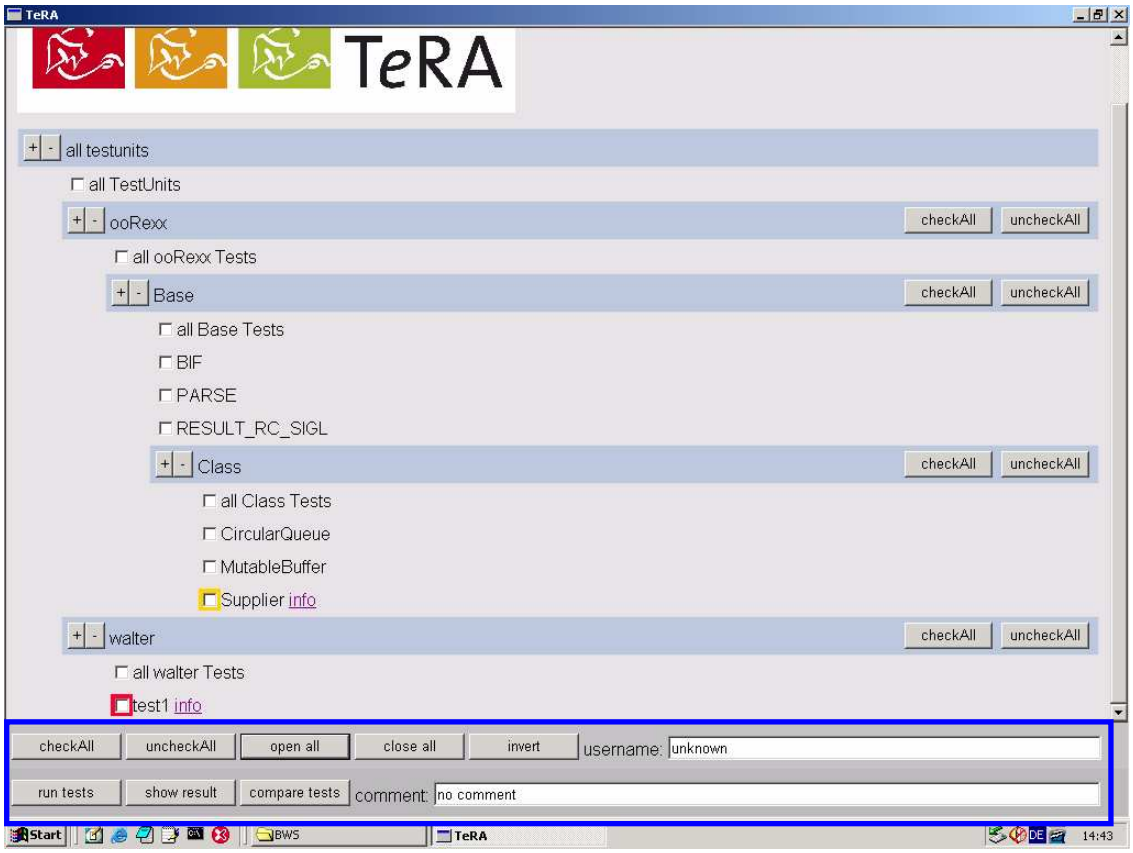

Figure 11: TeRA.HTA (Buttons).

4.2.1.1 TeRA.HTA

As already mentioned in chapter 4.2.1 ("*TeRA.REX*", page 32), TeRA.HTA is produced by TeRA.REX. It is used as graphical user interface to facilitate the navigation for the users.

Figure 12 shows all references of TeRA.HTA to other components of the TeRA test runner.

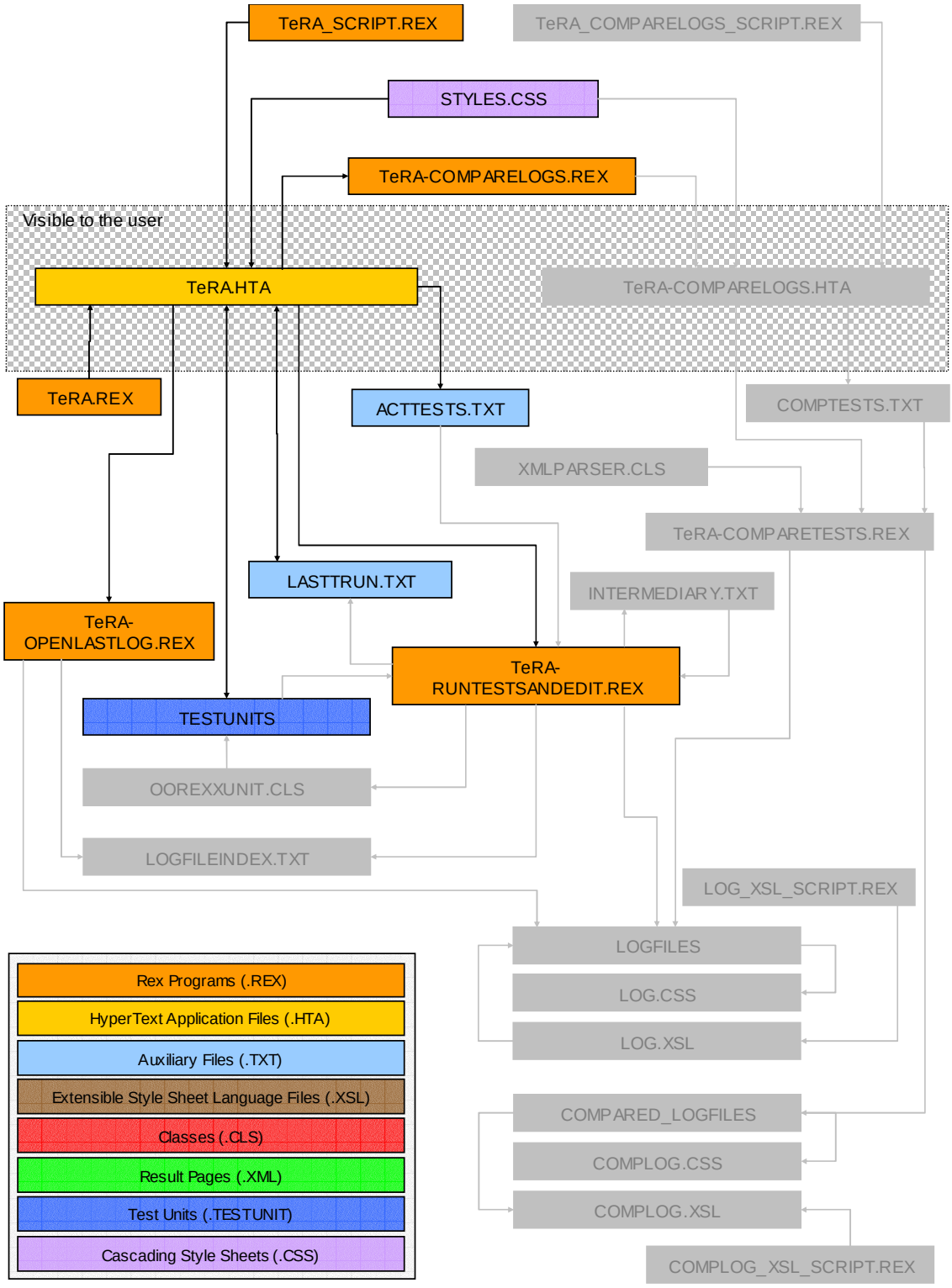

Figure 12: TeRA.HTA, Reference to Other Components.

Microsoft Internet Explorer (MS IE) is used to display TeRA.HTA. It is a proprietary graphical web browser developed by Microsoft. [cf. W3Ie07]

The Component Object Model[1] (COM) technology is extensively used in Internet Explorer. It allows to add functionality and allows websites to offer rich content via ActiveX[2]. [cf. W3Ie07]

IE uses a zone-based security framework. This means that sites are grouped, based upon certain conditions and allows the restriction of broad areas of functionality and specific functions to be restricted. [cf. W3Ie07]

Internet Explorer almost fully supports: [cf. W3Ie07]

- HTML 4.01,
- CSS Level 1,
- XSL,
- XML 1.0 and
- DOM[3] Level 1, with minor implementation gaps.

It partially supports: [cf. W3Ie07]

- CSS Level 2,
- DOM Level 2, with some implementation gaps and conformance issues and
- XHTML.

The reason why Microsoft Internet Explorer is used in combination with TeRA is that it provides a lot of functions which are quite useful (e.g. the printing function and the searching function) and it is able to display files even if they are not valid, which can be a problem with other browsers.

---

[1] "Platform for software componentry introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology." [cf. W3Co07]

[2] ActiveX is the name of a set of "strategic" object-oriented programming technologies and tools. The main technologie is COM (Component Object Model). The main thing that you create when writing a program to run in the ActiveX environment is a component, a self-sufficient program that can be run anywhere in your ActiveX network. One main advantage of a component is that it can be re-used by many applications. A COM component object (ActiveX control) can be created using one of several languages (e.g. Visual Basic, Power Builder) or development tools (e.g. VBScript). For example: Word and Excel documents can be viewed directly in a Web Browser that supports ActiveX. [cp. W3Ac07]

[3] "***D**ocument **O**bject **M**odel,* the specification for how objects in a Web page (text, images, headers, links, etc.) are represented." [cf. W3Do07]

4.2.1.1.1 TeRA.HTA – User Interface

To get an idea of how the user interface works, all parts of TeRA.HTA will be explained in this chapter.

Figure 13 shows the user interface TeRA.HTA.

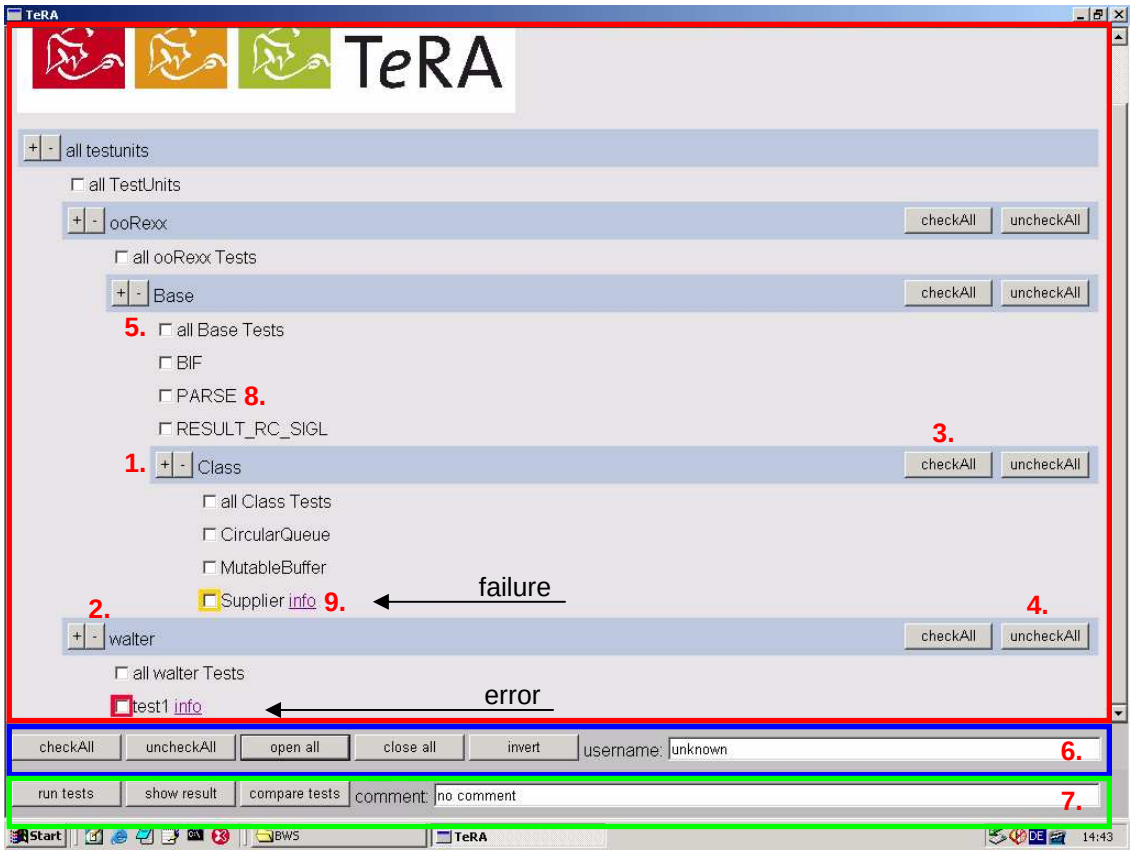

Figure 13: TeRA.HTA – Three Sections (Red: `content_container`, Blue: `footer1`, Green: `footer2`).

- The button `+` opens a folder of the test unit checkbox tree (in this case the folder `class`). [cp. figure 13 - **1.**]
- The button `-` closes a folder of the test unit checkbox tree (in this case the folder `walter`). [cp. figure 13 - **2.**]
- `checkAll` checks all test unit within a folder (i.e. in this case all test units of the folder class). [cp. figure 13 - **3.**]
- `uncheckAll` unchecks all test units within a folder. [cp. figure 13 - **4.**]
- When clicking an `all` checkbox, all following test units of the branch are selected. [cp. figure 13 - **5.**]

- The buttons `checkAll` and `uncheckAll` check/uncheck all checkboxes of the test unit checkbox tree. [cp. figure 13 - **6.**]
- `openAll` and `closeAll` open/close all folders of the checkbox tree. [cp. figure 13 - **6.**]
- The button `invert` checks all checkboxes, which are not selected and deselects all checkboxes, which are selected. [cp. figure 13 - **6.**]
- The text area `username` is used to add a user name to the log file name. [cp. figure 13 - **6.**]
- With `run tests` the program TeRA-RUNTESTSANDEDIT.REX is started and tests the selected test units. [cp. figure 13 - **7.**]
- `show results` opens the last log file. [cp. figure 13 - **7.**]
- `compare tests` starts the program TeRA-COMPARELOGS.REX, which generates the second user interface TeRA-COMPARELOGS.HTA (to compare log files). [cp. figure 13 - **7.**]
- The text area `comment` is used to add a comment to the log file. [cp. figure 13 - **7.**]
- When clicking on a test unit in the checkbox tree, the corresponding test unit will be opened. [cp. figure 13 - **8.**]
- When clicking the `info` link next to a test unit, the last log file is opened at the position where the test run information of this test unit is situated. [cp. figure 13 - **9.**]

4.2.1.1.2 TeRA_SCRIPT.REX

TeRA_SCRIPT.REX contains the HTA script section for TeRA.HTA with all the necessary routines (compare chapter 4.2.1.1.2, “*TeRA_SCRIPT.REX*”, page 39). When TeRA.HTA is loaded, the routines are loaded as well by appointing the path to TeRA_SCRIPT.REX within the script tag of TeRA.HTA (`<script language='Object Rexx' src='path-to-TeRA_SCRIPT.REX'></script>`).

The following routines are defined in TeRA_SCRIPT.REX:

- **SelectedTestUnits**: collects the values (paths) of the selected test units, stores them to the auxiliary file (ACTTESTS.TXT) and starts the program TeRA-RUNTESTSANDEDIT.REX.
- **openLastLog**: opens the last generated log file by executing the ooRexx program TeRA-OPENLASTLOG.REX.
- **compareLogFiles**: starts the program TeRA-COMPARELOGS.REX.
- **invert**: checks all checkboxes which are not selected and unchecks all checkboxes which are selected.
- **uncheckAll**: is used to uncheck all checkboxes in the test unit checkbox tree.
- **checkAll**: is used to check all checkboxes in the test unit checkbox tree.
- **openAll**: opens all folders and subfolders of the test unit checkbox tree.
- **closeAll**: closes all folders and subfolders of the test unit checkbox tree.
- **checkmore**: is concerned with checking all test units within one branch of the test unit checkbox tree. (i.e. when selecting the all base tests checkbox, all base tests and class tests are selected, because the class folder is a subfolder of base).
- **Tucode**: is used to highlight the test units which raised a failure (yellow) or an error (red). Furthermore it displays the info links in the HTA file, which link to the matching positions of the last log file (e.g. if the Supplier test unit raises a failure, the corresponding checkbox is highlighted yellow and the info link is displayed on the right side of the checkbox (compare figure 13). When clicking this link, the last log file is opened at the position, where the failure log information of the Supplier test unit starts). All information about the last test run, which are used by this routine is stored in LASTTRUN.TXT.
- **folder_checkAll**: checks all test unit checkboxes within a folder. The corresponding buttons are situated in the folder's header.

- **folder_uncheckAll**: unchecks all test unit checkboxes within a folder. The corresponding buttons are situated in the folder's header.

Figure 14 shows two highlighted test units in the checkbox tree (failure highlighted in yellow, error highlighted in red).

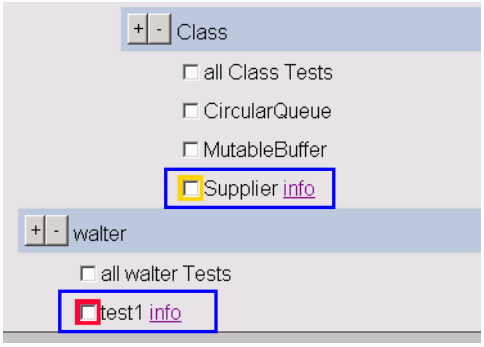

Figure 14: Highlighted Test Units in the Checkbox Tree and Info Link.

The whole source code can be found in chapter 7.8.1.2 ("*TeRA_SCRIPT.REX*", page 117).

4.2.1.1.3 STYLES.CSS

STYLES.CSS (Cascading Style Sheet) is the main CSS file, where all important styling rules for TeRA.HTA and TeRA-COMPARELOGS.HTA are stored (compare chapter 7.8.2.1, "*STYLES.CSS*", page 151).

The HTML body of the two HTA files (TeRA.HTA and TeRA-COMPARELOGS.HTA) is divided into three sections - the content_container, the footer1 and the footer2. This is due to the fact that Microsoft's Internet Explorer (up to version 6) was not able to use the `position: fixed;` CSS rule, which is used to allocate HTML elements to special (fixed) positions. These fixed objects do not move, even if the HTA or HTML page is scrolled down. The `position:fixed;` is normally used for task bars. As TeRA should also be working with all versions of Internet Explorer (versions 5, 6 and 7), another solution had to be aspired.

A solution was to define different areas for each section. The content_container contains the TeRA logo and the test units checkbox tree. The section is set to a height of 88% of the page.

The footer1 and footer2 sections have a height of 6% each.

Within footer1, the checkAll, uncheckAll, openAll, closeAll and invert buttons can be found.

Within footer2, the run tests, show results and compare tests buttons are situated.

To show the functionality and importance of Cascading Style Sheets to this test runner, figure 15 shows a HTA file, which does not use STYLES.CSS, while figure 16 shows the same HTA file using the Cascading Style Sheet.

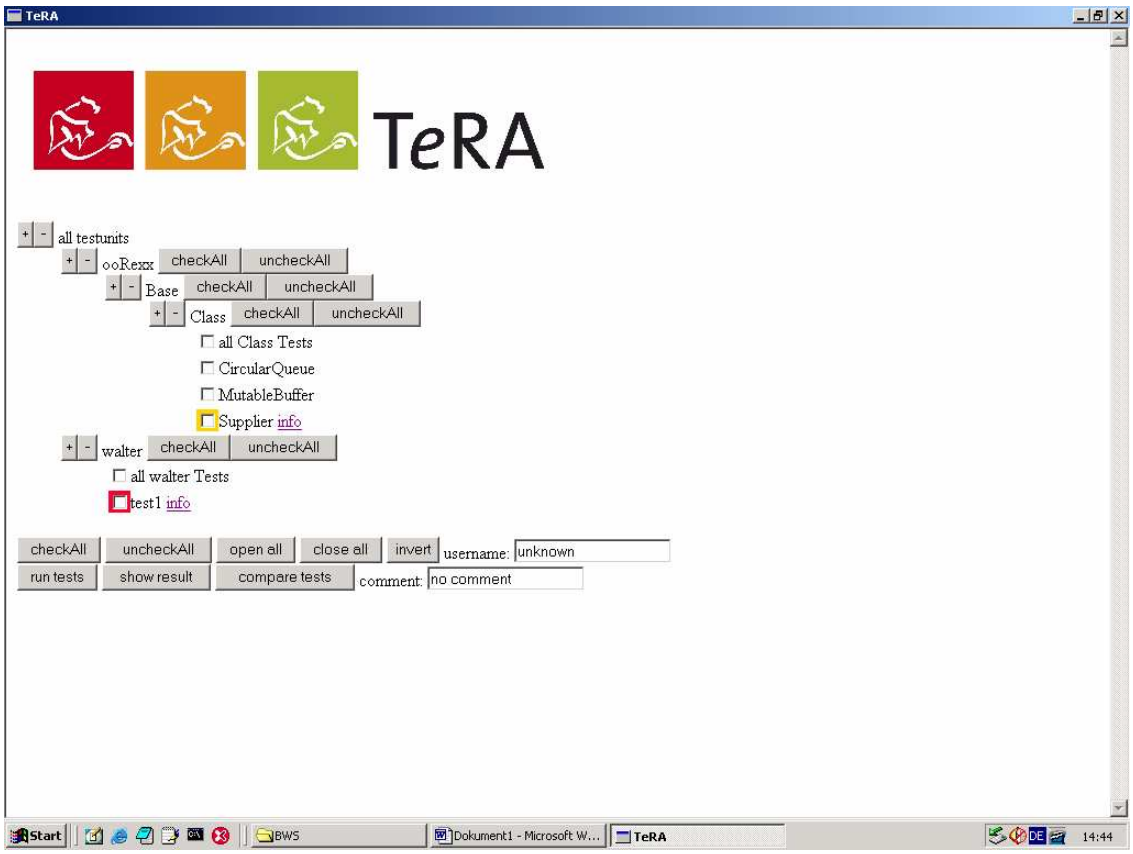

Figure 15: TeRA.HTA (without STYLES.CSS).

Figure 16 shows TeRA.HTA file using STYLES.CSS.

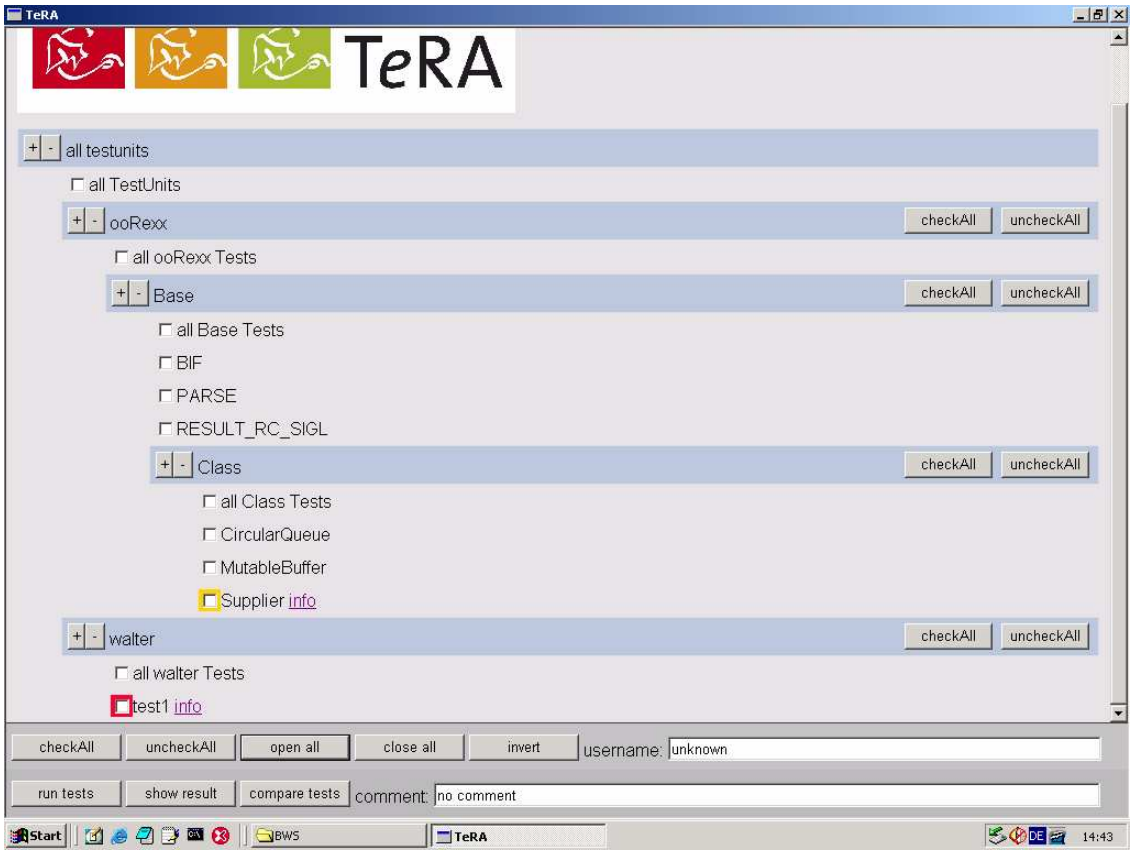

Figure 16: TeRA.HTA (using STYLES.CSS).

4.2.1.1.4 TeRA-OPENLASTLOG.REX

TeRA-OPENLASTLOG.REX is the smallest program of TeRA. Figure 17 shows the references of TeRA-OPENLASTLOG.REX to the other components.

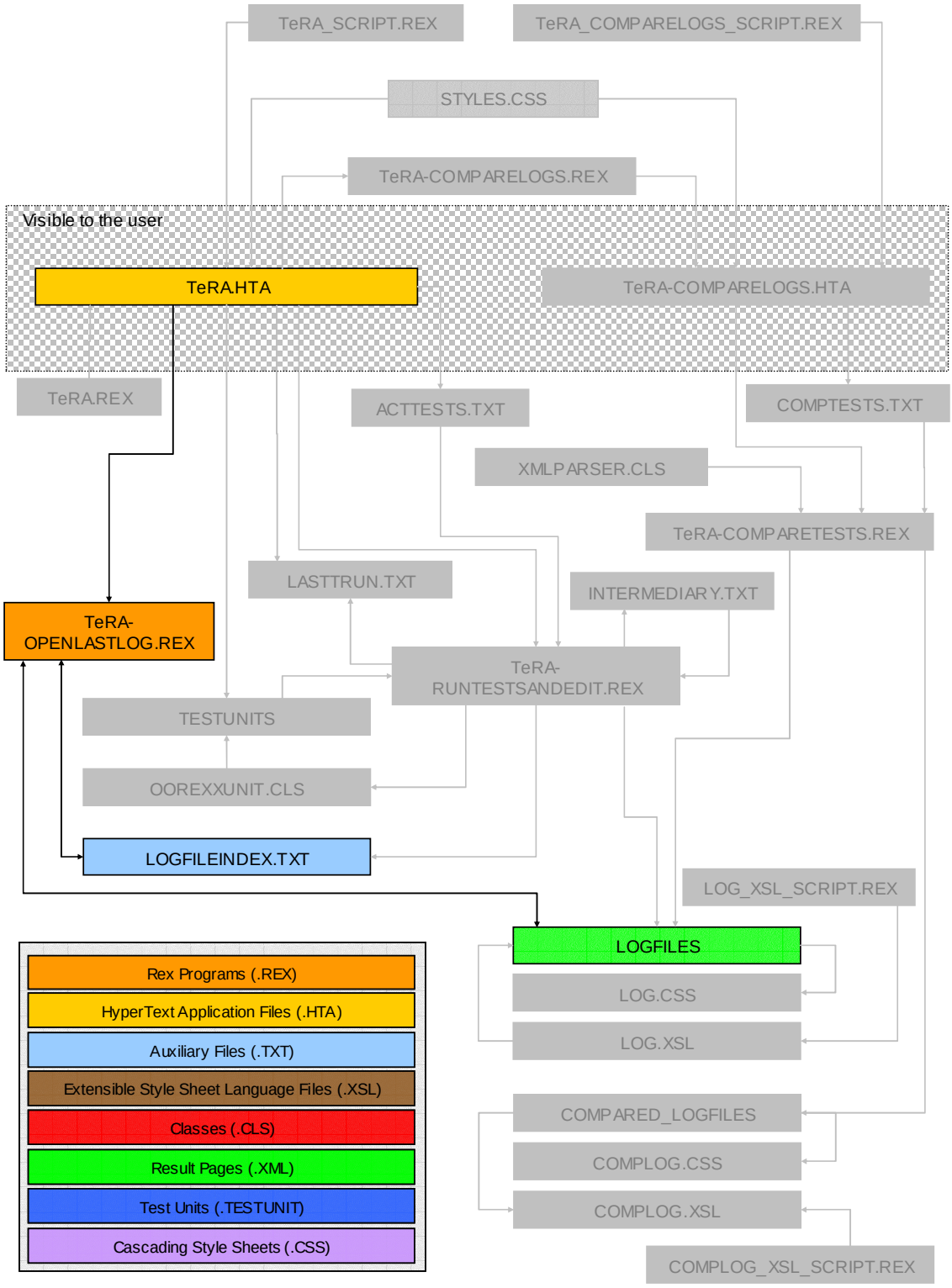

Figure 17: TeRA-OPENLASTLOG.REX, Reference to Other Components.

This short Rexx program simply opens the last generated log file. This is done by getting the last entry of LOGFILEINDEX.TXT (list of log file names) and opening the corresponding log file.

The whole source code can be found in chapter 7.8.1.3 (“*TeRA-OPENLASTLOG.REX*”, page 122).

4.2.1.1.5 TeRA-RUNTESTSANDEDIT.REX

The program TeRA-RUNTESTSANDEDIT.REX is a major part of the TeRA test runner. It is responsible for running tests, creating the auxiliary files INTERMEDIARY.TXT, ACTTESTS.TXT, LOGFILEINDEX.TXT and LASTTRUN.TXT. Furthermore it creates the log files, where the whole information about the test runs is stored.

Figure 18 shows all references of TeRA-RUNTESTSANDEDIT.REX to the other components of TeRA.

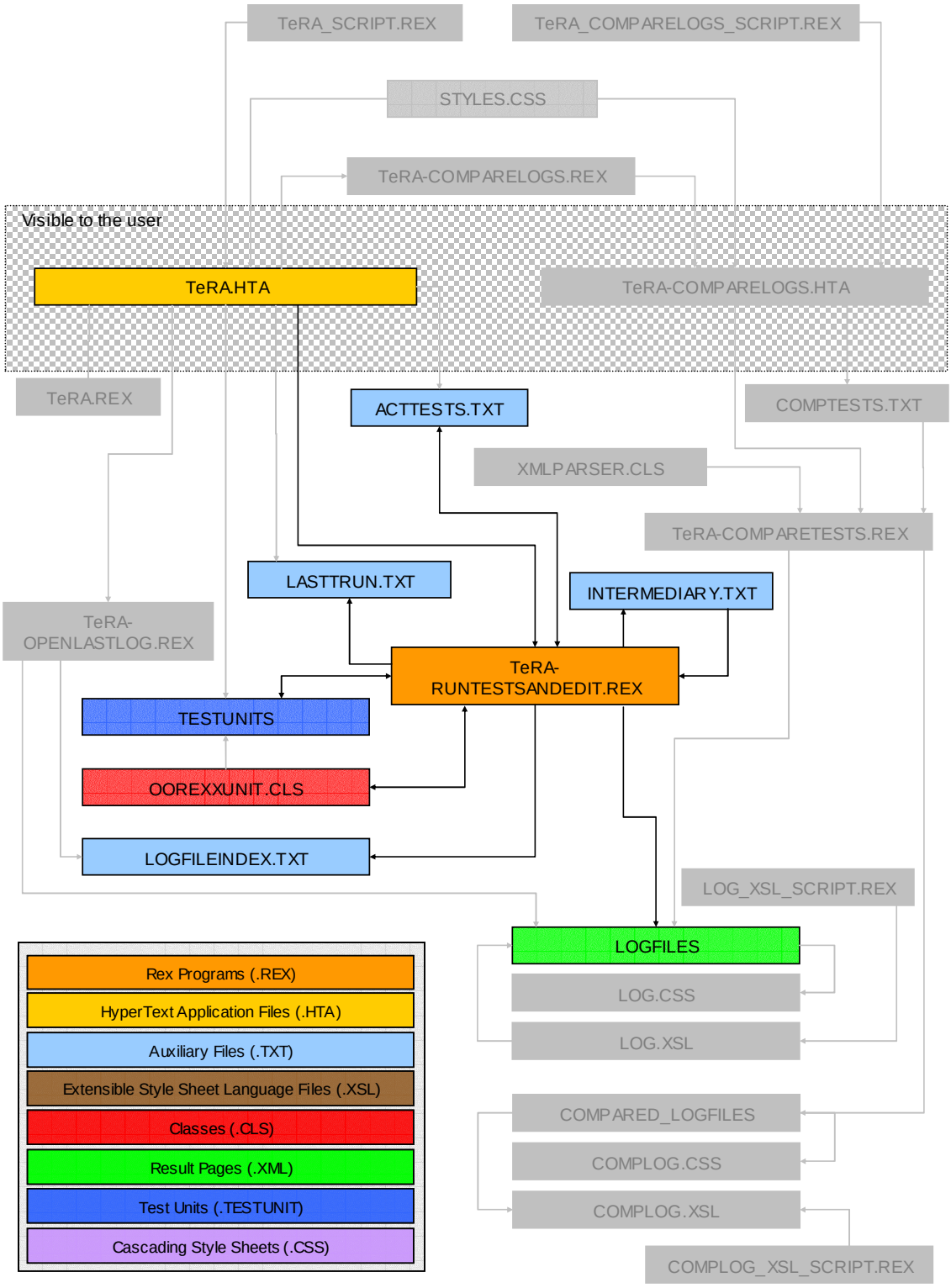

Figure 18: TeRA-RUNTESTSANDEDIT.REX, Reference to Other Components.

First, TeRA-RUNTESTSANDEDIT.REX creates the name of the log file which is going to be created. The log file's name is made up of a time stamp (date: *yyyymmdd* and time: *hh.mm.ss*), the user name, which can optionally be stated

within TeRA.HTA (if not stated a string “unknown” is added instead - compare figure 23) and a string (“LOG”), all connected with “_”.

Figure 19 shows, how the names of log files can look like.

```
log_20070112_16.39.06_unknown.xml
log_20070113_03.15.31_unknown.xml
log_20070113_03.16.26_unknown.xml
log_20070113_03.26.17_unknown.xml
```

Figure 19: Log File Names.

There are three directories, which are important, because the whole information (log files and compared log files) and the auxiliary files are stored there. These three (TeRA_files, logfiles and compared_files) are created in TeRA-RUNTESTSANDEDIT.REX by getting the actual position (path) and using `SysmkDir(path)` to create.

The following folders are created:

- TeRA_files: is used for all auxiliary files (e.g. LASTTRUN.TXT).
- logfiles: contains all generated log files and the auxiliary file LOGFILEIN-DEX.TXT.
- compared_files: is used for the log files which have been compared.

After creating the three directories, the program gets the information, which test units should be tested, out of the auxiliary file ACTTESTS.TXT (created by the routine SelectedTestUnits, chapter 4.2.1.1.2, “*TeRA_SCRIPT.REX*”, page 39).

Figure 20 shows an example of the content of ACTTESTS.TXT.

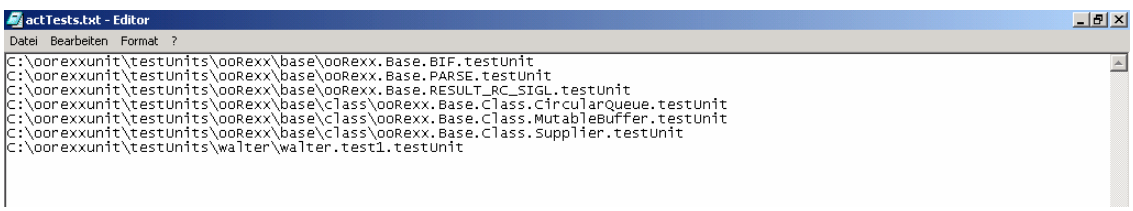

Figure 20: An Example of the Content of ACTTESTS.TXT.

Moreover, the testing is started using the ooRexxUnit.CLS. The results stored in `logQueue` are dumped and stored in another auxiliary file called INTERMEDIARY.TXT (in TeRA auxiliary files are no XML files).

In figure 21 a single test case (no errors or failures) is highlighted blue. The first line contains the information about the start of the test case (`[startTest]`), whereas the second line contains the end information (`[endTest]`)

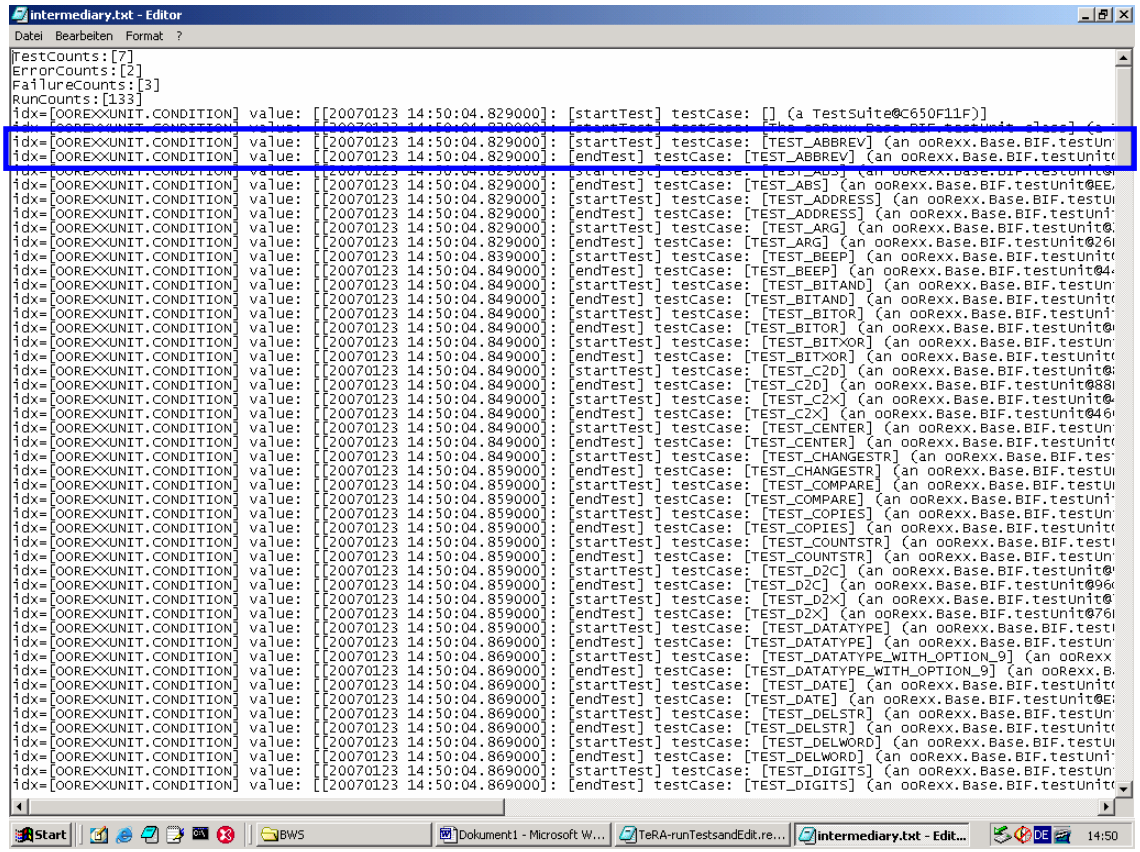

Figure 21: A Part of INTERMEDIARY.TXT (No Errors or Failures).

Figure 22 shows how a test case looks like, when a failure (`[failure]`) occured in the test run.

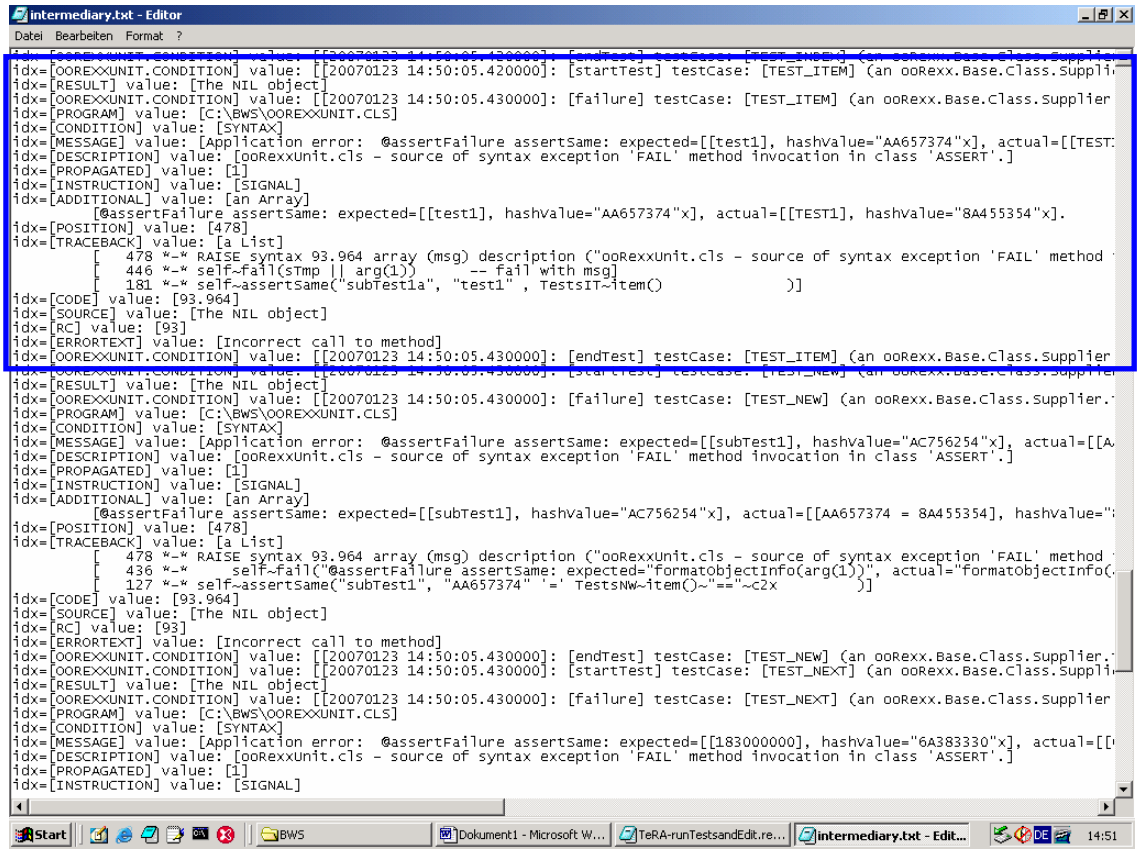

Figure 22: A Part of INTERMEDIARY.TXT (Failure).

After creating the auxiliary file INTERMEDIARY.TXT, the program TeRA-RUNTESTSANDEDIT.REX parses the information (of INTERMEDIARY.TXT) and the important data is stored to the log file. The name of the log files is added to LOGFILEINDEX.TXT as another index entry.

Figure 23 shows an example of the content of LOGFILEINDEX.TXT.

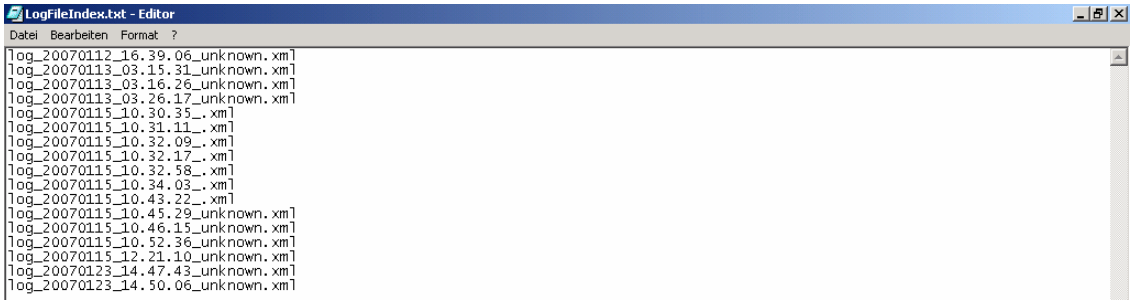

Figure 23: An Example of the Content of LOGFILEINDEX.TXT.

If a failure or an error occurred in a test run, an info link is added to the corresponding test unit in the test unit checkbox tree in TeRA.HTA. With this info link it is possible to jump to the position of the test unit in the log file, where

the error or failure occurred. As the names of the log files can not be predicted in advance (because the name is just generated when the program starts) and therefore the links can not be associated to the corresponding log file, there has to be a log file with a known name. This problem is solved by copying the last generated log file (always under the same name LastLogFile.XML) to the directory TeRA_files. With this proceeding it is possible to link to the right file.

The log file is stored to the directory logfiles.

Afterwards the auxiliary files INTERMEDIARY.TXT and ACTTESTS.TXT are deleted.

Figure 24 shows a part of TeRA.HTA (footer1 and footer2, Highlighted: User Name)

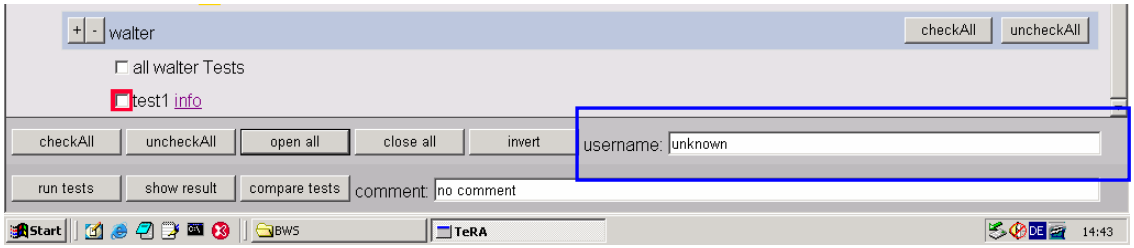

Figure 24: User Name (Optional).

The whole source code can be found in chapter 7.8.1.4, (“*TeRA-RUNTESTSANDEDIT.REX*”, page 123).

4.2.1.1.6 Log Files

A log file is a collection of important data, which shows the results of a test run. It is created in the form of a XML file. Colours in the rendering are used to highlight special information:

- Light yellow : start and end date and time of the test suite.
- Light grey : start and end date and time of a test case.
- Yellow : to highlight test cases which raised a failure.
- Red : to highlight test cases which raised an error.

Figure 25 shows all references of the log files to other components.

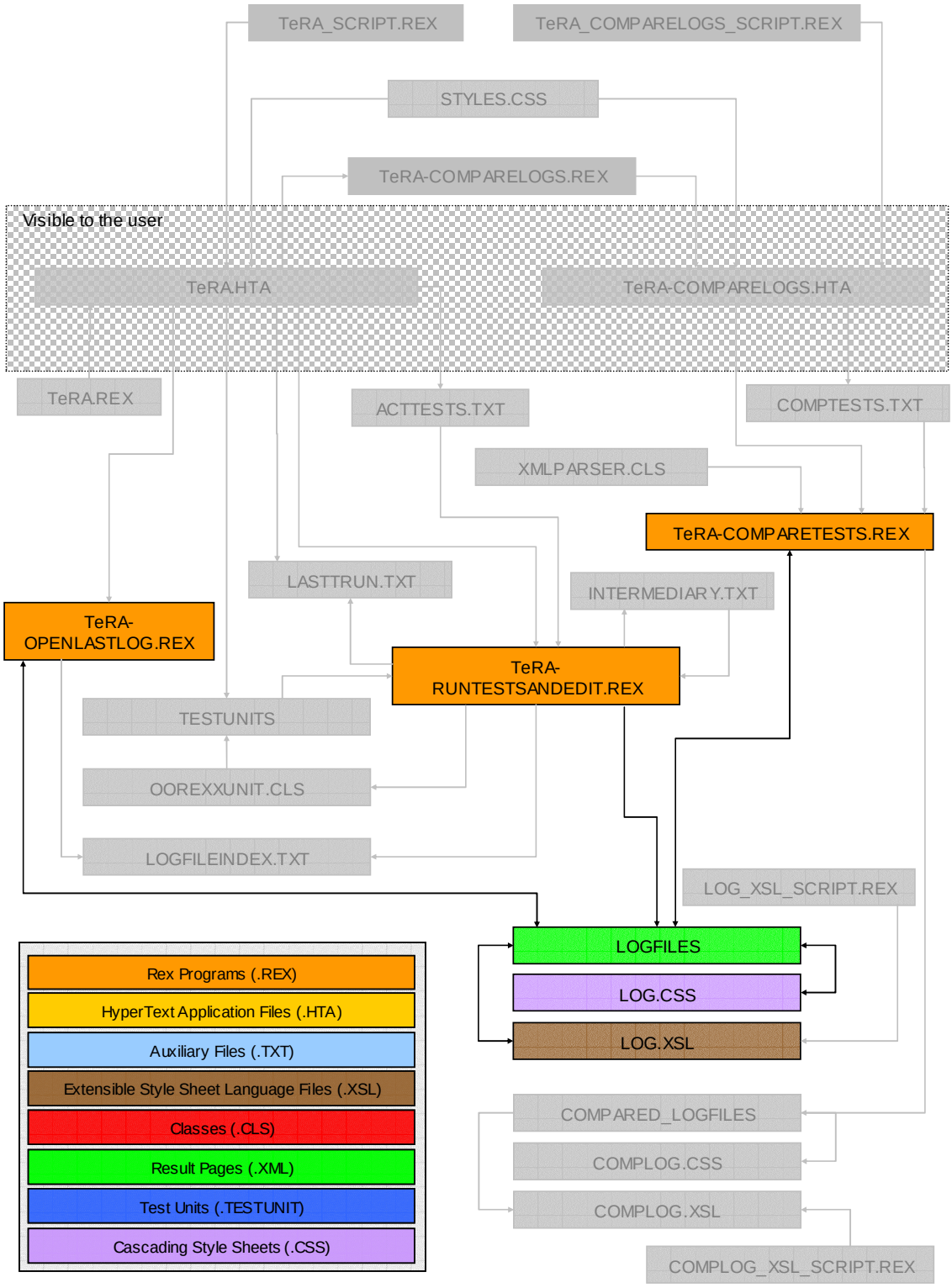

Figure 25: Log Files, Reference to Other Components.

Figure 26 shows a part of INTERMEDIARY.TXT (green: *status=okay*, blue: *status=failure*).

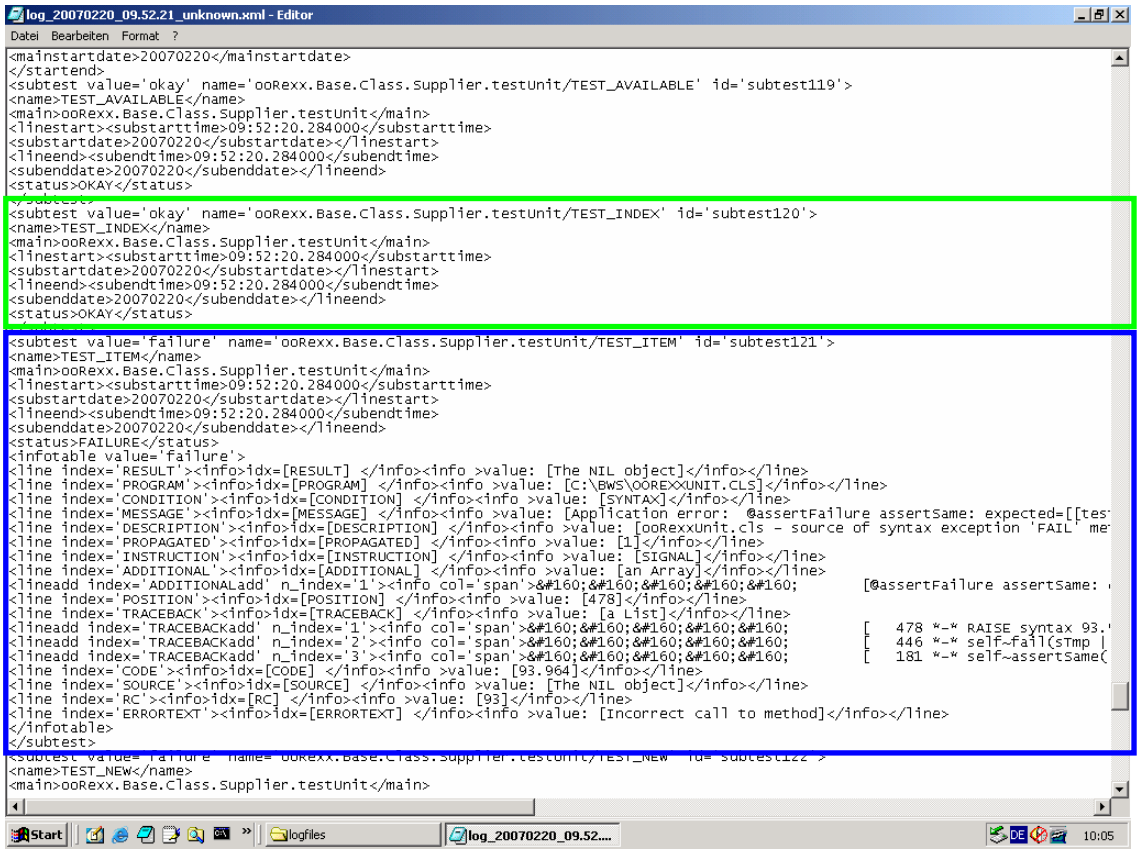

Figure 26: A Part of the Log File – Failure (Blue) and Okay (Green).

The following XML tags are in every test case (log files), independent of the status (okay, failure or error).

- **<subtest>**: contains three attributes: **value**: `value="okay"` *or* `value="failure"` *or* `value="error"`; **name**: `name="test unit name/test case name"`, giving the whole name of the test unit and test case; **id**: `id="subtest+number"`.
  - **<name>**: contains the test case name.
  - **<main>**: contains the test unit name.
  - **<linestart>**: contains the tags `<substarttime>` and `<substartdate>`.
    - **<substarttime>**: start time of the test run of the test case.
    - **<substartdate>**: start date of the test run of the test case.
  - **<lineend>**: contains the tags `<subendtime>` and `<subenddate>`.
    - **<subendtime>**: end time of the test run of the test case.
    - **<subenddate>**: end date of the test run of the test case.

- **<status>**: contains the status of the test case.

Moreover, if a test case raises an error or failure, the following tags are added to the above mentioned tags:

- **<infotable>**: table for the error of failure information.
  - **<line>**: contains the info tags.
  - **<info>**: contains all different information about the error or failure.

Figure 27 shows the XML structure of test cases (tests, which did not raise an error or failure, alternatively the first part of test cases, which raised an error or failure).

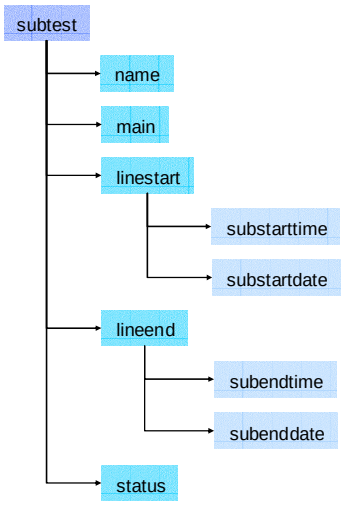

Figure 27: XML Structure of a Test Case.

Figure 28 shows the XML structure of test cases which raised a failure or an error in the test run.

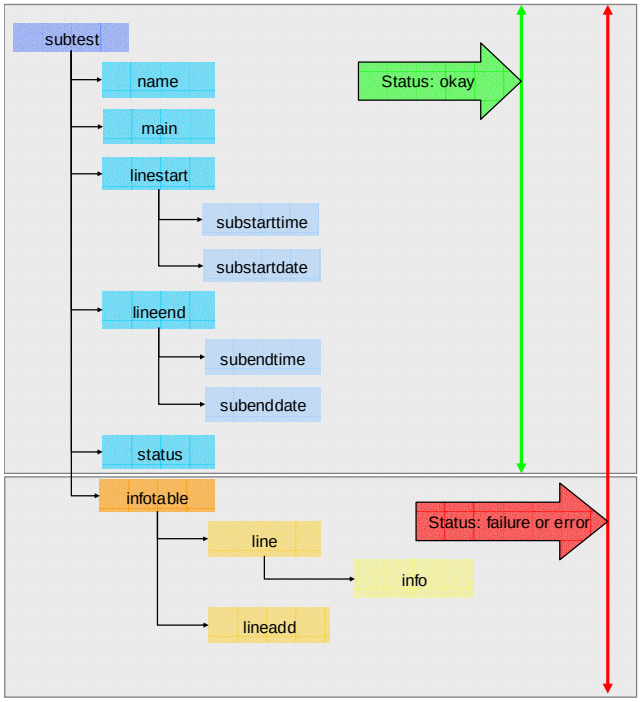

Figure 28: Test Case (with the Status: Error or Failure)

4.2.1.1.6.1 Log File – User Interface

Figure 29 shows an example of a log file.

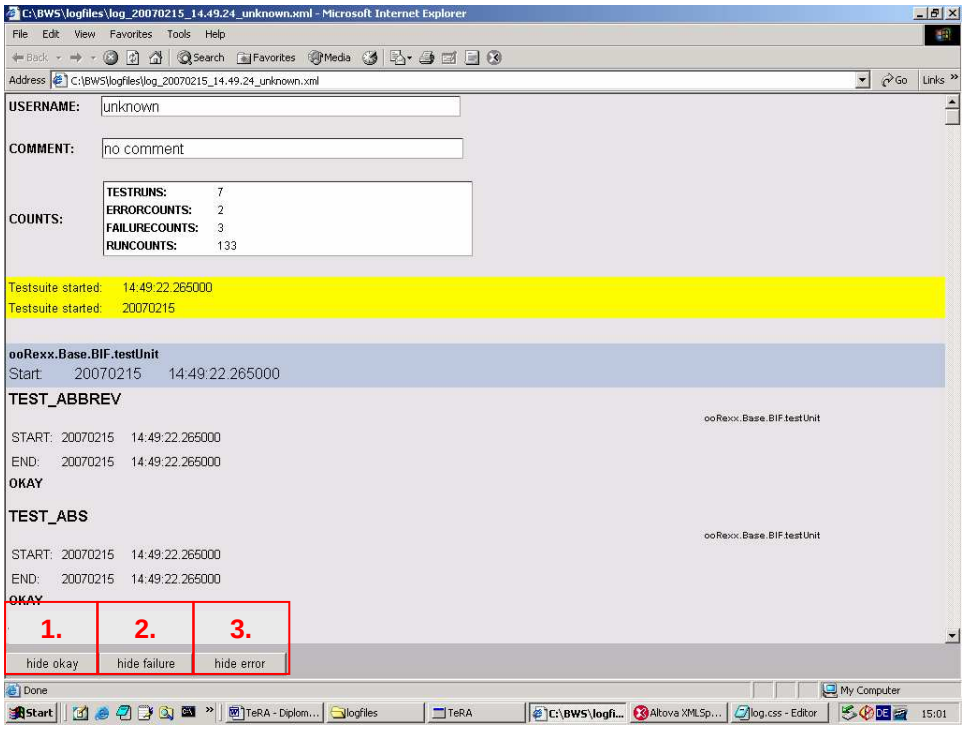

Figure 29: A Log File Example.

When opening a log file (XML transformed to HTML with LOG.XSL) there are three buttons in the footer:

- `hide okay /show okay`: shows or hides test cases, which did not raise a failure or an error. (**1.**)
- `hide failure/ show failure`: shows or hides test cases, which raised a failure. (**2.**)
- `hide error/show error`: hides or shows tests cases, which raised an error. (**3.**)

4.2.1.1.6.2 LOG.XSL

Within LOG.XSL there is the whole information about the transformation of the log files (XML files) into HTML files. [cp. chapter 7.7, "*Excursus: Extensible Style Sheet Language*", page 109]

Figure 30 shows how a log file looks like, if there is no XSL file used.

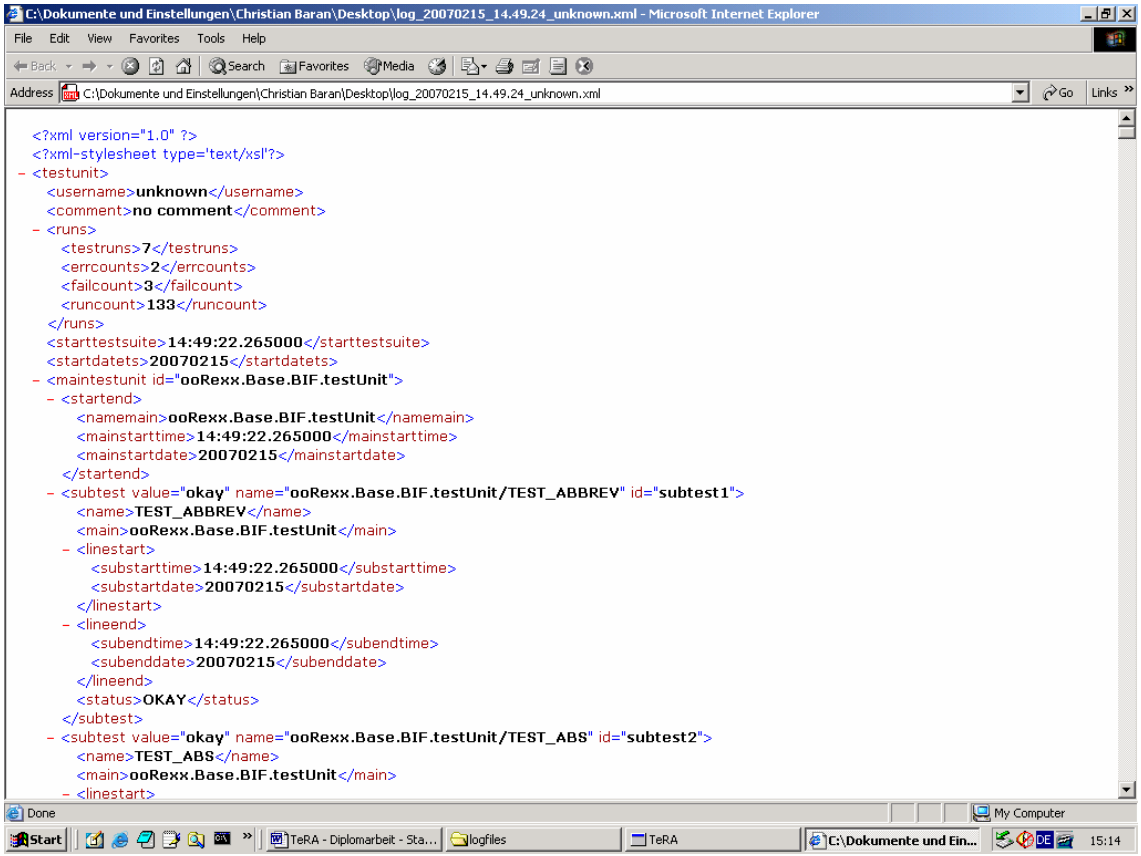

Figure 30: Log File (XML), without Using XSL.

Figure 31 shows the same log file as figure 30, but this time using a XSL file (LOG.XSL).

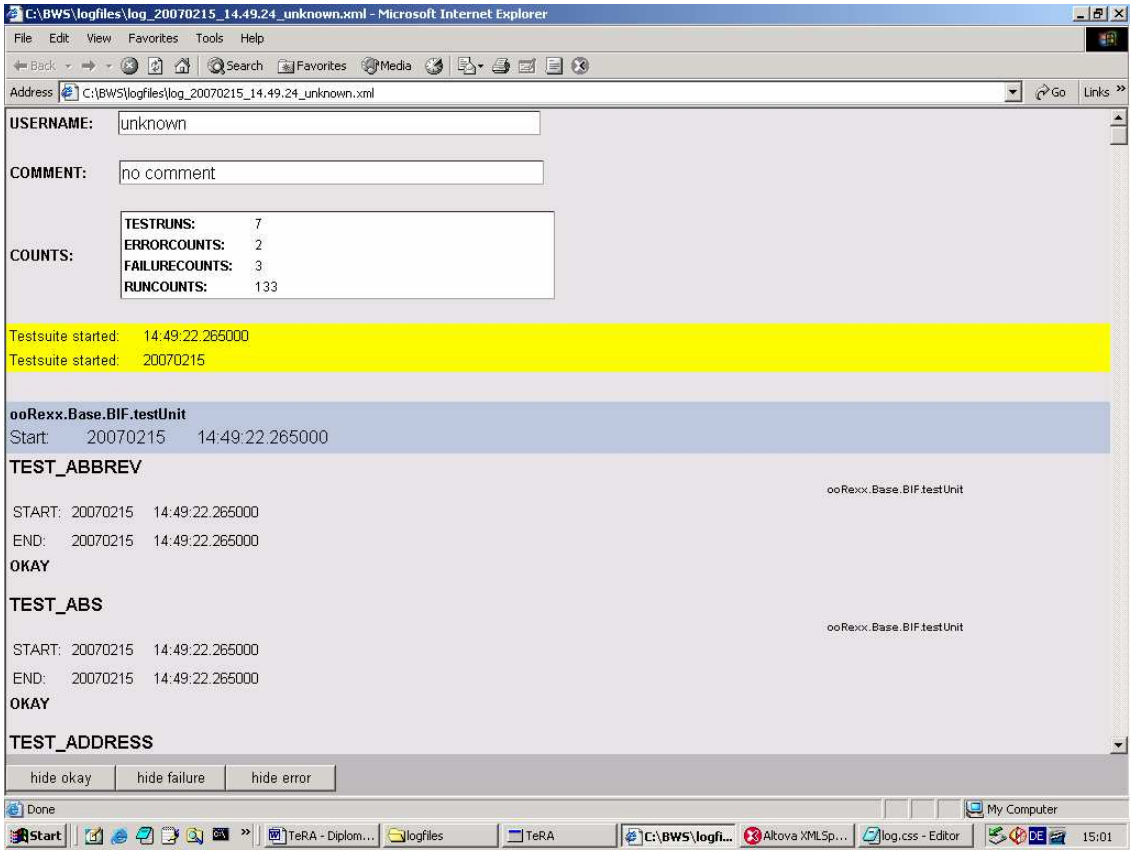

Figure 31: Same Log File (XML), Using XSL.

To get an idea, how failures look like in the log files, figure 32 shows a failure. This failure occurred in the test unit oorexx.base.class.Supplier, in the test case TEST_NEXT.

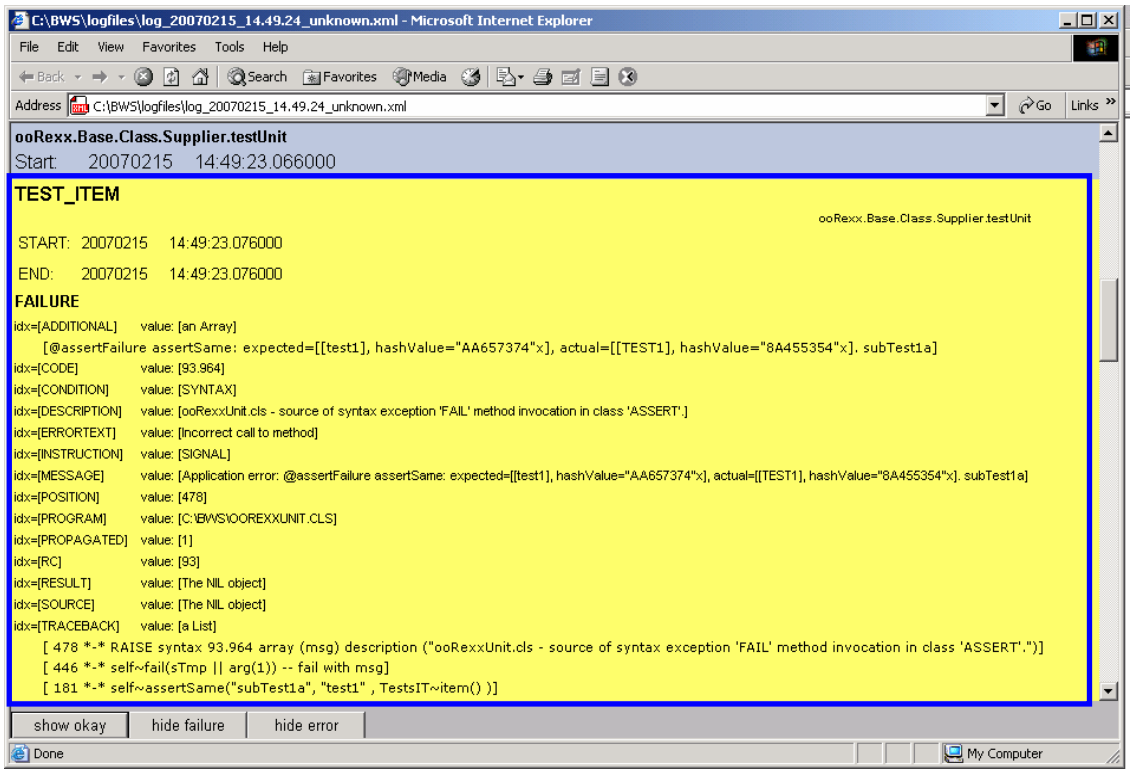

Figure 32: Log File (Failure).

Figure 33 shows an error in the test unit walter.test1.testUnit, which occurred in the test case `Test_TC1`.

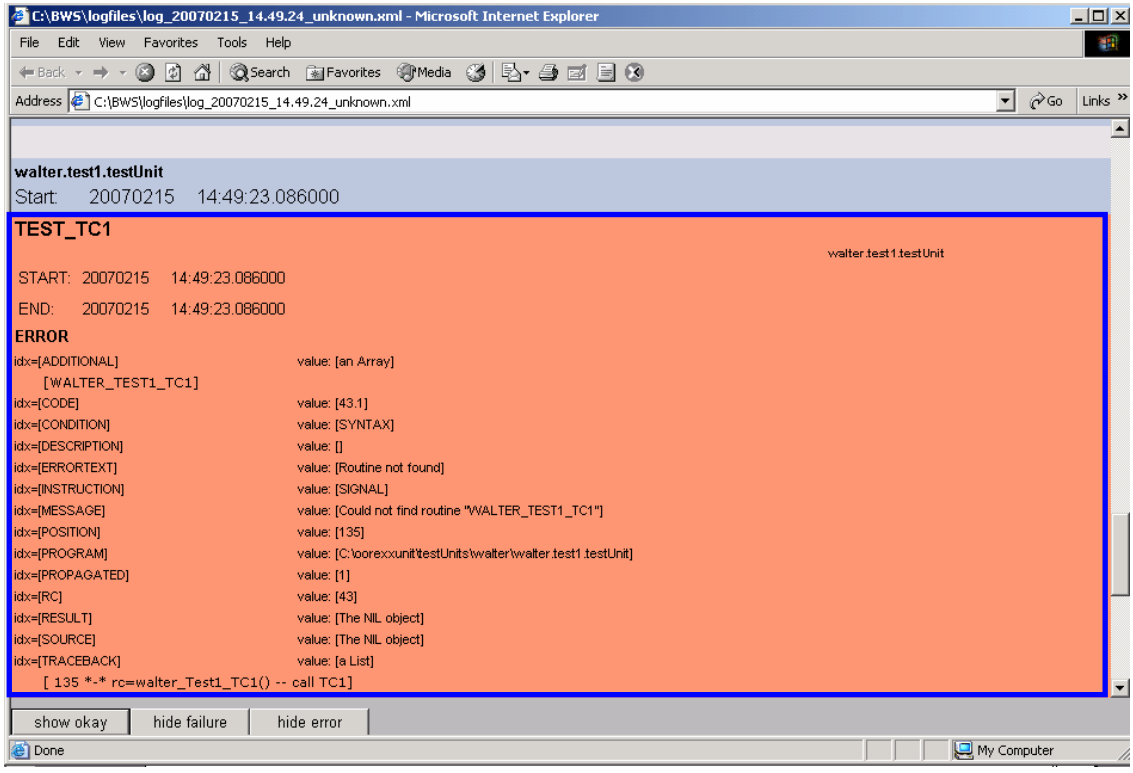

Figure 33: Log File (Error).

The whole source code of LOG.XSL can be found in chapter 7.8.5.1, (“*LOG.XSL*”, page 186).

To get an idea of using XML and XSL, there are two examples of simple XSL transformations subsequent (also used in LOG.XSL in a similar way).

Example 1: Simple XSL Transformation.

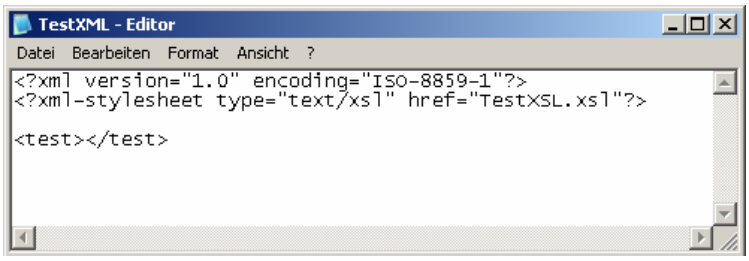

Figure 34: Example of a Simple XML File.

The XML file in figure 34 is very simple, because it only contains one element (`<test>`). The first line gives information about the file’s version and encoding. The second line is the link to the corresponding XSL file and the fourth line contains the XML tags.

Figures 34 and 35 show how XSL practically works and figure 36 shows the result of the transformation.

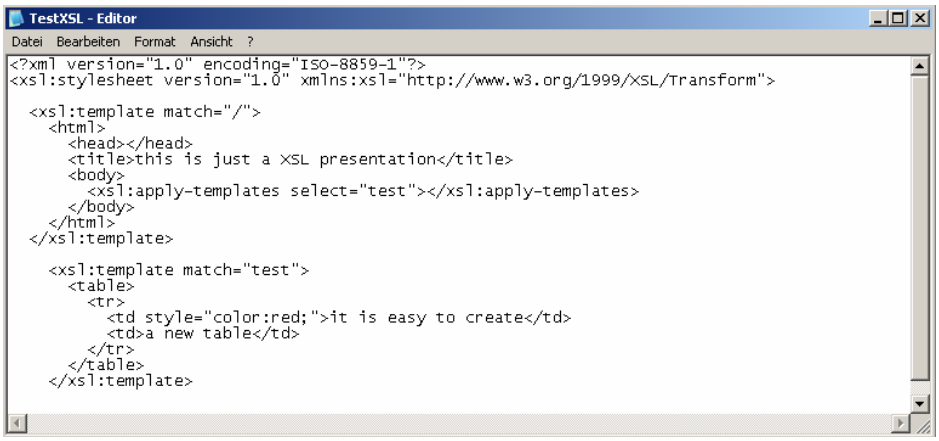

Figure 35: Example of a Simple XSL File.

The XSL file in figure 35 contains the structure for the XML file in figure 34.

- First of all, `<xsl:template match="/">` (figure 33) matches to the root node of the XML file, then the HTML structure (part after the `<xsl:template match="/">`) is inserted.
- The tag `<xsl:apply-templates select="test">` (figure 33) indicates that there is another template, which is used if there is a test-tag in the XML file. In this case a test-tag can be found. The program continues with `<xsl:template match="test">` and builds up the table (part after the template-tag).

Figure 36: Result (Table) of a Simple XSL Transformation (Example 1).

Example 2: Get text data.

With XSL it is also possible to get the text data (`<tagname>text-data</tagname>`) and import it into the transformed HTML file. This is important for TeRA to move tag-information from XML to HTML (result pages).

Figures 37 and 38 show how text is imported and figure 39 shows the result.

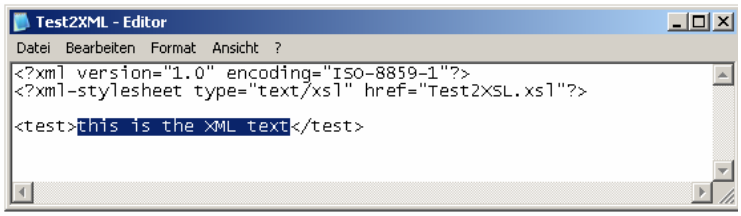

Figure 37: An XML File with Text Data.

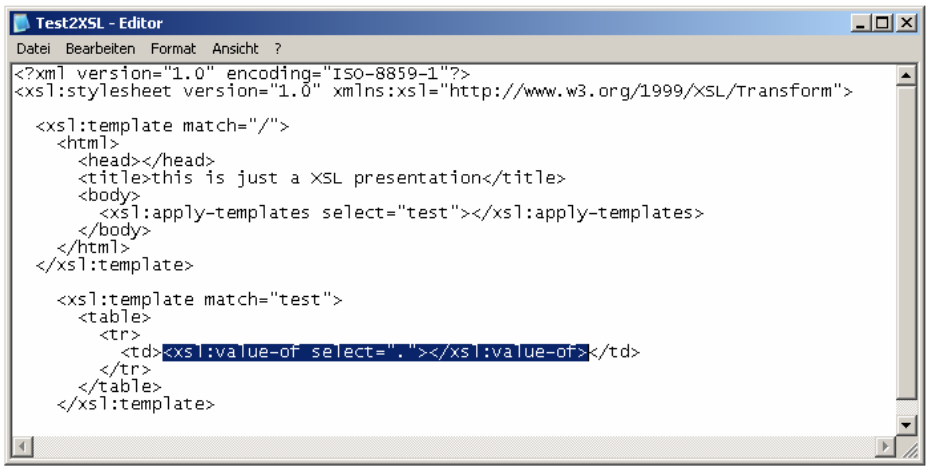

Figure 38: XSL – Get the Text of XML to HTML File.

The tag `<xsl:value-of select=".">` (figure 38) extracts the text of the test-tag and inserts it into the table data cell `<td>`.

Figure 39: Result (Table) – Get Text Data (Example 2).

To compare this, figure 40 shows the same XML file (figure 39), when the XSL file is not used.

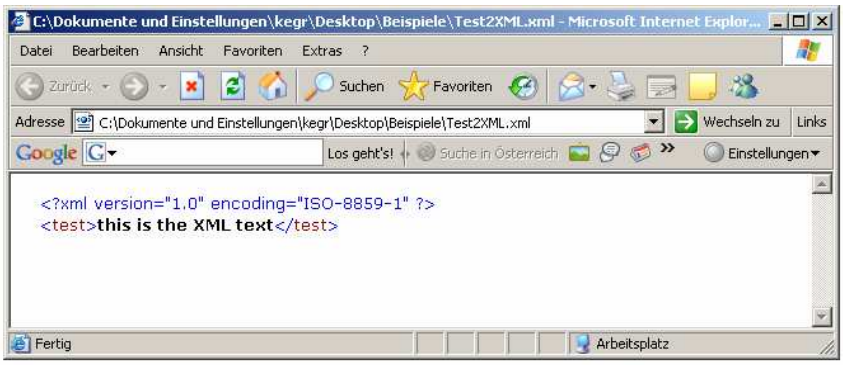

Figure 40: XML File, Not Using XSL.

With this quite simple proceeding it is easily possible to move tag-information from XML files to HTML (result pages).

4.2.1.1.6.3 LOG_XSL_SCRIPT.REX

The following routines are situated in LOG_XSL_SCRIPT.REX:

- **okay**: when pressing the corresponding button (hide okay / show okay, this routine changes the value of display directly into the corresponding Cascading Style Sheet. The tables of the test cases, which did not raise a failure or error, are then displayed or hidden, depending on their actual status.[1]
- **failure**: does the same as the okay routine, but for all tables of test cases, which raised a failure.
- **error**: does the same as the okay and failure routines, but for all tables of test cases, which raised an error.

The whole sourde code can be found in chapter 7.8.1.8 (“*LOG_XSL_SCRIPT.REX*”, page 146)

4.2.1.1.6.4 LOG.CSS

LOG.CSS is a Cascading Style Sheet, where the styling rules for the log files are stored.

---

[1] The alternative to display or hide test case tables was to change the display value of each table. As there can be masses of tables within one log file, this lasts too long to execute completely, because the routine has to loop through all tables of test cases, which in this case did not raise a failure or error and change the display value. To avoid this time consuming procedure a class attribute “okay” was added to the element tag, which has a styling rule in the corresponding style sheet. The task of the progam is now to gain access to the log file and change only this single styling rule. The XML file then inherits the changes from the matching CSS file (by reloading the page in the browser), which saves a lot of time.

Figure 41 shows a part of LOG.CSS.

```
.okay
{
        display:inline;
        background:#e0e0e0;
}

.failure
{
        display:inline;
        background:#FFFF6B;
}

.error
{
        display:inline;
        background:#FF9473;
}

.blank
{
        display:inline;
}

th
{
        width:15%;
}

.testsuite
{
        width:100%;
        background:yellow;
}

.testcase
{
        width:100%;
        background:#BDC6DE;
}

.space
{
        height:20px;
}
```

Figure 41: LOG.CSS.

The whole code can be found in chapter 7.8.2.2 (“*LOG.CSS*”, page 7.8.2.2).

4.2.2 TeRA-COMPARELOGS.REX

TeRA-COMPARELOGS.REX is comparable to TeRA.REX. It generates the HTA file TeRA-COMPARELOGS.HTA in almost the same manner.

Figure 42 shows all references of TeRA-COMPARELOGS.REX to other components of TeRA.

As TeRA.REX it provides the whole HTML structure for the HTA file (TeRA-COMPARELOGS.HTA).

Furthermore it sets up two different tables, where the already generated log files are inserted. This is done by searching for all files in the directory logfiles, using `*.xml` as searching criteria. The reason why there are two tables is that the log files in one table are sorted ascending and in the other table descending. The ascending sorted table is displayed when the HTA file is loaded.

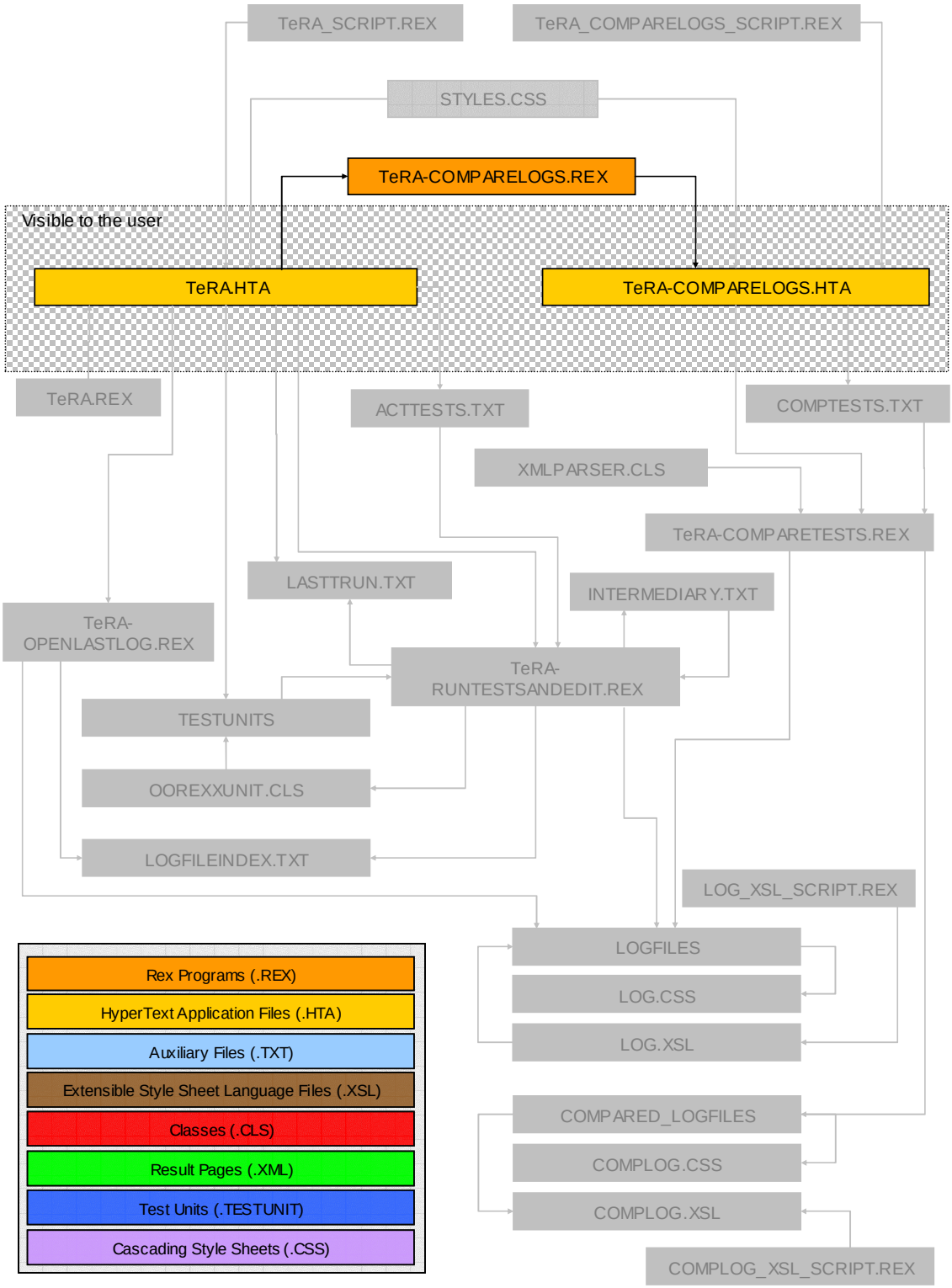

Figure 42: TeRA-COMPARELOGS.REX, Reference to Other Components.

Another task of TeRA-COMPARELOGS.REX is to allocate the buttons which are used to run the different routines (compare chapter 4.2.2.1.2, "*TeRA_COMPARELOGS_SCRIPT.REX*", page 69). This is done in the same

way as in TeRA.REX. The buttons are added to either footer1 (interaction with the HTA file) or footer2 (links, start external programs).

The ascending and descending sorted tables are situated in the content_container section.

The whole source code of TeRA-COMPARELOGS.REX can be found in chapter 7.8.1.5 (“*TeRA-COMPARELOGS.REX*”, page 131).

4.2.2.1 TeRA-COMPARELOGS.HTA

TeRA-COMPARELOGS.HTA is the second graphical user interface of the TeRA test runner. Its task is to provide navigation to facilitate the log file comparison.

Figure 43 shows the references of TeRA-COMPARELOGS.HTA to other parts of TeRA and figure 44 the user interface TeRA-COMPARELOGS.HTA.

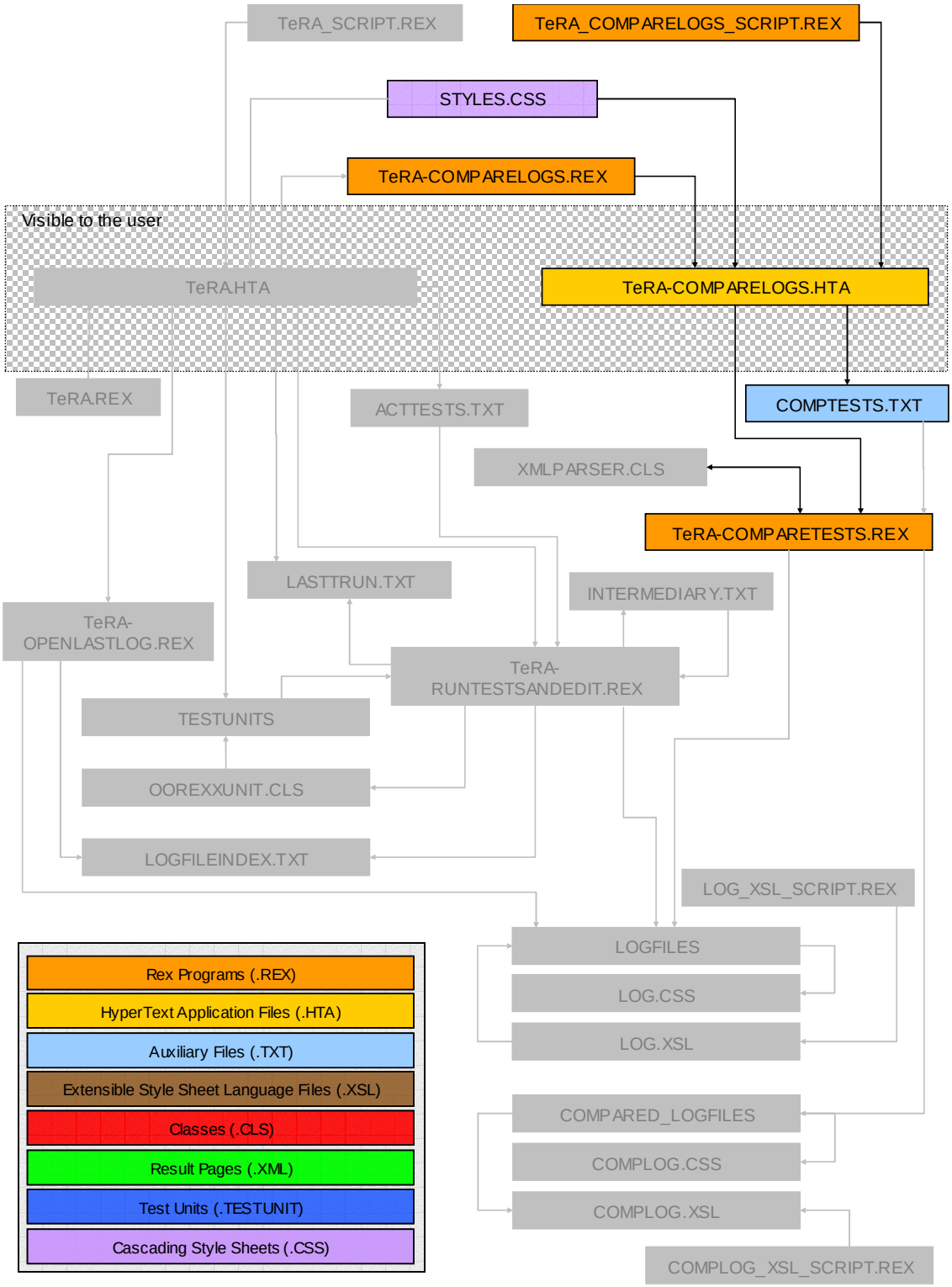

Figure 43: TeRA-COMPARELOGS.HTA, Reference to Other Components.

This HTA file is build up of three sections.

The first section is the HTML body (id: `content_container` – highlighted blue in figure 44), where all log files of the run test units are listed within tables, to allow to select them for comparison. There are two tables, one where the test

units are sorted in ascending order (by date and time) and the other one sorted descending (by date and time). When loading TeRA-COMPARELOGS.HTA the ascending sorted table is displayed, while the other table is hidden. When clicking the `descending` button, the descending sorted table is displayed and the ascending table is hidden.

The second section is the HTML body section (id: `footer1` – highlighted in green in figure 44)

The third is the `footer2` section (highlighted in orange in figure 44), where the buttons for starting the routines can be found.

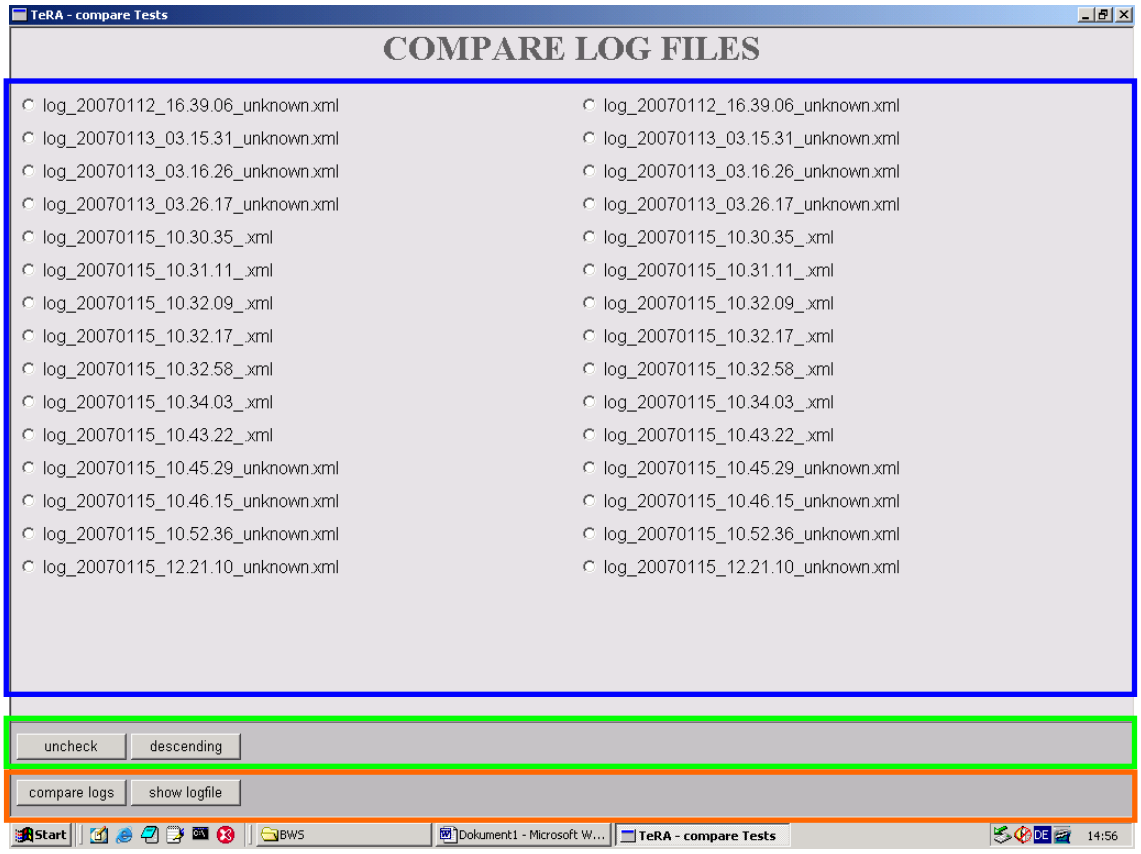

Figure 44: TeRA-COMPARELOGS.HTA (Ascending Tables are Displayed).

Contrary to TeRA.REX, there are radio buttons to select the log files. The advantage of radio buttons in this context is that there can only be one radio button selected within a list. It is not possible to select two radio buttons of one list at the same time. This is quite useful, because the users are supposed to select only one log file from the first list and the second log file from the second

list. With the radio buttons there can only be two log files selected (one in each list) at the same time. [cp. figure 46]

Another problem is the possibility to select the same log files from each list. To avoid that the same log file is compared, which would make no sense, because the log file is physically identical and therefore would not show any changes or differences, the corresponding radio button is disabled, when selecting a log file in one table (e.g. if one log file is selected in table 1.1 the corresponding log file – radio button – in table 1.2 is disabled). [cp. figure 46] The disabling of the radio button is done by the routine checked (compare chapter 4.2.2.1.2, "*TeRA_COMPARELOGS_SCRIPT.REX*", page 69).

4.2.2.1.1 TeRA-COMPARELOGS.HTA – User Interface

Figure 45 shows the second user interface TeRA-COMPARELOGS.HTA.

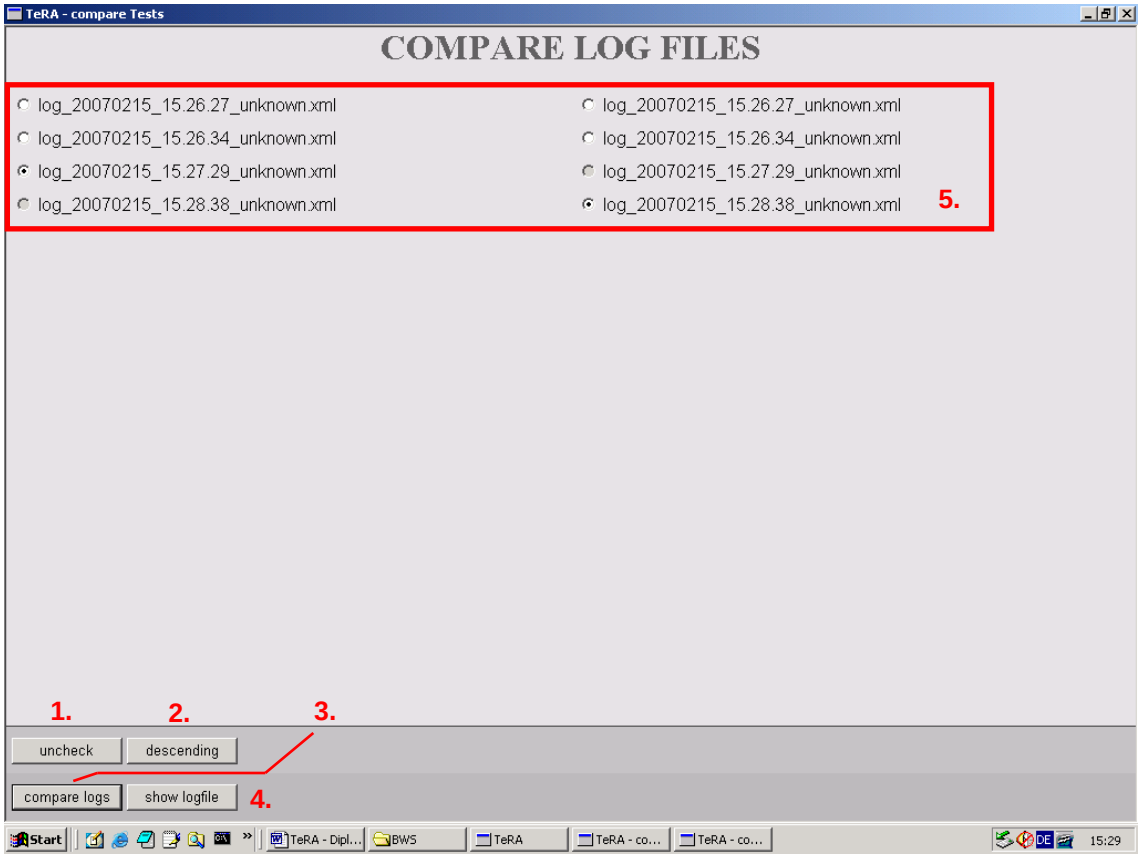

Figure 45: TeRA-COMPARELOGS.HTA – User Interface.

- The button `uncheck` (**1.**) is used to uncheck all radio buttons (**5.**). [cf. figure 45]
- The button `descending/ascending` (**2.**) is used to sort the radio button list (**5.**) ascending or descending. [cp. figure 45]
- The button `compare logs` is used to compare the selected log files (**3.**). [cp. figure 45]
- The `show logfile` button opens a selected log file (**4.**). [cp. figure 45]

4.2.2.1.2 TeRA_COMPARELOGS_SCRIPT.REX

Within this program, there are all important routines for TeRA-COMPARELOGS.HTA.

The following routines can be found:

- **showLogFile**: is used to show the content of a selected log file (selected in TeRA-COMPARELOGS.HTA) by getting the value (path) of the element and opening it.
- **checked**: if a radio button of a log file (in table 1.1) is selected, the radio button of the corresponding log file (in table 1.2) is disabled (can not be selected). The reason of this proceeding is that it would make no sense to compare a log file with itself (no changes). [cp. figure 46]
- **uncheckAll**: unchecks all selected radio buttons by looping through all id attributes and deselecting the corresponding radio buttons.
- **arrange**: is used to show or hide the ascending or descending table by setting the table's display rule to `inline` (show) or `none` (hidden). This routine is invoked by clicking the ascending/descending button.
- **compareLogs**: gets the value (path) of the log files, which should be compared and stores the paths to an auxiliary file called COMPTESTS.TXT situated in the directory TeRA_files.

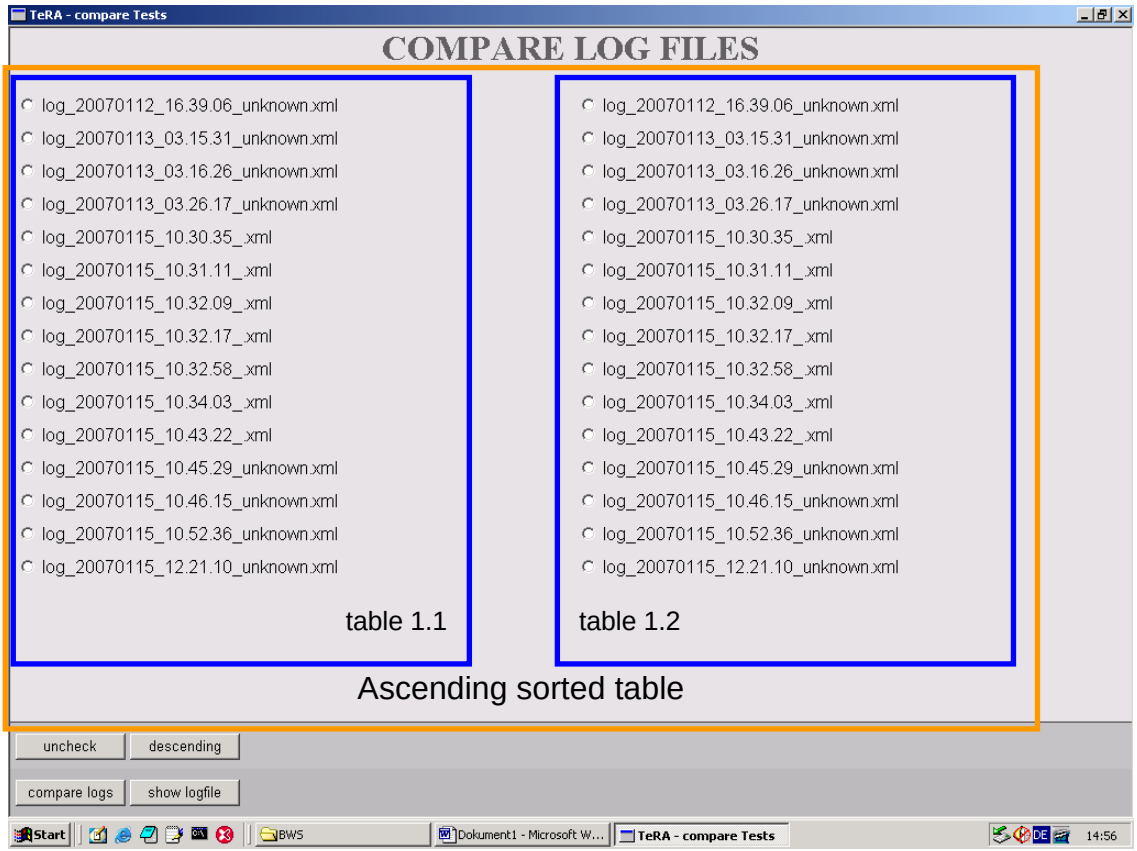

Figure 46: Ascending Sorted Table.

Figure 47 shows an example of the content of COMPTESTS.TXT.

Figure 47: COMPTESTS.TXT.

The whole source code of TeRA_COMPARELOGS_SCRIPT.REX can be found in chapter 7.8.1.6 (“*TeRA_COMPARELOGS_SCRIPT.REX*”, page 134).

4.2.2.1.3 TeRA-COMPARETESTS.REX

TeRA-COMPARETESTS.REX is the part of the TeRA test runner, which compares log files.

Figure 48 shows the references to the other components of TeRA.

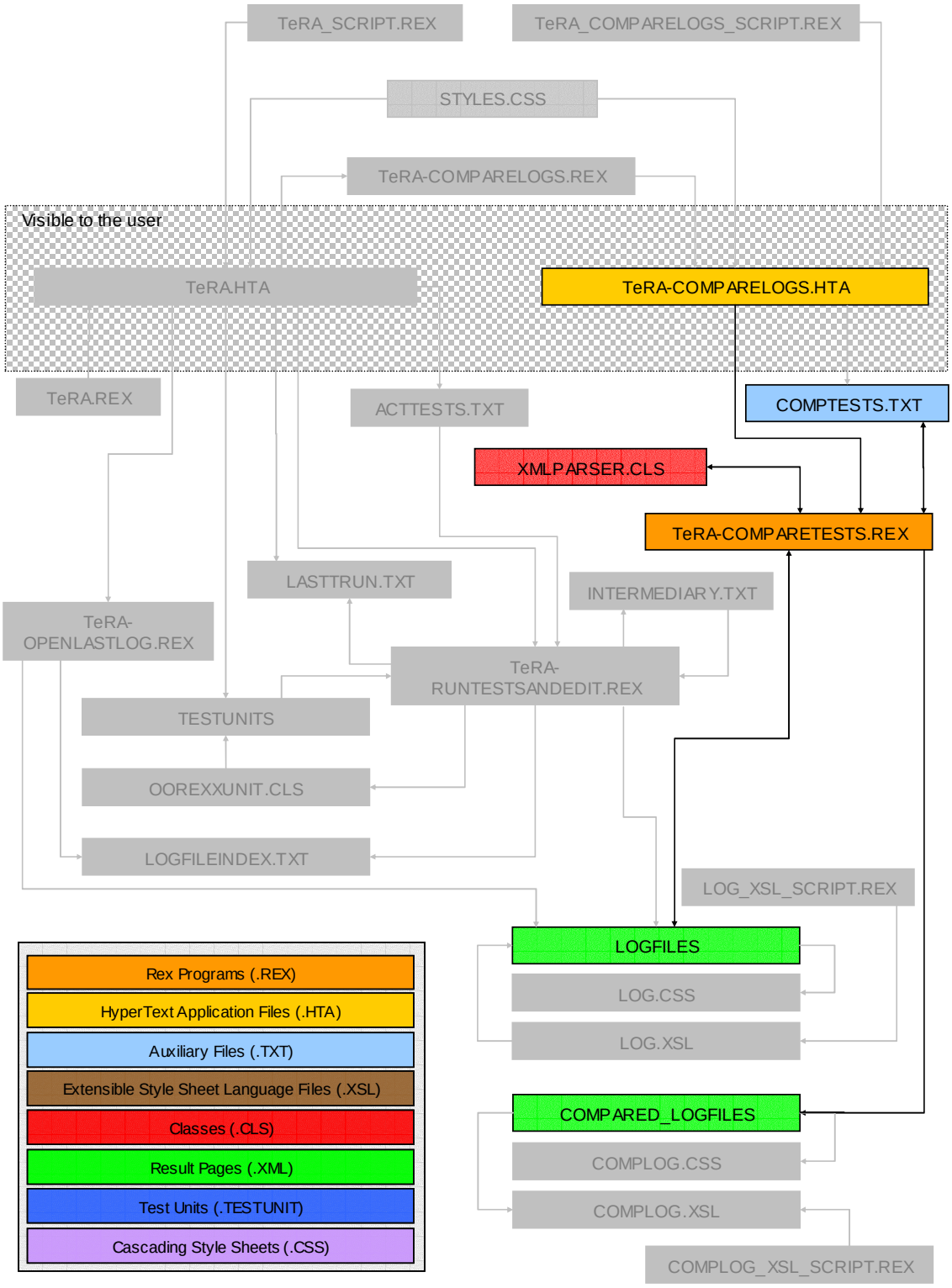

Figure 48: TeRA-COMPARETESTS.REX, Reference to Other Components.

The file name of compared log files is put together in the following way:

- `Logfilename1__Logfilename2` (`logfilename1`: is the name of the log file, which is selected in TeRA-COMPARELOGS.HTA in the table 1.1 – ascending sorted – or 2.1 – descending sorted; `logfilename2`: is the

name of the log file, which is selected in TeRA-COMPARELOGS.HTA in the table 1.2 – ascending sorted – or 2.2 – descending sorted). [cp. figure 46]

To assure that no log files combination is compared twice the program reads the data, which is stored in COMPTESTS.TXT (two log file paths) and combines the log file names in two ways:

- `Logfilename1__Logfilename2` and
- `Logfilename2__Logfilename1`.

Assumption: the log file A was already compared with log file F (the compared log file name would be A__F). A new comparison will be executed, where log file F shall be compared with log file A (the compared log file name would be F__A). The results of this comparison will be the same as A__F, but in a different order (first F and then A). To assure that this comparison will not be executed, the program checks both name combinations and will find A__F and open it. As there will be quite a lot of test units in the future, this saves a lot of time, because the new comparison will not be executed, but the already existing file will be opened.

If there is no file with one of these name combinations existing, the program creates a new file. The compared log files are XML files.

4.2.2.1.3.1 Compared Log Files – User Interface

There are three sections within a compared log file:

- `Changes`: contains all changes, which are found when comparing two log files or, if there are no differences found, the string "`no changes found`".
- `Log1`: lists all test cases, which run in the first log file.
- `Log2`: contains all test cases, which run in the second log file.

Figure 49 shows an example of a compared log file.

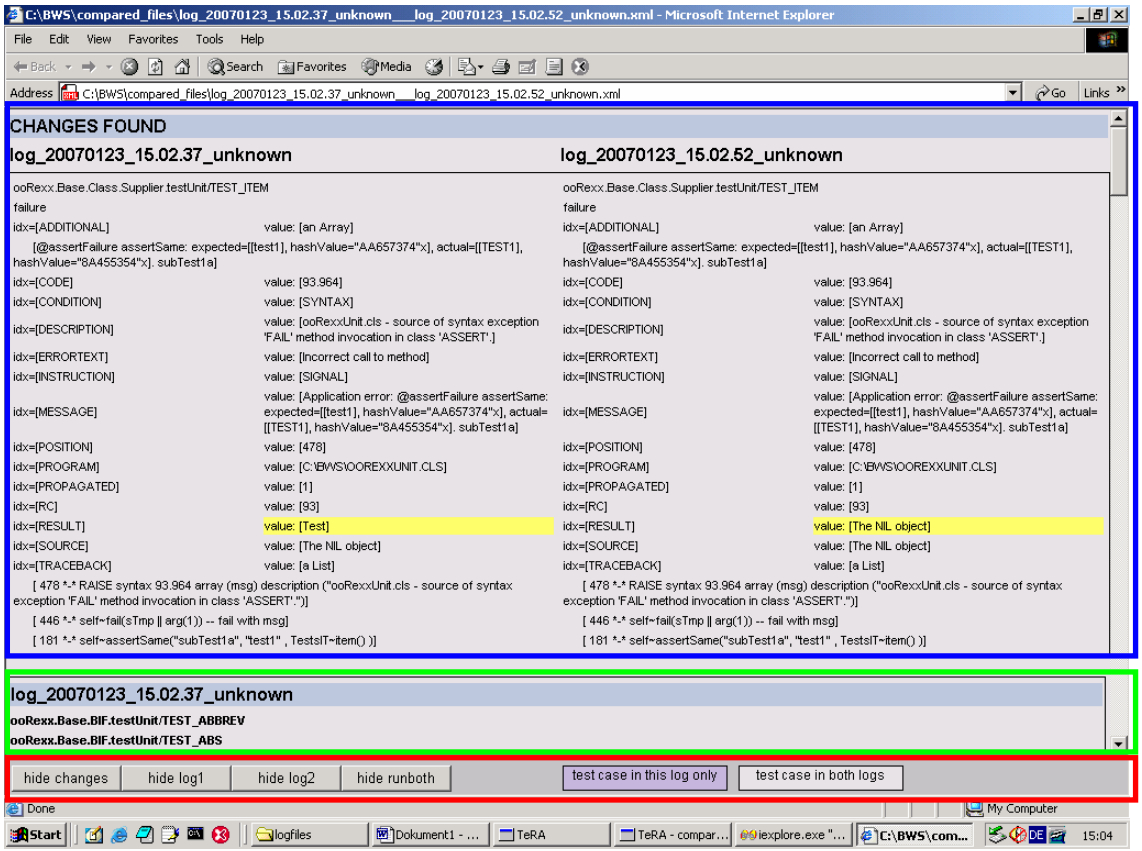

Figure 49: Example of a Compared Log File (Blue: Section `Changes`, Green: Section `Log1`, Red: `footer`).

- The button `hide changes/show changes` (cp. figure 49 - highlighted in red) shows or hides the section `Changes`.
- The button `hide log1/show log1` (cp. figure 49 - highlighted in red) hides or shows the section `Log1`.
- The button `hide log2/show log2` (cp. figure 49 - highlighted in red) hides or shows the section `Log2`.

Figure 50 shows the Sections `Log1` and `Log2` of a compared log file.

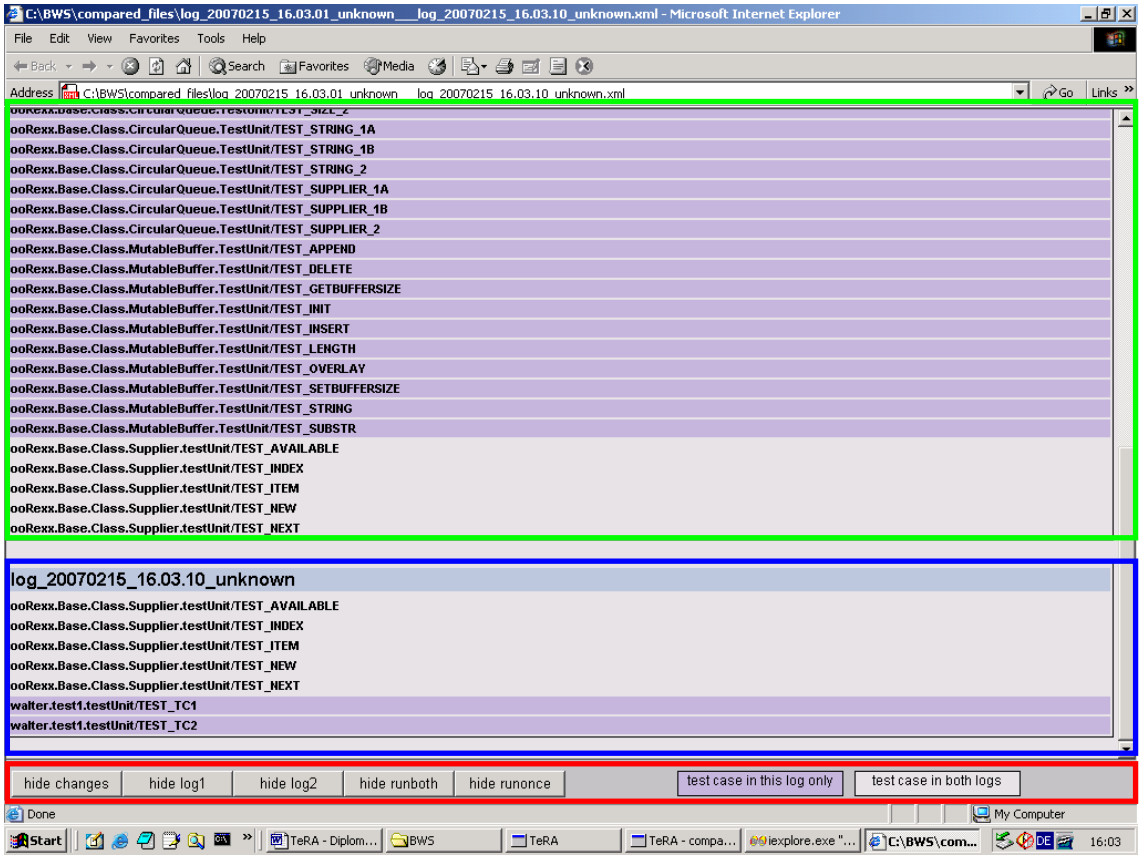

Figure 50: Compared Log File – Sections `Log1` and `Log2` (Green: Section `Log1`, Blue: Section `Log2`, Red: `footer`).

The section `Log1` contains the test case names of the test cases, which were tested in the test run that created the first log file (compare chapter 4.2.1.1.5, "*TeRA-RUNTESTSANDEDIT.REX*", page 45). The same applies to the section `Log2`.

If there are test cases in `Log1`, which are also contained in `Log2`, they are called `runboth` (not highlighted) in the course of TeRA.

If there are test cases in one log-section, which are not contained by the other section, they are called `runonce` (highlighted in magenta).

- The `hide runboth/show runboth` button (cf. figure 50 - highlighted in red) hides or shows the test cases, which are part of both log-sections.
- The `hide runonce/show runonce` button (cf. figure 50 - highlighted in red) hides or shows the test cases, which are part of only one log-section.

4.2.2.1.3.2 Comparison

The program starts parsing the log files (XML files) using the XMLPARSER class (XMLPARSER.CLS).

The extracted data (e.g. test case names, time test case started/ended, failure, error, okay, error information, failure information, etc.) is then stored in two arrays (`attrarr1` for the first and `attrarr2` for the second log file – these two arrays are part of TeRA-COMPARETESTS.REX). Then the program compares the data of these two arrays by checking one entry in the first array and then looping through the other array, searching for a matching counterpart. If the counterpart is found and the information is equal, then the next entry in the first array is checked. If there are no items in the first array left, then the same proceeding is repeated, but this time with the items of the second array (searching in the first array for counterparts). If all test cases of the two arrays contain identical information, the program adds the string `no changes found` to the `Changes` section.

If the program finds matching test cases (test case names) in the two arrays (`attrarr1` and `attrarr2`), which have different test run information (e.g. status, error information, failure information etc.), the program adds the whole information of these test cases to the `Changes` section (log file 1 on the left and log file 2 on the right side – and the parts which differ are highlighted in yellow in both log files). [cp. figures 49 and 51]

The sections `Log1` and `Log2` show all test cases which are included in the corresponding log file (e.g. log file 1 contains information about the test cases test1 and test2, then the section `Log1` lists all test case names – test1 and test2).

These two sections (`Log1` and `Log2`) can contain different test cases as well. Hence, test cases, which are included in both log files are not highlighted at all, whereas test cases, which are only included in one log file are highlighted in magenta ( ). [cp.figure 52]

Figures 51 and 52 show a compared log file.

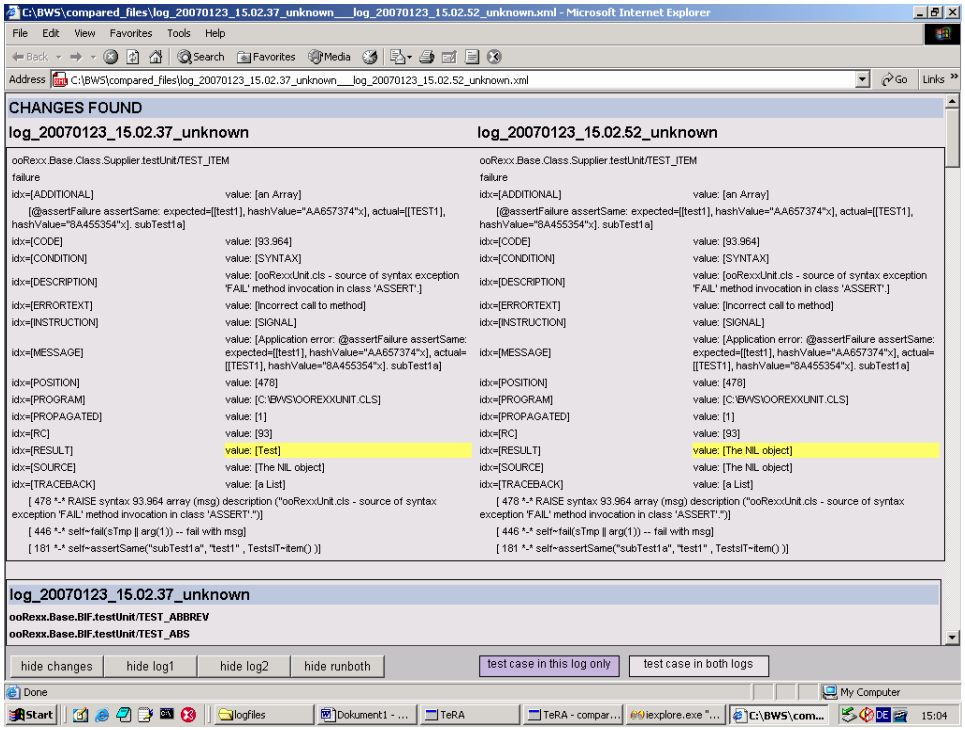

Figure 51: Compared Log File (`Changes` Section – Differences Highlighted in Yellow).

Figure 52 shows the sections `Log1` and `Log2` in a compared log file.

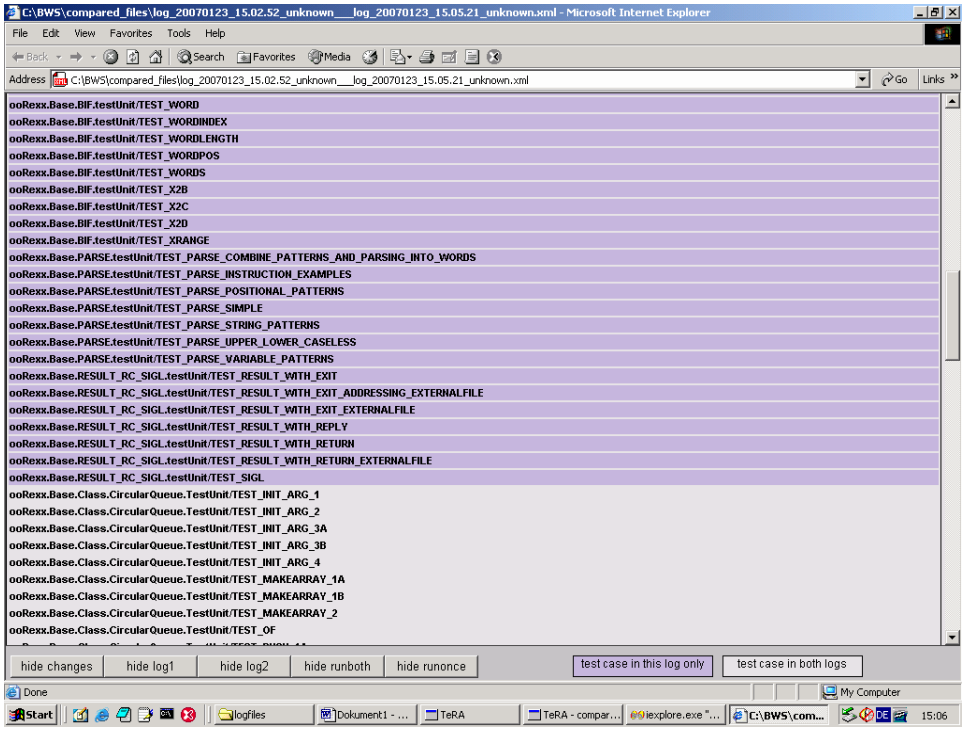

Figure 52: Compared Log File (Magenta: `runonce`, Not Highlighted: `runboth`).

The sections `Log1` and `Log2` can contain two different kinds of test cases:

- **runboth**: these test cases have been tested in both test runs (listed in both log files and not highlighted).
- **runonce**: these test cases have been tested in one single test run (i.e. these test cases are listed in one log-section, but not in the other one – highlighted in magenta).

Figure 52 shows a compared log file, where both `runonce` and `runboth` test cases can be found.

4.2.2.1.3.3 The XMLPARSER Class

The XMLPARSER class was created by David Ashley. [cp. Ashl07]

Its task is to strip XML files by tags and extract the information – tag names and attributes as well as text.

The whole source code can be found in chapter 7.8.3.2 (“*XMLPARSER.CLS*”, page 175).

The advantage of using the XMLPARSER.CLS is that TeRA can use every method of the parser class, without parsing the XML files by itself (this saves a lot of time when parsing XML files and a lot of source code lines as well). The XMLPARSER.CLS does the parsing and the program TeRA-COMPARETESTS.REX gets the results and handles them.

The example (code 3 and 4) at the end of this chapter will show how the XMLPARSER works.

Figure 53 shows the architecture of the XMLPARSER class.

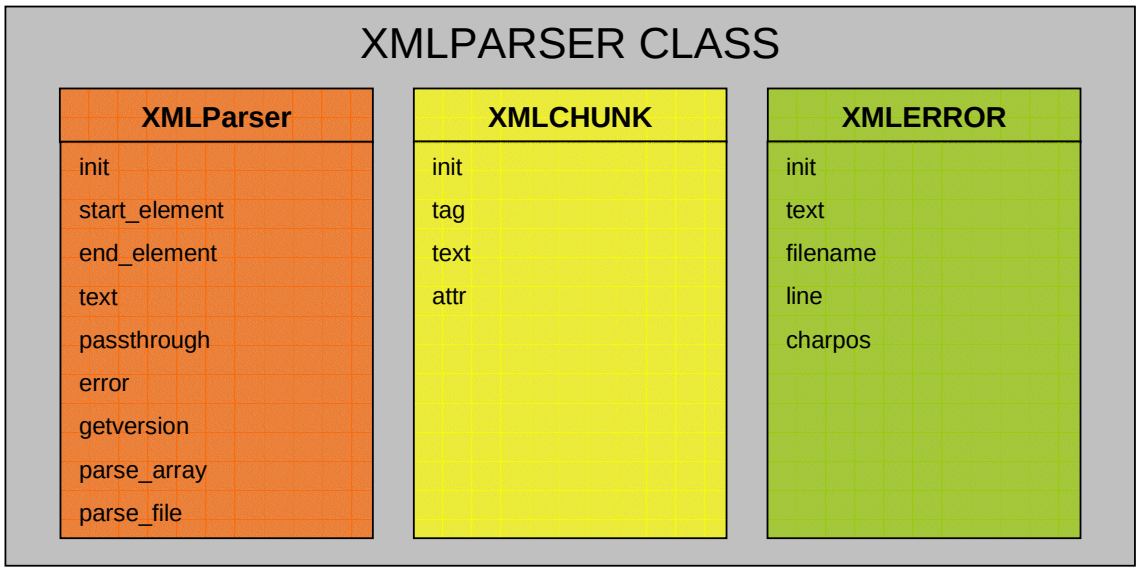

Figure 53: XMLPARSER Architecture.

The XMLPARSER class contains the following methods:

- **init**: instance initialization. Create new instance of the XMLPARSER class.
- **start_element**: invoked when a start element tag has been encountered.
- **end_element**: invoked when an end element tag has been encountered.
- **text**: invoked when character data has been encountered.
- **passthrough**: invoked when comment tag or a processing instruction has been encountered.
- **error**: invoked on an error.
- **getversion**: returns the version of this class.
- **parse_array**: parse the specified array of XML code.
- **parse_file**: parse the specified file of XML code.

The XMLCHUNK class contains the following methods:

- **init**: instance initialization. Creates a new instance of XMLCHUNK class.
- **tag**: extracts the XML tag name.
- **text**: extracts the XML text.

- **attr**: extracts the XML tag attributes.

The XMLERROR class contains the following methods:

- **init**: instance initialization. Creates a new instance of XMLERROR class.
- **text**: contains the error message text.
- **filename**: contains the XML file name.
- **line**: contains the error's line number.
- **charpos**: contains the error's character position.

The author explains in [Ashl07]: "The XMLPARSER class is a very simple parser for XML files. It is not an official 100% compatible XML parser because it has a lot of limitations, most of which you could probably care less about.

- The parser only understands ASCII, which is a valid subset of UTF-8. It does not understand any other encoding except 8-bit ASCII.
- It does not test that the document is well-formed. It assumes that the document is a well-formed XML document.
- The parser does not know how to handle XML processing instructions. It passes those instructions through the passthrough method intact so that the user can try to make sense of them.

To use the XMLPARSER you need to be aware of the following.

- The parser uses a SAX[1]-like interface, but methods of the class are used instead of a call-back mechanism. The default methods perform no actions. The user will need to subclass the XMLPARSER class and override the call-back methods in order to insert their own actions for each XML chunk type.

[1] SAX (**S**imple **A**PI for **X**ML) is a serial access parser API for XML, which provides a mechanism for reading data from an XML document. It is a popular alternative to the Document Object Model (DOM).

- The call-back methods use the XMLCHUNK class to pass data to the methods. This is a very simple class and is used as a container for specific types of XML chunks.

- Text chunks (CDATA) are passed through the text method intact. If the parser encounters multiple lines of text it invokes the text method for each line individually. It also does not collapse white space chars.

- XML tags are collapsed. This means that if a tag crosses a line boundary then the lines are collapsed together. This is important for processing instruction tags, comment tags and other special tags."

To get an idea of how the XMLPARSER works, the following example (cp. code 3 and 4) will explain some parts, which are of importance to TeRA.

EXAMPLE: Parse a simple XML file.

```
<?xml version="1.0"?>
<list>
    <entry>
        <programming_language>Java</programming_language>
        <unit_test>JUnit</unit_test>
    </entry>
    <entry>
        <programming_language>.NET</programming_language>
        <unit_test>NUnit</unit_test>
    </entry>
    <entry>
        <programming_language>Pearl</programming_language>
        <unit_test>PearlUnit</unit_test>
    </entry>
</list>
```

Code 3: A Simple XML File.

Code 4 shows a Rexx program, which uses the XMLPARSER.CLS to parse the XML file shown in code 3.

```
parser = .myparser~new()
errortxt = parser~parse_file('test1.xml')

return

::requires 'xmlparser.cls'


::class myparser subclass xmlparser

::method mystream attribute
::method programming_language attribute
::method unit_test attribute
::method curtag attribute

::method start_element
  use arg chunk
  self~curtag = chunk~tag
  select
     when chunk~tag = 'list' then do
        /* open the output file */
        self~mystream = .stream~new('UNIT_TESTING.TXT')
```

```
        status = self~mystream~open('write replace')
        end
     when chunk~tag = 'entry' then do
        /* create a new record */
        self~programming_language = ''
        self~unit_test = ''
        end
     otherwise nop
     end
  return

::method end_element
  use arg chunk
  select
     when chunk~tag = 'list' then do
        /* close the output file */
        status = self~mystream~close()
        end
     when chunk~tag = 'entry' then do
        /* create the record */
        record = '"' || self~programming_language~changestr('"', '""') ||'"'
        record = record || ',"' || self~unit_test~changestr('"', '""') ||'"'
        /* write the record */
        self~mystream~lineout(record)
        end
     otherwise nop
     end
  return

::method text
  use arg chunk
  select
     when self~curtag = 'programming_language' then self~programming_language = -
                                                    chunk~text~strip()
     when self~curtag = 'unit_test' then self~unit_test = chunk~text~strip()
     otherwise nop
     end
  return

::method error
  use arg err
  say err~text
  return
```

Code 4: Rexx Program Using the XMLPARSER.CLS.

The first line in code 4 creates a new object of the class `myparser`, which subclasses XMLPARSER. The second line gets the information about the XML file, which is going to be parsed. The method `start_element` is searching for the matching start tag within the XML file (`chunk~tag` is the tag name) and opens a new stream (`mystream`). The method `end_element` is searching for end tags matching the given strings (`list` and `entry`). If there is a closing tag (`</list>`), the stream is closed. If there is a closing tag (`</entry>`), a new record is added to the stream and stored to the new generated file UNIT_TESTING.TXT (cp. figure 49). The method `text` adds the text values (between the tags) to the record.

The order of the methods in code 4 is negligible, because the `end_element` method is executed, if the parser finds an end tag (e.g. `</programming language>`). In this case the `start_element` method is started when the parser

finds the starting tags `<list>` and `<entry>`, then the method `text` adds the text to the record and finally, the `end_element` method finds the tags `</entry>` and `</list>`,stores the record to the file and closes the stream.

Figure 54 shows the result of the example (code 3 and 4)

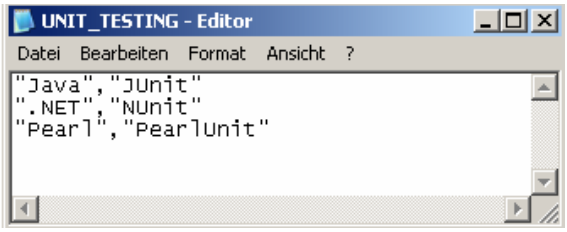

Figure 54: Result of the Rexx Program (Code 4).

4.2.2.1.3.4 COMPLOG.CSS

COMPLOG.CSS is the file, where all style rules for the compared log files are situated.

Figure 55 shows a part of the Cascading Style Sheet COMPLOG.CSS.

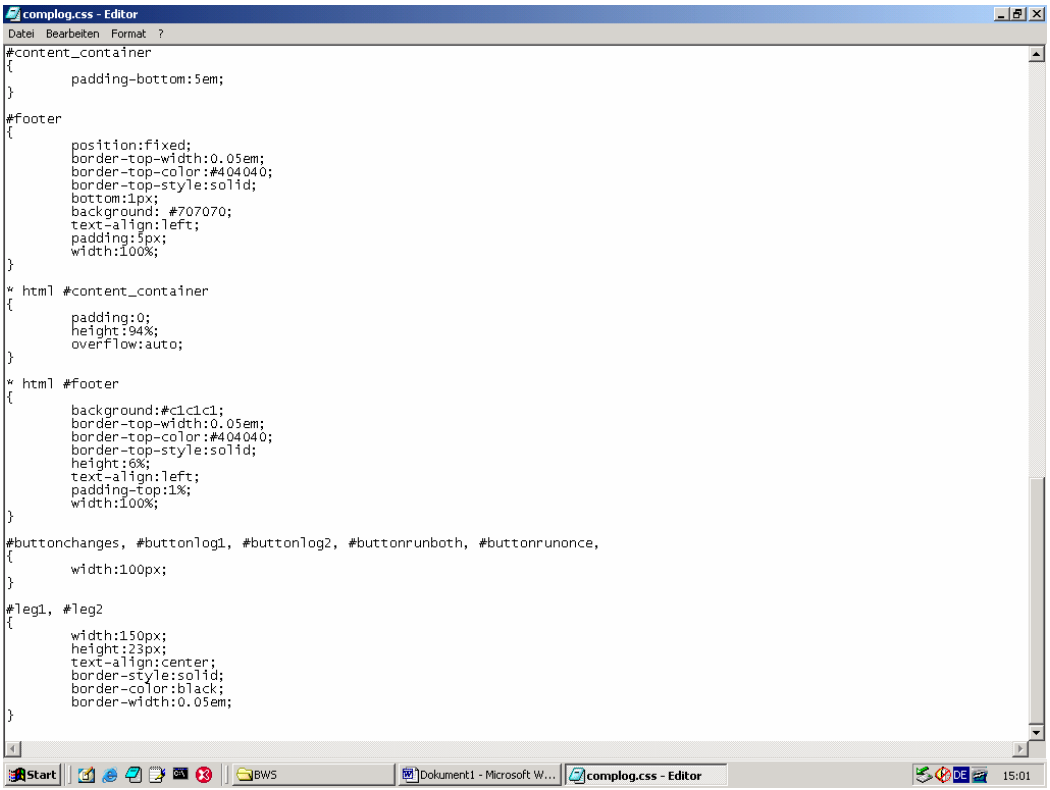

Figure 55: A Part of COMPLOG.CSS.

The whole source code of COMPLOG.CSS can be found in chapter 7.8.2.3 (“*COMPLOG.CSS*”, page 157).

4.2.2.1.3.5 COMPLOG.XSL

COMPLOG.XSL is the XSL file for the compared log files. It contains the structure for the XSL Transformation from XML to HTML. The routines for the transformed HTML file are situated in the file COMPLOG_XSL_SCRIPT.REX.

The whole source code can be found in chapter 7.8.5.2 (“*COMPLOG.XSL*”, page 191).

4.2.2.1.3.6 COMPLOG_XSL_SCRIPT.REX

The following routines are part of COMPLOG_XSL_SCRIPT.REX:

- **buttons**: checks if there are test cases within the`Log1` and `Log2` sections, which are only in one section (`runonce`) or in both sections (`runboth`). Depending on the result the `runonce` and `runboth` buttons are displayed or hidden (i.e. if there are no test cases – `runboth`, run in both log file test runs – the `runboth` button will not be displayed).
  `runonce` button: to hide or show test cases, which are only part of one log file section.
  `runboth` button: to hide or show test cases, which are part of both log file sections.
  For example: If there are exactly the same test cases in both log file sections, the `runonce` button is unneeded (because there are no test cases, which are only contained in one log-section) and therefore hidden. If there are only test cases in `Log1`, which are not part of `Log2` and the other way round, the `runboth` button is hidden (because there are no test cases, which are in both log-sections).
- **changes**: allows to hide or show the `Changes` section by changing the style rule of the corresponding table directly within the matching style sheet (COMPLOG.CSS).
- **log1**: hides or shows the whole section `Log1` (Log-section: defined in COMPLOG.CSS and corresponding compared log file).

- **log2**: hides or shows the whole section `Log2`.
- **runboth**: displays or hides the test cases, which are part of both log file sections.
- **runonce**: displays or hides the test cases, which are part of only one log file section.

The whole source code can be found in chapter 7.8.1.9 (“COMPLOG_XSL_SCRIPT.REX”, page 147.

5 Roundup and Outlook

Software engineering and especially software testing is getting more and more important. This is due to the fact that

- software systems are getting larger,
- software systems are getting more complex,
- some software systems are vital (i.e. software for hospitals),
- software reliability is increasingly important,
- staff resources particularly in quality management are short and
- testing of software and especially the automation of testing is of particular importance.

As programming languages are the fundamental building blocks of every software, there need to be especially accurate testing tools (i.e. if the programming language does not work properly, the created software will not work in the desired way as well).

ooRexxUnit is such a unit testing tool (framework) for Open Object Rexx.

ooRexxUnit was created to allow the users (developers) to test all parts of the programming language Open Object Rexx.

TeRA is a graphical test runner for ooRexxUnit to run and analyse test units for ooRexx. It is intended to facilitate the actual testing for the users as well as arranging and highlighting the results. Furthermore, there is the possibility to compare log files to easily analyse the results of different test runs and identify potential improvements or worsenings faster.

TeRA test runner application should be an assistance for everybody who is working with Open Object Rexx.

To increase the usability of TeRA for the users, XML, XSL and HTML were used in the course of creating the test runner application. Therewith it is possible to change most parts of TeRA, just with knowledge in XML, XSL and HTML.

This notably applies to the XML parser and all result pages (log files and compared log files). With the use of these languages the reusability and further development will be easier.

As TeRA is designed in a way that the reusability of its components is high, further development will be due. Especially in the case of usage with other browsers (e.g. Mozilla Firefox) and operating systems (e.g. Linux).

A trend in software testing is the automation. In order to be effective, large numbers of test cases must be generated, executed and the results evaluated. Every opportunity to automate parts of the process have to be carefully examined. [cf. W3St07]

With increasing automation, test runners are going to be more and more important, because larger numbers of test cases and test units have to be handled, therefore test runners may become important for developers. Without test runners it would get too circuitously to run all test cases and test units, which would lead to the converse effect – decreasing automation.

6 Bibliography

[Ashl07] Ashley, David: XMLPARSER.CLS.
http://sourceforge.net/project/showfiles.php?group_id=119701&package_id=204018&release_id=447749,
as of (2007-01-31)

[Cowl94] Cowlishaw, Mike: The early history of Rexx.
http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/an/&toc=comp/mags/an/1994/04/a4toc.xml&DOI=10.1109/85.329753,
1994-12-10, as of (2006-11-27)

[CySc02] Cyganiak, Richard; Schulze Daniel: JUnit - Ein Framework für Unit Tests.
http://richard.cyganiak.de/2002/junit/,
2002, as of (2007-01-18)

[Ehmk07] Ehmke, Dierk: Rechnergestützte Diagnose in Software-Entwicklung und -Test.
http://pi.informatik.uni-siegen.de/stt/25_1/01_Fachgruppenberichte/TAV/TAV22P2Ehmke.pdf,
2007-01-18, as of (2007-01-18)

[Flat05] Flatscher, Rony G.: Employing Rexx as a Scripting Language for Java.
http://www.informit.com/articles/article.asp?p=418864&rl=1,
2005-10-28, as of (2006-11-27)

[Flat06] Flatscher, Rony G.: ooRexxUnit: A JUnit Compliant Testing Framework for ooRexx Programs.
http://wi.wu-wien.ac.at/rgf/rexx/orx15/2004_orx15_orx-win.pdf,
2006-12-30, as of (2007-01-24)

[HaNe05] Hansen, Hans R; Neumann, Gustav: Wirtschaftsinformatik 1. Grundlagen und Anwendungen. 9. Aufl. Lucius & Lucius, Stuttgart 2005.

[Ragg05] Raggett, Dave: Getting started with HTML. http://www.w3.org/MarkUp/Guide/, 2005-05-24, as of (2006-11-27)

[W3Ab07] N.N.: What is abstraction? http://searchsmb.techtarget.com/sDefinition/0,,sid44_gci343038,00.html, 2007-01-17, as of (2007-01-17)

[W3Ac07] N.N.: What is ActiveX? http://searchwinit.techtarget.com/sDefinition/0,,sid1_gci211521,00.html, 2007-02-14, as of (2007-02-14)

[W3Co07] N.N.: Component Object Model. http://en.wikipedia.org/wiki/Component_object_model, 2007-01-17, as of (2007-01-17)

[W3Cs06a] N.N.: Cascading Style Sheets Homepage. http://www.w3.org/Style/CSS/, 2006-07-23, as of (2006-11-27)

[W3Cs06b] N.N.: Cascading Style Sheets. http://en.wikipedia.org/wiki/Cascading_Style_Sheets, 2006-10-05, as of (2006-11-27)

[W3Cp06a] N.N.: Common Public License – Version 1.0. http://www-128.ibm.com/developerworks/library/os-cpl.html, 2007-01-16, as of (2007-01-16)

[W3Cp06b] N.N.: Common Public License – Frequently asked questions. http://www-128.ibm.com/developerworks/library/os-cplfaq.html, 2007-01-16, as of (2007-01-16)

[W3Cp06c] N.N.: Common Public License.
http://en.wikipedia.org/wiki/Common_Public_License,
2007-01-16, as of (2007-01-16)

[W3Do07] N.N.: Document Object Model.
http://www.webopedia.com/TERM/D/DOM.html,
2007-01-17, as of (2007-01-17)

[W3Id07] N.N.: Integrated Development Environment.
http://en.wikipedia.org/wiki/Integrated_development_environment,
2007-01-17, as of (2007-01-17)

[W3Ie07] N.N.: Internet Explorer.
http://en.wikipedia.org/wiki/Internet_explorer,
2007-01-17, as of (2007-01-17)

[W3Ju07] N.N.: The JUnit framework.
http://web.cs.wpi.edu/~gpollice/TTG/Presentations/The%20JUnit%20Framework.pdf,
2007-01-18, as of (2007-01-18)

[W3En06] N.N.: What is encapsulation?
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212060,00.html,
2007-01-17, as of (2007-01-17)

[W3Gu07] N.N.: Graphical User Interface
http://en.wikipedia.org/wiki/Graphical_user_interface,
2007-01-17, as of (2007-01-17)

[W3Jc07] N.N.: JUnit: A Cook's Tour
http://junit.sourceforge.net/doc/cookstour/cookstour.htm,
2007-01-19, as of (2007-01-19)

[W3Op05] N.N.: Object-oriented programming.
http://searchwinit.techtarget.com/sDefinition/0,,sid1_gci212681,00.html,
2005-02-08, as of (2006-11-27)

[W3Op06a] N.N.: Object-oriented programming concepts.
http://java.sun.com/docs/books/tutorial/java/concepts/index.html,
2006-06-14, as of (2006-11-27)

[W3Op06b] N.N.: Object-oriented programming.
http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html,
2006-06-14, as of (2006-11-27)

[W3Op06c] N.N.: Object-oriented programming.
http://en.wikipedia.org/wiki/Object_oriented_programming,
2006-05-10, as of (2006-11-27)

[W3Pl06] N.N.: Programming language.
http://en.wikipedia.org/wiki/Programming_language,
2006-10-10, as of (2006-11-27)

[W3Pp06a] N.N.: Procedural programming language.
http://en.wikipedia.org/wiki/Procedural_programming_language,
2006-04-20, as of (2006-11-27)

[W3Pp06b] N.N.: Procedural programming.
http://javascript.about.com/od/reference/g/gprocedure.htm,
2006-05-10, as of (2006-11-27)

[W3Pp06c] N.N.: Procedural language.
http://www.answers.com/topic/procedural-language,
2006-05-10, as of (2006-11-27)

[W3Re02] N.N.: Rexx history.
http://www.cetus-links.org/oo_rexx.html
2002-05-20, as of (2006-11-27)

[W3Re06a] N.N.: Rexx history
http://www.scoug.com/OPENHOUSE/REXXINTRO/RexxHist1.2.html,
2006-04-20, as of (2006-11-27)

[W3Re06b] N.N.: Rexx history.
http://en.wikipedia.org/wiki/REXX,
2006-04-24, as of (2006-11-27)

[W3Re06c] N.N.: Rexx history.
http://hosthelp.de/REXX/History/history.html,
2006-04-20, as of (2006-11-27)

[W3Re06d] N.N.: Rexx language.
http://home.nvg.org/~sk/lang/lang.html,
2006-02-14, as of (2006-11-27)

[W3Re06e] N.N.:Rexx language.
http://listserv.uark.edu/scripts/wa.exe?A2=ind0103&L=ibmvm&T=0&P=10944,
2001-03-06, as of (2006-11-27)

[W3Se07a] N.N.: Software Engineering.
http://www.webopedia.com/TERM/S/software_engineering.html,
2007-01-18, as of (2007-01-18)

[W3Se07b] N.N.: Software Engineering.
http://www.answers.com/topic/software-engineering,
2007-01-18, as of (2007-01-18)

[W3Sg06] N.N.: Standard Generalized Markup Language.
http://etext.virginia.edu/bin/tei-tocs?div=DIV1&id=SG,
2007-01-16, as of (2007-01-16)

[W3Si07] N.N.: SilverMark's Enhanced JUnit.
http://www.silvermark.com/Product/java/enhancedjunit/documentation/enhancedjunitmanual.pdf,
2007-01-19, as of (2007-01-19)

[W3St07] N.N.: Software Testing.
http://www.mcs.le.ac.uk/people/glaycock/testing.html,
2007-01-31, as of (2007-01-31)

[W3Td06] N.N.: Top-Down and Bottom-Up design.
http://en.wikipedia.org/wiki/Top_down,
2007-01-16, as of (2007-01-16)

[W3Tr07] N.N.: What is Test Runner?
http://www.codeproject.com/debug/testrunner.asp,
2007-01-16, as of (2007-01-16)

[W3Uc06e] N.N.: Universal Character Set.
http://www.answers.com/topic/universal-character-set,
2007-01-16, as of (2007-01-16)

[W3Ut07] N.N.: Unit Testing.
http://en.wikipedia.org/wiki/Unit_test,
2007-01-18, as of (2007-01-18)

[W3Un07] N.N.: Unit Testing Tools.
http://www.testingfaqs.org/t-unit.html,
2007-01-18, as of (2007-01-18)

[W3We06] N.N.: Web and XML Glossary.
http://dret.net/glossary/xsl,
2007-01-17, as of (2007-01-17)

[W3Xm06a] N.N.: Extensible Markup Language (XML).
http://www.w3.org/XML/,
2006-10-06, as of (2006-11-27)

[W3Xm06b] N.N.: Introduction to XML.
http://www.w3schools.com/xml/xml_whatis.asp,
2006-10-06, as of (2006-11-27)

[W3Xm06c] N.N.: Extensible Markup Language.
http://en.wikipedia.org/wiki/Xml,
2006-10-06, as of (2006-11-27)

[W3Xs06a] N.N.: XSLT Tutorial.
http://www.w3schools.com/xsl,
2006-10-06, as of (2006-11-27)

[W3Xs06b] N.N.: The Extensible Stylesheet Language Family (XSL).
http://www.w3.org/Style/XSL/,
2006-10-06, as of (2006-11-27)

[W3Xs06c] N.N.: Extensible Stylesheet Language.
http://en.wikipedia.org/wiki/Xsl,
2006-10-06, as of (2006-11-27)

7 Appendix

The following chapters contain a collection of basics (*Excursus: Programming Language*, *Excursus: Rexx*, *Excursus: Common Public License*, *Excursus: HyperText Markup Language*, *Excursus: Cascading Style Sheets*, *Excursus: Extensible Markup Language* and *Excursus: Extensible Stylesheet Language*), which are helpful, when reading this Thesis and the source codes of the components of TeRA.

The different source code chapters are not sorted after the structure of the components in chapter 4, but programs with the same purpose are arranged together (e.g. Cascading Style Sheets or Rexx programs are in the same chapter).

7.1 Excursus: Programming Language

A programming language can be used to control the behaviour of a machine (in most cases a computer). It is an artificial language. Programming languages have syntactic and semantic rules which are used to define the meaning. It enables the organisation and manipulation of information and allows expressing algorithms precisely. In contrast to the manifold forms of human expression, programming languages require a greater degree of precision and completeness. Human authors and speakers can be ambiguous and make small errors and still expect their intent to be understood. Computers do exactly what they are told to do. They cannot understand the code the programmer "intended" to write. The language definition, the program, and the program's inputs must fully specify the external behaviour that occurs when the program is executed.

Sometimes the expression “computer language” is used for more limited artificial languages.

There is a wide range of programming languages. Only a few of them have become popular. [cp. W3Pl06]

7.1.1 Definitions

There are many positions on the precise definition of programming language. Some definitions are nearly equal and listed following. [cp. W3Pl06]

FUNCTION stands for the performance of some kind of computation and/or organization of the flow of control between mechanical devices. [cp. W3Pl06]

TARGET means the communication instructions to machines. Programming languages are used by one program or machine to program another, in some cases. For example, the postscript[1] source code is frequently generated programmatically to control a computer printer or display. [cp. W3Pl06]

The term **CONSTRUCT** stands for the definition and manipulation of data structures or for controlling the flow of execution. [cp. W3Pl06]

EXPRESSIVE POWER: Languages can be classified by the computations they can express. All Turing[2] complete languages can implement the same algorithms. ANSI/ISO SQL and Charity are examples of languages that are not Turing complete, yet are often called programming languages. [cp. W3Pl06]

There are also non-computational languages, such as markup languages like HTML or formal grammars like BNF. They are usually not considered programming languages; however, informal usage sometimes includes them. [cp. W3Pl06]

7.1.2 Trends and Purpose

There has been a trend in the development of programming languages to add more ability to solve problems using a higher level of abstraction. The first programming languages were very dependent on the underlying hardware of the computer. In the new programming languages a lot of features are included.

---

[1] Postscript: programming language, which provides operators and procedures e.g to forward. texts to printers.

[2] Alan Mathison Turing (1912 - 1954) – Turing test - is a proposal for a test of a machine's capability to perform human-like conversation.

Because of the fact that programmers are less tied to the needs of the computer, their programs are more efficiently with, at the same time, less effort from the programmer. This is the reason why programmes are able to write more programs in the same amount of time. [cp. W3Pl06]

To eliminate the need for a specialised language for programming, natural language processors have been developed. [cp. W3Pl06]

7.1.3 Object-Oriented Programming

Most of the popular programming languages support object-oriented programming, and a lot of popular programming frameworks, like Java and the .NET Framework, are built on object-oriented principles. [cp. W3Op05, W3Op06a, W3Op06b, W3Op06c]

The idea behind object-oriented programming is that a computer program may be seen as comprehending a collection of individual units or objects. These units and objects act on each other. This is contrary to the traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer. Each object is able to receive messages, process data, and send messages to other objects. Each object can be seen as an independent little part of the programming language, with a distinct role or responsibility. [cp. W3Op05, W3Op06a, W3Op06b, W3Op06c]

Object-oriented programming is suggested to provide greater flexibility and maintainability in programming. It is widely popular in large-scale software engineering. It is furthermore claimed to be easier to learn for those who are new to computer programming, and to be simpler to develop and to maintain, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than other programming methods. Critics deny this statement, at least for some domains. [cp. W3Op05, W3Op06a, W3Op06b, W3Op06c]

7.1.3.1 Fundamental Concepts

There are some concepts which are important to understand the functionality of object oriented programming:

- Class,

- Object,
- Message,
- Method,
- Inheritance,
- Abstraction and
- Encapsulation.

A **CLASS** is a description of objects. It serves as pattern for objects of the same type. The class assigns the object type. Objects can be created from a class via Instantiation. These created objects have the same (in the class predefined) characteristics. [cf. HaNe05, 215]

The definitions will be demonstrated by the means of the following example.

A new class `car` is created.

An **OBJECT** is an encapsulation of data (status) and functions (methods) in e entity. To change or query the status of an object, methods of the object are used exclusively. [cf. HaNe05, 214]

`Jaguar` and `Porsche` are two objects of the class `car`.

A **MESSAGE** is used to interact. This is done by sending these messages from one object to another. An object defines through its interface which methods can be processed. [cf. HaNe05, 215]

To assure that the objects interact, in this example a message would be send to the `Jaguar` with the appropriate message operator (i.e. Open Object Rexx: `Jaguar~drive()`).

A **METHOD** can be described as the ability of an object. [cp. W3Op05, W3Op06a, W3Op06b, W3Op06c]

The class `car` has the methods `drive()` and `brake()`.

The **INHERITANCE** relates to the ability to pass object characteristics from a pattern (normally along a class hierarchy). Via inheritance the reuse of characteristics can be achieved. At the inheriting class (or objects) the characteristics can be added not modified (*extension*) or can be replaced by a characteristics with the same name but different content (*overwrit*e) [cp. HaNe05, 217]

The object `Porsche` also has the method `drive()` and `brake()` from the `car` class, but can have other methods itself, which could be inherited to an object of `Porsche`, i.e. `911`, `Cayman` or `Cayenne`.

ABSTRACTION is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics. [cf. HaNe05]

Abstraction is one of three central principles of object oriented programming (abstraction, encapsulation and inheritance). “Through the process of abstraction, a programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency. In the same way that abstraction sometimes works in art, the object that remains is a representation of the original, with unwanted detail omitted.” [cf. W3Ab07]

ENCAPSULATION is the inclusion within a program object of all the resources need for the object to function - basically, the methods and the data. [cf. W3En06]

7.1.4 Procedural Programming

Procedural programming languages support the Top-Down[1] strategy by allowing to combine a consequence of commands in the form of a procedure or subroutine. Procedures can be called on any position within the program. Furthermore different procedures can be provided for other programs in the form of program libraries. When calling a procedure, the control flow goes to the begin-

[1] “Top-down programming is a programming style, the mainstay of traditional procedural languages, in which design begins by specifying complex pieces and then dividing them into successively smaller pieces.” [cf. W3Td06]

ning of this particular procedure and after executing it, the control flow goes back to the next procedure.

7.2 Excursus: Rexx

Rexx is the acronym of **Re**structured E**x**tended E**x**ecutor.

7.2.1 The History of Rexx

Rexx was developed by IBM (Mike Cowlishaw) in 1979. Originally Rexx was intended as a scripting programming language for main frame systems to replace the batch languages EXEC and EXEC 2. It is easy to learn and easy to read compared to other programming languages. It was introduced at a conference in Houston and became an IBM product in 1982. [cf. Cowl94, W3Re02, W3Re06a, W3Re06b, W3Re06c]

Rexx was included in nearly all operating systems within IBM. Shortly after that IBM made versions available for Linux and Windows. There were other versions for Amiga, many variants of UNIX, Atari, Solaris, Palm OS, Macintosh and Mac OS too. [cf. Cowl94, W3Re02, W3Re06a, W3Re06b, W3Re06c]

Two newer variants of Rexx appeared from IBM in the mid 1990s:

- NetRexx and
- Object Rexx.

Object Rexx is an upward-compatible, object-oriented version of the original Rexx. In 2004 IBM turned over their Object Rexx implementation to the Rexx Language Association (RexxLA) which assigned it to open source software and released Open Object Rexx in 2005 under the Common Public License (CPL). [cp. Cowl94, W3Re02, W3Re06a, W3Re06b, W3Re06c]

7.2.2 Characteristics and Features

REXX has the following characteristics and features:

- Dynamic data typing (no declarations),
- No reserved keywords (except in local context),
- Arbitrary numerical precision,

- Decimal arithmetic (floating-point),
- A rich selection of built-in functions,
- Content-addressable data structures,
- Straightforward access to system commands and facilities,
- Simple error-handling, and built-in tracing and debugger,
- Few limitations and
- Simplified I/O facilities. [cf. W3Re06b]

A Rexx standard X3.274-1996 was issued in the year 1996 by the American National Standards Institute (ANSI) for the programming language. [cf. Cowl94, W3Re02, W3Re06a, W3Re06b, W3Re06c]

“REXX is a procedural programming language that allows programs and algorithms to be written in a clear and structured way. It is easy to use by experts and casual users alike. REXX has been designed to make easy the manipulation of the kinds of symbolic objects that people normally deal with such as words and numbers. Although REXX has the capability to issue commands to its host environment and to call programs and functions written in other languages, it is also designed to be independent of its supporting system software when such commands are kept to a minimum.” [cf. W3Re06d]

“IBM Object REXX is an object-oriented programming language suited both for beginners and experienced OO programmers. It is upward-compatible with previous versions of "classic" REXX and provides an easy migration path to the world of objects. Because it can be used with REXX conventional programming, Object REXX protects your investment in existing REXX program code. It provides many programming interfaces to existing applications, such as DB2, C and C++ applications.“ [cf. W3Re06e]

7.3 Excursus: Common Public License – CPL

The **C**ommon **P**ublic **L**icense (CPL) is a free software or open-source software license published by IBM. The latest version is the CPL v1.0, which is the successor of the CPL v0.5, which in turn is the successor of the IPL (IBM Public License. The license terms of the CPL v1.0 have been approved by the Open Source Initiative (posted on the OSI site in June 2002) and Free Software Foundation. [cf. W3Cp06a, W3Cp06b, W3Cp06c]

The CPL was written to generalize the usage terms of the IPL so that any Open Source originator can use the terms found in the IPL. The CPL is suitable to be used by everybody. The CPL's aims are to support and encourage collaborative open source development. The ability to use the CPL content with software licensed under other licenses, including many proprietary licenses is still retaining. [cf. W3Cp06a, W3Cp06b, W3Cp06c]

There are some terms in the CPL that are similar to the GNU General Public License. But there are some key differences as well. [cf. W3Cp06a, W3Cp06b, W3Cp06c]

One key difference is relicensing: a program licensed under the CPL may be compiled without modification and the result in turn commercially licensed in accordance with the terms of the CPL. [cf. W3Cp06a, W3Cp06b, W3Cp06c]

Another key difference is in a patent clause: it is designed to prevent unscrupulous contributors from contributing code which infringes on their patents, and then attempting to charge royalties; in such a situation, the CPL requires the contributor to grant a royalty-free license to all recipients. A similarity between the CPL and the GPL (General Public License) is related to a distribution of a modified program: under either license (CPL or GPL), one is obligated to make the source code to the modified Program available to others. [cf. W3Cp06a, W3Cp06b, W3Cp06c]

7.4 Excursus: HyperText Markup Language – HTML

HTML stands for **H**yper**T**ext **M**arkup **L**anguage and is a special kind of text document. It is used to present text and graphics by web browsers and can be used to describe, to some extent, the appearance and semantics of a document. Its grammar structure is the HTML DTD which was created using SGML syntax[1] (**S**tandard **G**eneralized **M**arkup **L**anguage). Originally HTML was defined by Tim Berners-Lee and further developed by the IETF. HTML is now an international standard (ISO/IEC 15445:2000) and its specifications are maintained by the World Wide Web Consortium (W3C). [cp. Ragg05]

HTML uses markup start-tags to indicate the start of an element, and markup end-tags to indicate the end of an element. The markup start-tag has the form "`< tagname >`"and the end-tag the form "`</tagname>`" (i.e. tags for a first-level heading: "`<h1> headline text </h1>`"). Very often HTML documents are also referred to as "Web pages". The browsers retrieve web pages from web servers over the Internet. [cp. Ragg05]

There are a lot of editors for creating HTML (NotePad, TextEdit, Dreamweaver, etc.). [cp. Ragg05]

HTML documents are accessible on a wide range of browsers such as for example Mozilla Firefox, Opera, Internet Explorer and Safari. [cp. Ragg05]

7.4.1 HTML Markup

HTML markup consists of several types of entities. This includes elements, attributes, data types and character references. [cp. Ragg05]

[1] SGML: "is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form." [cf. W3Sg06]

7.4.1.1 Elements

The basic structure for HTML markup is the element. Each element has two basic properties: attributes and content. To generate valid HTML each attribute and each content has certain restrictions that must be followed. [cp. Ragg05]

Listed below there are several types of markup elements used in HTML:

- Structural markup
  "`<h2>Volkswagen</h2>`" establishes "Volkswagen" as a second-level heading.

Structural markup does not denote any specific rendering. Most of the web browsers have standardized on how elements should be formatted. But in general, if styling of HTML is desired, this should be done by using Cascading Style Sheets (CSS).

- Presentational markup
  "`<b>Passat</b>`" indicates that the content should be rendered in bold text.

Presentational markup describes the appearance of the text independent of its function. It is largely deprecated. A better form to control presentation is the assignment of CSS. [cp. Ragg05]

It is also possible to link parts of the document to other documents. This is done by using anchor elements as hyperlinks. [cp. Ragg05]

The element "`<a href="http://www.w3.org>w3 homepage</a>`" links to the W3 homepage, presumed that the URL[1] is valid. [cp. Ragg05]

7.4.1.2 Attributes

Attributes of elements are name-value pairs, separated by `=`. They are written after the element's name within the start tag of an element. The attribute's value should be enclosed in single or double quotes otherwise it is considered as unsafe. [cp. Ragg05]

There are a lot of attributes, such as `id`, `class` and `style`.

The most important attribute is the `id` attribute. It provides a document-wide unique identifier for an element. It can be used by style sheets to provide presentational properties, by browsers to locate specific elements or by scripts to alter the contents or presentation of a specific element. [cp. Ragg05]

By using the `class` attribute elements can i.e. be classified for presentation purposes. These classes of elements might be gathered together and presented in a special way. This would be done by using `.classname` and creating some styling rules within the CSS. [cp. Ragg05]

The `style` attribute is used to add presentational properties to a particular element. To demonstrate these various attributes the span element can be used. [cp. Ragg05]

```
<span id="check1" class="myclass" style="font-size:30px;
color:red;">TeRA-A TestRunner Application 4 ooRexxUnit</span>
```

Code 5: Span Element

Figure 56 shows the result of the `span` element within an HTML file.

Figure 56: Span Element.

[1] **U**niform **R**esource **L**ocator (URL): means of locating the resource by describing its primary access mechanism

7.5 Excursus: Cascading Style Sheets – CSS

Cascading Style Sheets (CSS) are used to store the presentation rules of a document. The most important application is to style web pages written in HTML and XHTML. It can also be applied to any kind of XML document. The specifications for Cascading Style Sheets are maintained by the World Wide Web Consortium. [cp. W3Cs06a, W3Cs06b]

There are various levels and profiles of CSS. Each new level of CSS builds upon the previous one. The newer level typically adds new features. These levels are denoted as CSS1, CSS2, and CSS3. [cp. W3Cs06a, W3Cs06b]

Profiles are typically a subset of one or more levels of CSS. They are built for a particular device or user interface. [cp. W3Cs06a, W3Cs06b]

Style sheets are used to define nearly all aspects of document presentation. It separates the content of a document (HTML or a similar markup language) from document presentation. Separation of content and presentation can improve content accessibility, provide more flexibility and control in the specification of presentational characteristics. It can reduce complexity and repetition in the structural content, too. Identical HTML or XML markup can be displayed in a variety of styles by using different CSS. [cp. W3Cs06a, W3Cs06b]

CSS can be provided by various sources:

- External style sheets (a separate CSS-file referenced to from the document),
- Embedded style (blocks of CSS information inside the HTML document),
- Inline styles (inside the HTML document, style information on a single element, specified using the "style" attribute),
- A local CSS-file specified by the user using options in the web browser (acting as an override, to be applied to all documents) and
- The default style sheet applied by the user agent (browser).

To determine which style rules apply, if more than one rule matches against a particular element, CSS specifies a priority scheme. This priority scheme is called 'cascade'. Priorities (also called 'weights') are calculated and then assigned to rules. [cp. W3Cs06a, W3Cs06b]

Advantages of using CSS include:

- Presentation information can be held in one place. Updates can be done easily.
- Different CSS for different users.
- The document code is reduced in size and complexity (no implicitly containing presentational markup).

Each style sheet consists of a list of rules, which itself consists of one or more comma-separated selectors and a declaration block. A declaration-block consists of a list of semicolon-separated declarations in curly braces. Each declaration itself consists of a property, a colon (:) then a value. [cp. W3Cs06a, W3Cs06b]

The following example shows a rule for a level 1 headline:

```
h1                          rule
{                           start of declaration block
    font-size:24px;         declaration: value
}                           end of declaration block
```

[cf. W3Cs06a, W3Cs06b]

7.6 Excursus: Extensible Markup Language – XML

The E**x**tensible **M**arkup **L**anguage (XML) is a language for defining markup languages. Recommended by the W3C, it is capable of describing many different kinds of data and contains data, as well. XML is a simplified subset of Standard Generalized Markup Language (SGML) with the primary purpose to facilitate the sharing of data across different systems, especially via the Internet connected systems. XML based languages are defined in a formal way. This allows programs to modify and validate documents in these languages, also without prior knowledge of their particular form. [cp. W3Xm06a, W3Xm06b, W3Xm06c]

XML describes and applies a tree-based structure to information, by providing a text-based means. All information is manifested as text, which is equipped with markup that indicates the information's separation into a hierarchy of character data, container-like elements, and attributes of those elements. [cp. W3Xm06a, W3Xm06b, W3Xm06c]

The basic unit in XML is the character. It is defined by the **U**niversal **C**haracter **S**et (UCS)[1]. Characters are combined to form an XML document, which itself consists of one or more entities, whereas in turn each is part of the document's characters, stored in a text file. [cp. W3Xm06a, W3Xm06b, W3Xm06c]

[1] Universal Character Set: “The international standard ISO/IEC 10646 defines the Universal Character Set (UCS) as a character encoding. It contains nearly a hundred thousand abstract characters, each identified by an unambiguous name and an integer number called its code point.” [cf. W3Uc06e]

7.7 Excursus: Extensible Stylesheet Language – XSL

XSL is a family of languages which describes how files (encoded in the XML standard) are to be formatted or transformed. This family is divided into tree different languages:

XML languages:

- **XSL T**ransformations (XSLT) is the processing language for XSL. It is used to convert XML documents into HTML or other document types.
- **XSL F**ormatting **O**bjects (XSL-FO): specifying the visual formatting of an XML document (provides the format vocabulary).

Non-XML language:

- the **X**ML **Path** Language (XPath): used by XSLT, and also available for use in non-XSLT contexts, for addressing (identify and select tagged elements within an XML document) the parts of an XML document.

The above mentioned specifications are available in the form of W3C recommendations. [cp. W3Xs06a, W3Xs06b, W3Xs06c]

XSL can be used for displaying XML documents. XSL can be divided into two steps. The first is a transformation of the XML document using XSLT. The second bothers with the rendering of the result of the transformation, which is done by using XSL-FO. XSL covers the same application area as CSS. Furthermore it is much more powerful, because the transformation step can perform arbitrarily complex transformations of the XML document. CSS is not able to make any structural changes to the XML document. [cf. W3We06]

7.8 Source Code Listing

The following chapters are a collection of the source code of the different components of TeRA. They are not sorted after the structure of the components in chapter 4, but programs with the same purpose are arranged together (i.e. Cascading Style Sheets or Rexx programs are in the same chapter).

7.8.1 Rexx Programs

Within this chapter, there are all Rexx programs of TeRA.

7.8.1.1 TeRA.REX

TeRA.REX is the starting program of the TeRA test runner application.

```
/*
   name:             TeRA.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - provides navigation and selecting

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/
```

```
say "opening TeRA.hta........."

dir=directory()      --get the actual directory
bs=countstr("\",dir)      --count backslash
sl=countstr("/",dir)      --count slash

   if bs>=1 then op="\"      --if bs>=1 then operating system = Windows
   else op="/"      --if sl>=1 then operating system = Linux

path=dir||op||"TeRA.hta"      --create path

-------------------------------------------

TeRAfilespath=dir||op||"TeRA_files"

lastlogfile=TeRAfilespath||op||"lastlog.xml"      --get path for lastlogfile

-------------------------------------------
call sysmkdir dir||op||"logfiles"      /* create new folder ("logfiles" for log files)
*/
call sysmkdir dir||op||"TeRA_files"    /* create new folder ("TeRA_files" for additional
Info) */
call sysmkdir dir||op||"compared_files"      /* create new folder ("compared_files" for
compared log files) */

/* copy style sheets to folders */
address cmd 'copy' dir||op||"log.css" dir||op||"logfiles"
address cmd 'copy' dir||op||"log.css" dir||op||"TeRA_files"
address cmd 'copy' dir||op||"complog.css" dir||op||"compared_files"
address cmd 'copy' dir||op||"log_xsl_script.rex" dir||op||"logfiles"
address cmd 'copy' dir||op||"complog_xsl_script.rex" dir||op||"compared_files"

qte='"'      --variable for "
qtb="'"      --variable for '
rc=stream(path,"C","close")      -- to close the file TeRA.hta
call SysFiledelete path      --delete the file

--start writing .hta
call lineout path,"<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'>"
call lineout path,"<html>"
call lineout path,"<head>"

call lineout path,"<title>TeRA</title>"

call lineout path,"<link rel='stylesheet' href='styles.css' type='text/css'>"

-------------------------------------------------------------------------------------
--start script
-------------------------------------------------------------------------------------

call lineout path,"<script language='Object Rexx' src='TeRA_script.rex'></script>"

-------------------------------------------------------------------------------------
--end script
-------------------------------------------------------------------------------------

call lineout path,"</head>"
call lineout path,"<body onload='call TUcode'>"
call lineout path,"<div id='content_container'>"
call lineout path,"<img src='Tera_Logo4TeRA_rex.jpg' id='logo' alt='oorexx logo'-
                  height='20%'>"

-------------------------------------------------------------------------------------
--start generate tables for "checkbox-tree"
-------------------------------------------------------------------------------------
-- search files only, in all subdirs, return fully qualified path
flags      ="FSO"

/* to minimize loading time insert path to location of test units on your hard disk */
   if op="\" then
   call SysFileTree "C:\oorexxunit\*.testunit", "files.", flags

/* to minimize loading time insert path to location of test units on your hard disk */
   if op="/" then
   call SysFileTree "/*.testunit", "files.", flags

   if files.0=0 then
   do
   call rxmessagebox "no files found - aborting"
```

```
   exit -1
   end

l=.list~new       -- create a list
do i=1 to files.0      -- add each found file into the list
l~insert(files.i)
end

y=l~makearray      --make array from list
buttonid1=3      --couter for buttonID
buttonid2=4
index=y~items      --get number of arrayitems

call lineout path,"<form id='form1'>"
call lineout path,"<DL id='TUL1'>"
call lineout path,"<DT><input type='button' id='"buttonid1"' value='+' -
                  onclick="qte"document~getElementbyID('TB0')~style~display = -
                 "qtb"inline"qtb""qte"><input type='button' id='"buttonid2"' value='-' -
                  onclick="qte"document~getElementbyID('TB0')~style~display = -
                  "qtb"none"qtb""qte">  all testunits </DT>"
buttonid1=buttonid1+2
buttonid2=buttonid2+2
call lineout path,"<DD>"
call lineout path,"<Table id="TB0">"
call lineout path,"<TR>"
call lineout path,"<TD>"
call lineout path,"<input type='checkbox' alt="TB0" name='TU0' id='all0'-
                  value='level0' onclick='call checkmore'>all TestUnits</br>"
call lineout path,"</TD>"
call lineout path,"</TR>"

z=2 --counter: <DL> ID
setindex=0 --counter: value of textarea "output"
findex=1 --counter: attribute "alt" testunit-checkbox; attribute "ID" <table>
aindex=1 --counter: attribute "ID" "all"-checkbox
nindex=1 --counter: attribute "name" testunit-checkbox
no=0 --counter: attribute "ID" testunit-checkbox
name="TestUnit"      --Variable for html-checkboxnames
tablearray=.array~of()
table_index=1

   do i=1 to index      --do over all testunits
   g=y~at(i)      /* the following block cuts each testunitpath (e.g.:
result="oorexx.base.class.Supplier.testUnit") */
   lpos1=lastpos(op,g)
   lpos2=lastpos(".",g)
   full=substr(g,lpos1+1,(lpos2-1)-lpos1)
   dots=countstr(".",full)      --count dots for level
   posindex=1
   interarray=.array~of()
   x=2

      do c=1 to dots     /* extract each level name - level1 is predefined as "all
testunits" - (e.g. level2:"oorexx", level3:"base", level4:"class", level5:"Supplier") */
      fdotpos=pos(".",full,posindex)
      level=substr(full,posindex,fdotpos-posindex)
      interarray~put(g,1)
      interarray~put(level,x)      --insert test unit names into array
      x=x+1
      posindex=fdotpos+1
         if c=dots then
            do
            lastlevel=substr(full,posindex)
            interarray~put(lastlevel,x)
            x=x+1
            end
      end

   tablearray~put(interarray,table_index) --insert interarray into tablearray
   table_index=table_index+1
   end

p=tablearray~items
   do i=1 to p --run through tablearray for all test units and the right level
   a_array=tablearray~at(i)
   a_items=a_array~items

   do t=2 to a_items
   act_aitem=a_array~at(t)
```

```
   testname=a_array~at(1)

   posinfo=lastpos(op,testname)
   infotestname=substr(testname,posinfo+1) /* get testname for then anchor in the links
*/

/* if i=1 then level1, if t\=a_items then the actual item is not the last in the array
*/
      if (i=1 & t\=a_items) then      --start creating the test unit tree
      do
      call lineout path,"</Table>"
      call lineout path,"<DL id='"TUL||z"'>"
      call lineout path,"<DT><input type='button' id='"buttonid1"'-
                        value='+' onclick="qte"document~getElementbyID-
                       ("qtb"TB"||findex""qtb")~style~display = "qtb"inline"qtb""qte">-
                       <input type='button' id='"buttonid2"' value='-' -
                       onclick="qte"document~getElementbyID-
                       ("qtb"TB"||findex""qtb")~style~display = "qtb"none"qtb""qte">-
                        "act_aitem" <input type='button' class='but1' value='checkAll'-
                       alt="qtb"TB"||findex""qtb" onmouseover="qte"-
                       document~getelementbyid("qtb"output5"qtb")~value=this~alt"qte" -
                       onclick='call folder_checkAll'></input><input type='button' -
                       class='but2' value='uncheckAll' alt="qtb"TB"||findex""qtb" -
                       onmouseover="qte"document~getelementbyid-
                      ("qtb"output5"qtb")~value=-
                      this~alt"qte" onclick='call folder_uncheckAll'></input></DT>"
      buttonid1=buttonid1+2
      buttonid2=buttonid2+2
      call lineout path,"<DD>"
      call lineout path,"<Table id='TB"||findex"'>"
      z=z+1
      findex=findex+1

      call lineout path,"<TR>"
      call lineout path,"<TD>"
      call lineout path,"<input type='checkbox' alt='TB"||findex-1"' -
                        name='TU"||nindex"' id='all"||aindex"' value='level"t-1"' -
                        onclick='call checkmore'>all "act_aitem" Tests</br>"
      call lineout path,"</TD>"
      call lineout path,"</TR>"

      setindex=setindex+1
      nindex=nindex+1
      aindex=aindex+1
      end

/* actual item ist the last in the array - insert the following item into new folder */
         if (i=1 & t=a_items) then
         do
         call lineout path,"<TR>"
         call lineout path,"<TD>"
         call lineout path,"<input type='checkbox' alt='TB"||findex-1"' -
                           name='TU"||nindex"' id='"name||no"' value='"testname"'>-
                           <a href='"testname"' class='link_tu' -
                           alt='"a_array~at(a_items)"'-
                           id='"a_array~at(a_items)"' value='"testname"'>-
                          "a_array~at(a_items)"</a>   <a -
                          href='"lastlogfile"#"infotestname"'-
                          target='_blank' name='namelink"||no"' -
                          id='link"||nindex"'>info-
                          </a></br>"
         call lineout path,"</TD>"
         call lineout path,"</TR>"
         no=no+1
         setindex=setindex+1
         nindex=nindex+1
         end

/* not level1, other items are following */
         if (i\=1 & i\=p) then
         do
         b_array=tablearray~at(i-1)
         b_items=b_array~items
         act_bitem=b_array~at(t)
            if b_items>a_items then
            do
            q=b_items-a_items
            do r=1 to q+1
            call lineout path,"</Table>"
```

```
            call lineout path,"</DD>"
            call lineout path,"</DT>"
            call lineout path,"</DL>"
            end
            end

        /* create new folder, not the last item */
            if (act_aitem\=act_bitem & t\=a_items) then
            do
            call lineout path,"</Table>"
            call lineout path,"<DL id='"TUL||z"'>"
            call lineout path,"<DT><input type='button' id='"buttonid1"' -
                              value='+' onclick="qte"document~getElementbyID-
                             ("qtb"TB"||findex""qtb")~style~display = -
                             "qtb"inline"qtb""qte">-
                             <input type='button' id='"buttonid2"' value='-' onclick=-
                             "qte"document~getElementbyID-
                             ("qtb"TB"||findex""qtb")~style~display = -
                             "qtb"none"qtb""qte">  -
                             "act_aitem" <input type='button' class='but1' -
                             value='checkAll' -
                             alt="qtb"TB"||findex""qtb" onmouseover="qte"document~-
                             getelementbyid("qtb"output5"qtb")~value=this~alt"qte" -
                             onclick='call folder_checkAll'></input><input -
                             type='button' -
                             class='but2' value='uncheckAll' alt="qtb"TB"||-
                             findex""qtb" -
                             onmouseover="qte"document~getelementbyid-
                             ("qtb"output5"qtb")~value-
                             =this~alt"qte" onclick='call -
                             folder_uncheckAll'></input></DT>"
            buttonid1=buttonid1+2
            buttonid2=buttonid2+2
            call lineout path,"<DD>"
            call lineout path,"<Table id='TB"||findex"'>"
            z=z+1
            findex=findex+1

            call lineout path,"<TR>"
            call lineout path,"<TD>"
            call lineout path,"<input type='checkbox' alt='TB"||findex-1"' -
                              name='TU"||nindex"' id='all"||aindex"' value='level"t-1"'-
                              onclick='call checkmore'>all "act_aitem" Tests</br>"
            call lineout path,"</TD>"
            call lineout path,"</TR>"

            setindex=setindex+1
            nindex=nindex+1
            aindex=aindex+1
            end

/* last item in this folder */
            if (act_aitem\=act_bitem & t=a_items) then
            do
            call lineout path,"<TR>"
            call lineout path,"<TD>"
            call lineout path,"<input type='checkbox' alt='TB"||findex-1"' -
                              name='TU"||nindex"' id='"name||no"' value='"testname"'>-
                             <a href='"testname"' class='link_tu' -
                             alt='"a_array~at(a_items)"' -
                             id='"a_array~at(a_items)"' -
                             value='"testname"'>"a_array~at(a_items)"-
                             </a>   <a href='"lastlogfile"#"infotestname"' -
                             target='_blank' -
            name='namelink"||no"' id='link"||nindex"'>info</a></br>"
            call lineout path,"</TD>"
            call lineout path,"</TR>"
            no=no+1
            setindex=setindex+1
            nindex=nindex+1
            end
            end

/* block for closing all opened tags to this level */
          if i=p then
          do
          b_array=tablearray~at(i-1)
          b_items=b_array~items
```

```
         act_bitem=b_array~at(t)
            if t\=a_items then
            do
               if b_items>a_items then
               do
               q=b_items-a_items
               do r=1 to q+1
               call lineout path,"</Table>"
               call lineout path,"</DD>"
               call lineout path,"</DT>"
               call lineout path,"</DL>"
               end
               end
            end

/* new folder, not the last item */
            if (act_aitem\=act_bitem & t\=a_items) then
            do
            call lineout path,"</Table>"
            call lineout path,"<DL id='"TUL||z"'>"
            call lineout path,"<DT><input type='button' id='"buttonid1"' -
                              value='+' onclick="qte"document~getElementbyID-
                             ("qtb"TB"||findex""qtb")~style~display = -
                             "qtb"inline"qtb""qte">-
                             <input type='button' id='"buttonid2"' value='-' -
                             onclick="qte"document~getElementbyID-
                             ("qtb"TB"||findex""qtb")~style~display = -
                             "qtb"none"qtb""qte">  -
                             "act_aitem" <input type='button' class='but1' -
                             value='checkAll' -
                             alt="qtb"TB"||findex""qtb" -
                             onmouseover="qte"document~getelementbyid-
                              ("qtb"output5"qtb")~value=this~alt"qte" onclick='call -
                             folder_checkAll'>-
                             </input><input type='button' class='but2' -
                             value='uncheckAll' -
                             alt="qtb"TB"||findex""qtb" -
                             onmouseover="qte"document~getelementbyid-
                             ("qtb"output5"qtb")~value=this~alt"qte" onclick='call -
                             folder_uncheckAll'>-
                             </input></DT>"
            buttonid1=buttonid1+2
            buttonid2=buttonid2+2
            call lineout path,"<DD>"
            call lineout path,"<Table id='TB'||findex"'>"
            z=z+1
            findex=findex+1

            call lineout path,"<TR>"
            call lineout path,"<TD>"
            call lineout path,"<input type='checkbox' alt='TB'||findex-1"' -
                              name='TU'||nindex"' id='all'||aindex"' value='level"t-1"'-
                              onclick='call checkmore'>all "act_aitem" Tests</br>"
            call lineout path,"</TD>"
            call lineout path,"</TR>"

            setindex=setindex+1
            nindex=nindex+1
            aindex=aindex+1
            end

/* new folder, last item */
            if (act_aitem\=act_bitem & t=a_items) then
            do
            call lineout path,"<TR>"
            call lineout path,"<TD>"
            call lineout path,"<input type='checkbox' alt='TB'||findex-1"' -
                              name='TU'||nindex"' id='"name||no"' value='"testname"'>-
                              <a href='"testname"' class='link_tu' -
                              alt='"a_array~at(a_items)"' -
                              id='"a_array~at(a_items)"' -
                              value='"testname"'>"a_array~at(a_items)"-
                              </a>   <a href='"lastlogfile"#"infotestname"' -
                              target='_blank' -
                              name='namelink"||no"' id='link"||nindex"'>info</a></br>"
            call lineout path,"</TD>"
            call lineout path,"</TR>"
            no=no+1
```

```
            setindex=setindex+1
            nindex=nindex+1
            do f=1 to a_items
            call lineout path,"</Table>"
            call lineout path,"</DD>"
            call lineout path,"</DT>"
            call lineout path,"</DL>"
            end
         end
      end
   end
end


call lineout path,"</Table>"
call lineout path,"</DD>" --close html
call lineout path,"</DT>"
call lineout path,"</DL>"
call lineout path,"</form>"

-------------------------------------------------------------------------------
--end generate tables for "checkbox-tree"
-------------------------------------------------------------------------------

call lineout path,"</div>"

-------------------------------------------------------------------------------
--start input buttons
-------------------------------------------------------------------------------
call lineout path,"<div id='footer1'>"      --area for buttons

call lineout path,"<input type='button' name='checkAll' id='checkAll' -
                  value='checkAll' alt='checkAll' onClick='call checkAll'>"

call lineout path,"<input type='button' name='uncheckAll' -
                  id='uncheckAll' value='uncheckAll' alt='uncheckAll' -
                  onClick='call uncheckall'>"

call lineout path,"<input type='button' name='openAll' id='openAll' -
                  value='open all' alt='openAll' onClick='call openAll'>"

call lineout path,"<input type='button' name='closeAll' id='closeAll' -
                  value='close all' alt='closeAll' onClick='call closeAll'>"

call lineout path,"<input type='button' name='invert' id='invert' -
                  value='invert' alt='invert' onClick='call invert'>"

call lineout path,"<span>username: <input type='text' id='username' -
                  name='username' alt='username' value='unknown'></input></span>"

call lineout path,"</div>"

call lineout path,"<div id='footer2'>"

call lineout path,"<input type='button' name='runTests' id='runTests' -
                  value='run tests' alt='runTests' onClick='call SelectedTestUnits'>"

call lineout path,"<input type='button' name='showResult' -
                  id='showResult' value='show result' alt='showResult' -
                  onClick='call openLastLog'>"

call lineout path,"<input type='button' name='compTests' -
                  id='compTests' value='compare tests' alt='compTests' -
                  onClick='call compareLogFiles'>"

call lineout path,"<span>comment: <input type='text' id='comment' -
                  name='comment' alt='comment' value='no comment'></input></span>"

call lineout path,"</div>"
-------------------------------------------------------------------------------
--end input buttons
-------------------------------------------------------------------------------

call lineout path,"<p type='textarea' size='30' id='output' -
                  value='"setindex"'></p>"
call lineout path,"<p type='textarea' size='30' id='output2' -
                  name='output2'></p>"
call lineout path,"<p type='textarea' size='30' id='output3' -
                  value='"findex-1"'></p>"
call lineout path,"<p type='textarea' size='30' id='output4' -
```

```
                  value='"no-1"'></p>"
call lineout path,"<p type='textarea' size='30' id='output5' -
                  value='0'></p>"
call lineout path,"</body>"
call lineout path,"</html>"

rc=stream(path,"C","close")      --cut programm's access to created file

address cmd path      --open file
```

7.8.1.2 TeRA_SCRIPT.REX

Within TeRA_SCRIPT.REX, there are all routines for TeRA.HTA.

```
/*
   name:             TeRA_script.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - provides routines for TeRA.hta

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/


::routine SelectedTestUnits public
  a=username~value      --get value of username
  nospace=a~translate('__________','"/?* :\<>|+()[]{}&%$§!~;#') /* translate special
characters */
  a=nospace
  b=comment~value      --get value of comment
  home=syssearchpath('path','xmlparser.cls')      --get the right path
  win=countstr("\",home)
  lin=countstr("/",home)
  if win>=1 then op="\"
  if lin>=1 then op="/"
  posname=pos('xmlparser.cls',home)      --edit path
```

```
  file=substr(home,1,posname-2)
  TeRAfiles=file||op||'TeRA_files'
  actTests=TeRAfiles||op||'actTests.txt'
  userinformation=TeRAfiles||op||'userinformation.txt'
  call lineout userinformation,a      --write username to file
  call lineout userinformation,b      --write comment to file
  rc=stream(userinformation,'C','close')
  set=(output~value)
  LogFileList=.list~of()
  form = document~forms~item(0)
  name='TU'
    do i=0 to set
    check = form~item(name||i)~checked      --item checked or not
      if check=.true then
      do
      selection=form~item(name||i)~value      --if checked then get value
      sel=compare(selection,'level')
        if sel=1 then
        do
        LogFileList~insert(selection)
        gg=LogfileList~items
          if gg\=0 then
          call lineout actTests,selection      /* write test values to file (info for
TeRA-runtestsandedit.rex) */
        end
      end
    end
  noi=LogfileList~items      /* the following code: if no test is checked - alert! */
    if noi=0
    then call rxmessagebox 'to run a test, you have to choose at least one testunit'
  rc=stream(actTests,'C','close')
  ab=LogfileList~items
    if ab\=0 then
    do
    editadress=syssearchpath('path','TeRA-runTestsandEdit.rex')
    address cmd 'rexx 'editadress
    location~reload()
    end

::routine openLastLog public
  address cmd 'rexx TeRA-openLastLog.rex'

::routine compareLogFiles public
  address cmd 'rexx TeRA-compareLogs.rex'

::routine invert public
  set=(output~value)
  form = document~forms~item(0)
  name='TU'
    do i=0 to set
    check = form~item(name||i)~checked      /* if item is checked then uncheck it
onclick */
      if check=1 then
      do
      form~item(name||i)~checked=0
      end
      if check=0 then
      do
      getid=form~item(name||i)~id
      idcomp=compare('all',getid)
        if idcomp=1 then
        form~item(name||i)~checked=true
        if idcomp=4 then
        form~item(name||i)~checked=0
      end
    end

::routine uncheckAll public
  set=(output~value)
  form = document~forms~item(0)
  name='TU'
    do i=0 to set
    form~item(name||i)~checked=0
    end

::routine checkAll public
  set=(output~value)
  form = document~forms~item(0)
```

```
  name='TU'
    do i=0 to set
    form~item(name||i)~checked=1
    end

::routine openAll public      --open all folders
  xset=output3~value
    do h=0 to xset
    des=TB||h
    document~getElementbyID(des)~style~display='inline'
    end

::routine closeAll public      --close all folders
  xset=output3~value
    do h=0 to xset
    des=TB||h
    document~getElementbyID(des)~style~display='none'
    end

::routine checkmore public
  set=(output~value)
  form = document~forms~item(0)
  name='TU'
    do i=0 to set
    idstat = form~item(name||i)~id
    r=compare('all',idstat)
      if r=4 then
      do
      check=form~item(name||i)~checked
        if check=1 then
        do
        blevel=form~item(name||i)~value
        ilevel=substr(blevel,2)
          do x=i+1 to set
          bval=form~item(name||x)~value
          home=syssearchpath('path','xmlparser.cls')
          bs=countstr('\',home)
          sl=countstr('/',home)
            if bs>=1 then op='\'      --operating system Windows
            if sl>=1 then op='/'      --operating system Linux
          ival=substr(bval,2)
            if op='\' then
            do
            g=compare(':\',ival)       /* cut driveletter to be independent of drive
(Windows) */
              if g>=2 then      /* if result is unequal 2 the value is the path
declaration */
              form~item(name||x)~checked=1
              if g<2 then      /* if result is < 2 the value is the level declaration */
              do
              comp2=delstr(ival,1,4)        /* cut the name "level" and leave the level
number */
              comp1=delstr(ilevel,1,4)
                if comp1<comp2 then      /* check all checkboxes until the new level <
old level */
                form~item(name||x)~checked=1
                if comp1>=comp2 then
                do
                x=set
                i=set
                end
              end
            end
            if op='/' then
            do
            g=compare('/',ival)      /* the result should be 2 for the second position
(Linux) */
              if g=2 then
              form~item(name||x)~checked=1
              if g!=2 then      /* if result is unequal 2 the value is the level
declaration and not the path declaration */
              do
              comp2=delstr(ival,1,4)      /* cut the name "level" and leave the level
number */
              comp1=delstr(ilevel,1,4)
                if comp1<comp2 then     /* check all checkboxes until the new level <
old level */
                form~item(name||x)~checked=1
                if comp1>=comp2 then
```

```
                do
                x=set
                i=set
                end
              end
            end
        end
      end
    end
  end

::routine TUcode public
  xset=output3~value
    do h=0 to xset
    des=TB||h
    document~getElementbyID(des)~style~display='none' --close all forms
    end
  uset=output4~value
    do h=0 to uset
    name2='namelink'
    document~getElementbyID(name2||h)~style~visibility='hidden' --hide all links
    end
  folder=.array~of()
  f_count=1
  error=.array~of()
  failure=.array~of()
  d=1
  e=1
  home=syssearchpath('path','xmlparser.cls')
  posname=pos('xmlparser.cls',home)
  win=countstr("\",home)
  lin=countstr("/",home)
  if win>=1 then op="\"
  if lin>=1 then op="/"
  file=substr(home,1,posname-2)
  TeRAfiles=file||op||'TeRA_files'      --create paths
  lastTrun=TeRAfiles||op||'lastTrun.txt'
  c=lines(lastTrun,count)
    do i=1 to c
    cline=linein(lastTrun,i)
    dline=linein(lastTrun,i+1)
      if dline='ERROR' then
      do
      error~put(cline,d)
      d=d+1
      end
      if dline='FAILURE' then
      do
      failure~put(cline,e)
      e=e+1
      end
    end
  set=output~value
  form = document~forms~item(0)
  name='TU'
  name2='link'
    do i=0 to set
    check = form~item(name||i)~value
    get=compare(check,'level')
      if get=1 then
      do
      pos=lastpos('\',check)
        if pos\=0 then
        do
        t=delstr(check,1,pos)
        u=lastpos('.',t)
        r=delstr(t,u,9)
        end
      end
    m=error~items
      do v=1 to m
      n=error~at(v)
      u=lastpos('.',n)
      re=delstr(n,u,9)
        if re=r then
        do
        val1=form~item(name||i)~value
        val2=compare(val1,'level')
```

```
          if val2=1 then
          do
          form~item(name||i)~style~background='#FF0033'                    /* highlight
failures red */
          document~getElementbyID(name2||i)~style~visibility='visible'     /* make
links visible (errors) */
          check1 = form~item(name||i)~alt
          folder~put(check1,f_count)
          f_count=f_count+1
          end
        end
      end
    t=failure~items
      do v=1 to t
      p=failure~at(v)
      u=lastpos('.',p)
      re=delstr(p,u,9)
        if re=r then
        do
        val3=form~item(name||i)~value
        val4=compare(val3,'level')
          if val4=1 then
          do
          form~item(name||i)~style~background='#FFD000'                    /* highlight
failures orange */
          document~getElementbyID(name2||i)~style~visibility='visible'     /* make
links visible (failures) */
          check2 = form~item(name||i)~alt
          folder~put(check2,f_count)
          f_count=f_count+1
          end
        end
      end
    end
  count=folder~items
  do i=1 to count
  item=folder~at(i)
  document~getElementbyID(item)~style~display = 'inline'       /* open forms which
contain failures or errors */
  end

::routine folder_checkAll public
  form = document~forms~item(0)
  check=output5~value
  xset=output~value
    do h=0 to xset
    item_alt=form~item("TU"||h)~alt
      if item_alt=check then
      do
      item_id=form~item("TU"||h)~id
      nr_comp=compare("all",item_id)
        if nr_comp=1 then
        form~item("TU"||h)~checked=1
      end
    end
::routine folder_uncheckAll public
  form = document~forms~item(0)
  check=output5~value
  xset=output~value
    do h=0 to xset
    item_alt=form~item("TU"||h)~alt
      if item_alt=check then
      do
      item_id=form~item("TU"||h)~id
      nr_comp=compare("all",item_id)
        if nr_comp=1 then
        form~item("TU"||h)~checked=0
      end
    end
```

7.8.1.3 TeRA-OPENLASTLOG.REX

TeRA-OPENLASTLOG.REX simply opens the last generated log file.

```
/*
   name:             TeRA-openLastLog.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - opens the last run log file

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/


say "opening last logfile........."

TeRA_spec=syssearchpath("path","TeRA.rex")      --get path of folder "TeRA.rex"

bs=countstr("\",TeRA_spec)
sl=countstr("/",TeRA_spec)

   if bs>=1 then op="\"      --Windows
   if sl>=1 then op="/"      --Linux

fpos=pos("TeRA.rex",TeRA_spec)
check=substr(TeRA_spec,1,fpos-2)
logfiles=check||op||"logfiles"      --"logfile" path of folder

name=logfiles||op||logindex
qte='"'

lfi=syssearchpath("path",name)      --get path of folder "logindex"

   if lfi="" then
   call rxmessagebox "no files found"      --message if folder doesn't exist

   if lfi\="" then      --folder exists
   do
      path=lfi||op||"LogFileIndex.txt"
```

```
      last=lines(path,count)
      logname=linein(path,last)      --read last line = last logfile
      logpath_fspec=logfiles||op||logname
      logpath=qte||logpath_fspec||qte
      address cmd "iexplore.exe" logpath      --open logfile
   end
```

7.8.1.4 TeRA-RUNTTESTSANDEDIT.REX

TeRA-RUNTESTSANDEDIT.REX is the program, which actually runs the selected test units and creates the log files.

```
/*
   name:             TeRA-runTestsandEdit.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - runs and edits selected tests

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.         */
/*                                                                          */
/* This program and the accompanying materials are made available under     */
/* the terms of the Common Public License v1.0 which accompanies this       */
/* distribution. A copy is also available at the following address:          */
/* http://www.opensource.org/licenses/cpl1.0.php                            */
/*                                                                          */
/* Redistribution and use in source and binary forms, with or               */
/* without modification, are permitted provided that the following          */
/* conditions are met:                                                      */
/*                                                                          */
/* Redistributions of source code must retain the above copyright           */
/* notice, this list of conditions and the following disclaimer.            */
/* Redistributions in binary form must reproduce the above copyright        */
/* notice, this list of conditions and the following disclaimer in          */
/* the documentation and/or other materials provided with the distribution. */
/*                                                                          */
/* Neither the name of Rexx Language Association nor the names              */
/* of its contributors may be used to endorse or promote products           */
/* derived from this software without specific prior written permission.    */
/*                                                                          */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS      */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT        */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,    */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,      */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY   */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING  */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS       */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.             */
/*                                                                          */
/*--------------------------------------------------------------------------*/



say "testunits running........."
-------------------------------------------------------------
TeRA_spec=SyssearchPath("path","TeRA.rex")

bs=countstr("\",TeRA_spec)      --count backslash
sl=countstr("/",TeRA_spec)

   if bs>=1 then op="\"      --if bs>=1 then operating system = Windows
   if sl>=1 then op="/"      --if sl>=1 then operating system = linux

lp=pos("TeRA.rex",TeRA_spec)
file=substr(TeRA_spec,1,lp-2)      --path of main folder
```

```
newpath=file||op||"logfiles"
call sysmkdir newpath

------------------------------------------------------------

TeRAfiles=file||op||"TeRA_files"

ltrun_spec=TeRAfiles||op||"lastTrun.txt"
   if ltrun_spec\="" then
      do
         rc=stream(ltrun_spec,"C","close")
         call SysFileDelete ltrun_spec      --delete the old file "lastTrun.txt"
      end

acttests=TeRAfiles||op||"actTests.txt"

-------------------------------------------------------------------------

qte="'"
qtb='"'

number=lines(acttests,count)      --get the test data from "actTests.txt"
l=.list~of()
do i=1 to number
l1=linein(acttests,i)
l~insert(l1)
end

tSuite=.TestSuite~new --create a TestSuite object, which will contain all testUnits
call makeTestSuiteFromFileList l, tSuite

/* or alternatively:
   tSuite=makeTestSuiteFromFileList(l)  -- will create a testSuite object and return it
*/

tRes=.TestResult~new      /* create a TestResult object to be used to gather the test-
log infos */
tSuite~run(tRes)      /* run all the testUnits collected in the Testsuite */
/*
   or alternatively:
   tRes=tSuite~run     /* will create a testResult object and return it */
*/

intermediarypath=file||op||"intermediary.txt"      --path for "intermediary.txt"

call lineout intermediarypath,"TestCounts:"pp(tSuite~countTestCases),1
/* save # of TestCounts to logfile */
call lineout intermediarypath,"ErrorCounts:"pp(tRes~errorCount),2
/* save # of ErrorCounts to logfile */
call lineout intermediarypath,"FailureCounts:"pp(tRes~failureCount),3
/* save # of FailureCounts to logfile */
call lineout intermediarypath,"RunCounts:"pp(tRes~runCount),4
/* save # of RunCounts to logfile */

len=length(tRes~logQueue~items)+2
i=0
do item over tRes~logQueue
i=i+1
-- say "     queue item #" pp(i)~right(len)":" pp(item)
call dumpDir item          -- show contents of directory object
end


------------- now going through the logged items in 'TestCaseTable'..."-----------------

len=length(tRes~testCaseTable~items)+2
i=0
do item over tRes~testCaseTable
i=i+1
call dumpQueue tRes~testCaseTable~at(item) --show contents of directory object
end

call deleteacttests(acttests)

call edit(TeRAfiles)

rc=stream(intermediarypath,"C","close")      --to close "intermediary.txt"
call SysFileDelete intermediarypath
```

```
::requires oorexxunit.cls      --get the ooRexxUnit support
-------------------------------------------------------------------------------
::routine dumpDir
  use arg dir

  TeRA_spec=SyssearchPath("path","TeRA.rex")

  bs=countstr("\",TeRA_spec)      --count backslash
  sl=countstr("/",TeRA_spec)

     if bs>=1 then op="\"      --if bs>=1 then operating system = Windows
     if sl>=1 then op="/"      --if sl>=1 then operating system = Linux

   lp=pos("TeRA.rex",TeRA_spec)
   file=substr(TeRA_spec,1,lp-2)
-----------------------------------------------------

   intermediarypath=file||op||"intermediary.txt" --path for "intermediary.txt"

   do idx over dir
   --say "           idx="pp(idx) "value:" pp(dir~at(idx))

      call lineout intermediarypath,"idx="pp(idx) "value:" pp(dir~at(idx))

      tmpValue=dir~at(idx)      --dump content of an array or list object
         if tmpValue~hasMethod("SUPPLIER") then
         do
            do entry over tmpValue
            call lineout intermediarypath,"09"x pp(entry)
            end
         end
   end

::routine dumpQueue
  use arg queue
  do item over queue
     if item~class=.directory then      /* if a directory in hand, dump its content */
     call dumpDir(item)
  end



::routine edit public
use arg TeRAfiles
-------------------------------------------------------------------------------

TeRA_spec=SyssearchPath("path","TeRA.rex")

bs=countstr("\",TeRA_spec)      --count backslash
sl=countstr("/",TeRA_spec)      --count slash

   if bs>=1 then op="\"      --if bs>=1 then operating system = Windows
   if sl>=1 then op="/"      --if sl>=1 then operating system = Linux

lp=pos("TeRA.rex",TeRA_spec)
file=substr(TeRA_spec,1,lp-2)

TeRAfiles=file||op||"TeRA_files"
intermediarypath=file||op||"Intermediary.txt"

/*
the following code is used to name logfiles precisely, by adding the date and the time
of the testrun to the identifier
*/

crlf="0d0a"x      --CR-LF

part1="log"      --filetype

part2=date("S")      --part2=date without space

Time=.mutableBuffer~new --a new mutable buffer to change the format of time
Time~insert(Time())

/*the colons have to be deleted, because Microsoft Windows is not able to save files
with filenames
including colons*/

Time~delete(3,1)      --delete the first colon
Time~delete(5,1)      --delete the second colon
```

```
Time~insert(.,2)
Time~insert(.,5)

part3=Time~string()

userinformation=TeRAfiles||op||"userinformation.txt" --get the userinformation
a=linein(userinformation,1)

logfilename=part1"_"part2"_"part3"_"a".xml"

fpath=file||op||"logfiles"
logpath=fpath||op||logfilename

rc=stream(intermediarypath,"C","close") --to close "intermediary.txt"

linenumber=lines(intermediarypath,count) --count lines of the logfile

lasttrunarray=.array~of()
/* create new array for last testrun info (color highlighting and linking) */

lasttrunindex=1

xsl_path=file||op||"log.xsl"      --path for xsl file
-------------------------------------------------------------------------
call lineout logpath,"<?xml version='1.0'?>"
call lineout logpath,"<?xml-stylesheet type='text/xsl'  href='"xsl_path"' ?>"
call lineout logpath,"<testunit>"
-------------------------------------------------------------------------

/* now get the whole userinformation (name and comment) */

username=linein(userinformation,1)
comment=linein(userinformation,2)

call lineout logpath,"<username>"username"</username>"
call lineout logpath,"<comment>"comment"</comment>"

rc=stream(userinformation,'C','close')
call sysfiledelete userinformation


/* edit the first 4 lines of "Intermediary.txt" to get # of errors, testruns, failures
and testunits */

call lineout logpath,"<runs>"

/* test run information */
a=linein(intermediarypath,1)
Num=.mutableBuffer~new
Num~setBufferSize(1)
Num~insert(a)
b=lastpos("[",num~string)
c=lastpos("]",num~string)
d=Num~getBufferSize()
num~delete(c,d+1-c)
num~delete(1,b)
call lineout logpath,"<testruns>"num~string"</testruns>"

/* error count information */
a=linein(intermediarypath,2)
Num=.mutableBuffer~new
Num~setBufferSize(1)
Num~insert(a)
b=pos("[",num~string)
c=pos("]",num~string)
d=Num~getBufferSize()
num~delete(c,d+1-c)
num~delete(1,b)
call lineout logpath,"<errcounts>"num~string"</errcounts>"

/* failure count information */
a=linein(intermediarypath,3)
Num=.mutableBuffer~new
Num~setBufferSize(1)
Num~insert(a)
b=pos("[",num~string)
c=pos("]",num~string)
d=Num~getBufferSize()
num~delete(c,d+1-c)
```

```
num~delete(1,b)
call lineout logpath,"<failcount>"num~string"</failcount>"

/* run information */
a=linein(intermediarypath,4)
Num=.mutableBuffer~new
Num~setBufferSize(1)
Num~insert(a)
b=pos("[",num~string)
c=pos("]",num~string)
d=Num~getBufferSize()
num~delete(c,d+1-c)
num~delete(1,b)
call lineout logpath,"<runcount>"num~string"</runcount>"

call lineout logpath,"</runs>"
-------------------------------------------------------------------------

TestUnitTestNameBuffer=.mutableBuffer~new
TestUnitTestNameBuffer~setBufferSize(1)

----------create file ltrun.txt if not already existing--------------------

call sysmkdir TeRAfiles
xml_subtest_index=1
lasttrun=TeRAfiles||op||"lastTrun.txt"

lastlogfile=TeRAfiles||op||"lastlog.xml" --create path for lastlogfile
rc=stream(lastlogfile,"C","close") --close file
call sysfiledelete lastlogfile --delete the file

n_indexTrace=1
n_indexADD=1

i=5
   do while i<=linenumber
   call linein intermediarypath,i
   actline=result
----get all important positions of "Intermediary.txt"----

   posStart=pos("[startTest]",actline)

   posEnd=pos("[endTest]",actline)

   posTestSuite=pos("testCase: []",actline)

   posThe=pos("[The",actline)

   posClass=pos("class]",actline)

   posValue=pos("value: [[",actline)

   posBracket=pos("]:",actline)

   posTest=pos("[TEST_",actline)

   posFail=pos("[failure]",actline)

   posError=pos("[error]",actline)

-----------------------------------------------------------------

      if posStart>0 then      --get starttime
         do
         StartTimeBuffer=.mutablebuffer~new
         StartTimeBuffer~setBufferSize(1)
         StartTimeBuffer~insert(actline)
         STBS=StartTimeBuffer~getBufferSize()
         StartTimeBuffer~delete(posStart-3,(STBS+1)-(posStart-6))
         StartTimeBuffer~delete(1,(posStart-19))
         DateBuffer=.mutableBuffer~new      --get date
         DateBuffer~setBufferSize(1)
         Datebuffer~insert(actline)
         DBS=DateBuffer~getBufferSize()
         DateBuffer~delete(posStart-19,(STBS+1)-(posStart-19))
         DateBuffer~delete(1,(posStart-28))
         end

      if posEnd>0 then      --get endtime
         do
```

```
         EndTimeBuffer=.mutablebuffer~new
         EndTimeBuffer~setBufferSize(1)
         EndTimeBuffer~insert(actline)
         ETBS=EndTimeBuffer~getBufferSize()
         EndTimeBuffer~delete(posEnd-3,(ETBS+1)-(posEnd-6))
         EndTimeBuffer~delete(1,(posEnd-19))
         DateBuffer=.mutableBuffer~new      --get date
         DateBuffer~setBufferSize(1)
         Datebuffer~insert(actline)
         DBS=DateBuffer~getBufferSize()
         DateBuffer~delete(posEnd-19,(ETBS+1)-(posEnd-19))
         DateBuffer~delete(1,(posEnd-28))
         end

      if posThe>0 & posClass>0 then      --get testunitname
         do
         TestUnitNameBuffer=.mutableBuffer~new
         TestUnitNameBuffer~setBufferSize(1)
         TestUnitNameBuffer~insert(actline)
         TUNBS=TestUnitNameBuffer~getbuffersize()
         TestUnitNameBuffer~delete(posClass-1,(TUNBS+1)-(posClass-1))
         TestUnitNameBuffer~delete(1,(posThe+4))
         end

      if posTest>0 then      --get testname
         do
         posTestBracket=pos("]",actline,posTest+1)
         TestNameBuffer=.mutableBuffer~new
         TestNameBuffer~setBufferSize(1)
         TestNameBuffer~insert(actline)
         TNBS=TestNameBuffer~getBufferSize()
         TestNameBuffer~delete(posTestBracket,(TNBS+1)-posTestBracket)
         TestNameBuffer~delete(1,posTest)
         check1=TestUnitNameBuffer~string"/"TestNameBuffer~string";"
         posTestAttachment=pos(check1,TestUnitTestNameBuffer~string)
            if posTestAttachment=0
            then TestUnitTestNameBuffer~insert-
                 (TestunitNameBuffer~string"/"TestNameBuffer~string";")
         end
-------------------------------------------------------------------------

      if (posTestSuite>0 & posStart>0) then      --start of test suite
         do
         call lineout logpath,"<starttestsuite>"StartTimeBuffer~string-
                              "</starttestsuite>"
         call lineout logpath,"<startdatets>"DateBuffer~string-
                              "</startdatets>"
         end

      if (posTestSuite>0 & posEnd>0) then      --end of test suite
         do
         call lineout logpath,"<endtestsuite>"EndTimeBuffer~string-
                              "</endtestsuite>"
         call lineout logpath,"<enddatets>"DateBuffer~string-
                              "</enddatets>"
         call lineout logpath,"<subtestnumber id='subtestnumber' -
                              value='"xml_subtest_index-1"'>-
                              </subtestnumber>"
         call lineout logpath,"</testunit>"
         end

      if (posThe>0 & posClass>0 & posStart>0) then      --start of test case
         do
         call lineout logpath,"<maintestunit id='"TestUnitNameBuffer~string"'>"
         call lineout logpath,"<startend>"
         call lineout logpath,"<namemain>"TestUnitNamebuffer~string"</namemain>"
         call lineout logpath,"<mainstarttime>"StartTimeBuffer~string-
                              "</mainstarttime>"
         call lineout logpath,"<mainstartdate>"DateBuffer~string"</mainstartdate>"
         call lineout logpath,"</startend>"
         end

      if (posThe>0 & posClass>0 & posEnd>0) then      --end of test case
         do
         call lineout logpath,"<startend>"
         call lineout logpath,"<mainendtime>"StartTimeBuffer~string"</mainendtime>"
         call lineout logpath,"<mainenddate>"DateBuffer~string"</mainenddate>"
         call lineout logpath,"</startend>"
         call lineout logpath,"</maintestunit>"
```

```
         end

      if (posStart>0 & posTest>0) then
         do
         actline_1=linein(intermediarypath,i+1)
         posEnd=pos("[endTest]",actline_1)
            if posEnd>0 then
               do
               EndTimeBuffer=.mutablebuffer~new
               EndTimeBuffer~setBufferSize(1)
               EndTimeBuffer~insert(actline)
               ETBS=EndTimeBuffer~getBufferSize()
               EndTimeBuffer~delete(posEnd-3,(ETBS+1)-(posEnd-6))
               EndTimeBuffer~delete(1,(posEnd-19))
               call lineout logpath,"<subtest value='okay' name=-
                                    '"TestUnitNameBuffer~string"/-
                                    "TestNameBuffer~string"' -
                                    id='subtest"||xml_subtest_index"'>"
               call lineout logpath,"<name>"TestNameBuffer~string"</name>"
               call lineout logpath,"<main>"TestUnitNameBuffer~string"</main>"
               call lineout logpath,"<linestart><substarttime>"StartTimeBuffer~string-
                                    "</substarttime>"
               call lineout logpath,"<substartdate>"DateBuffer~string"</substartdate>-
                                    </linestart>"
               call lineout logpath,"<lineend><subendtime>"-
                                    EndTimeBuffer~string"</subendtime>"
               call lineout logpath,"<subenddate>"-
                                    DateBuffer~string"</subenddate></lineend>"
               call lineout logpath,"<status>OKAY</status>"
               call lineout logpath,"</subtest>"
               xml_subtest_index=xml_subtest_index+1
               end
            else  --if test raises an error or failure then execute the following code
               do
                  do ix=15 to 500
                  actline_ix=linein(intermediarypath,i+ix)
                  posEnd=pos("[endTest]",actline_ix)
                     if posEnd\=0 then
                     ix=500
                  end

                  if posEnd>0 then
--extract the end time - this is the last line of this test
                     do
                     actline_2=linein(intermediarypath,i+2)
                     posError=pos("[error]", actline_2)
                     posFail=pos("[failure]", actline_2)
                     EndTimeBuffer=.mutablebuffer~new
                     EndTimeBuffer~setBufferSize(1)
                     EndTimeBuffer~insert(actline)
                     ETBS=EndTimeBuffer~getBufferSize()
                     EndTimeBuffer~delete(posEnd-3,(ETBS+1)-(posEnd-6))
                     EndTimeBuffer~delete(1,(posEnd-19))
                        if posError>0 then /* if "[error]" can be found create an error
section in the xml file */
                           do
                           call lineout logpath,"<subtest value='error' -
                                                name='"TestUnitNameBuffer~string"/-
                                                "TestNameBuffer~string"' -
                                                id='subtest"||xml_subtest_index"'>"
                           call lineout logpath,"<name>"TestNameBuffer~string"</name>"
                           call lineout logpath,"<main>"TestUnitNameBuffer~string-
                                                "</main>"
                           call lineout logpath,"<linestart><substarttime>"-
                                                StartTimeBuffer~string"</substarttime>"
                           call lineout logpath,"<substartdate>"DateBuffer~string"-
                                                </substartdate></linestart>"
                           call lineout logpath,"<lineend><subendtime>"-
                                                EndTimeBuffer~string"</subendtime>"
                           call lineout logpath,"<subenddate>"DateBuffer~string-
                                                "</subenddate></lineend>"
                           call lineout logpath,"<status>ERROR</status>"
                           call lineout logpath,"<infotable value='error'>"
                           errarray=.array~of()
                           errarray~put(TestUnitNameBuffer~string,1)
                           errarray~put("ERROR",2)
                           errarray~put("//////",3)
                           lasttrunarray~put(errarray,lasttrunindex)
```

```
                        lasttrunindex=lasttrunindex+1
                        xml_subtest_index=xml_subtest_index+1
                        end
                     if posFail>0 then      /* if "[failure]" can be found create a
failure section in the xml file */
                        do
                        call lineout logpath,"<subtest value='failure' -
                                             name='"TestUnitNameBuffer~string"/-
                                             "TestNameBuffer~string"' -
                                             id='subtest"||xml_subtest_index"'>"
                        call lineout logpath,"<name>"TestNameBuffer~string"</name>"
                        call lineout logpath,"<main>"-
                                             TestUnitNameBuffer~string"</main>"
                        call lineout logpath,"<linestart><substarttime>"-
                                             StartTimeBuffer~string"</substarttime>"
                        call lineout logpath,"<substartdate>"DateBuffer~string"-
                                             </substartdate></linestart>"
                        call lineout logpath,"<lineend><subendtime>"-
                                             EndTimeBuffer~string"</subendtime>"
                        call lineout logpath,"<subenddate>"DateBuffer~string-
                                             "</subenddate></lineend>"
                        call lineout logpath,"<status>FAILURE</status>"
                        call lineout logpath,"<infotable value='failure'>"
                        failarray=.array~of()
                        failarray~put(TestUnitNameBuffer~string,1)
                        failarray~put("FAILURE",2)
                        failarray~put("//////",3)
                        lasttrunarray~put(failarray,lasttrunindex)
                        lasttrunindex=lasttrunindex+1
                        xml_subtest_index=xml_subtest_index+1
                        end
                  end
               end
            end

         if (posStart=0 & posEnd=0 & posFail=0 & posError=0) then
         do
         check4=countstr("*-*",actline)
            if check4=1 then      --additional info in this line (Traceback information)
            do
            call lineout logpath,"<lineadd index='TRACEBACKadd' -
                                 n_index='"n_indexTrace"'><info col='span'>-
                                      "actline"-
                                 </info></lineadd>"
            n_indexTrace=n_indexTrace+1
            end
         check3=compare("idx=",actline)
            if check3=5 then      --index information
               do
               valpos=pos("value:",actline)      /* extract the value information */
               part1=substr(actline,1,valpos-1)      /* cut the value; result=index */
               part2=substr(actline,valpos)      /* cut the index; result=value */
               bra1=pos("[",actline)
                  if bra1=5 then
                     do
                     bra2=pos("]",actline,bra1+1)
                     sortind=substr(actline,bra1+1,(bra2-1)-bra1)
                     end
               call lineout logpath,"<line index='"sortind"'><info>"part1-
                                    "</info><info >"part2"</info></line>"
               end
            if (check3\=5 & check4=0) then
/* like the traceback information, this is additional information */
            do
            call lineout logpath,"<lineadd index='ADDITIONALadd' -
                                 n_index='"n_indexADD"'>-
                                 <info col='span'>   -
                                   "actline"</info></lineadd>"
            n_indexADD=n_indexADD+1
            end
            check1=linein(intermediarypath,i+1)
            check2=pos("[endTest]",check1)
               if check2\=0 then
                  do
                  call lineout logpath,"</infotable>"
                  call lineout logpath,"</subtest>"
                  end
         end
```

```
i=i+1
end

------------------------------------------------------------------------
/* the following code loops through the lasttrunarray to write         */
/* information into lasttrun.txt                                       */
/* purpose: delete redundant information                               */
------------------------------------------------------------------------
lasttTeRAems=lasttrunarray~items /* get items for the first loop (multidimensional
array!) */
   do k=1 to lasttTeRAems
   ltrun_array1=lasttrunarray~at(k)
   ltrun_items1=ltrun_array1~items /* get items for the second loop */
      do l=1 to ltrun_items1
      ltrun_act_item1=ltrun_array1~at(l)
      ltrun_act_item_pos1=ltrun_array1~at(1)
         if k=1 then      -- if first array, write information
         call lineout lasttrun,ltrun_act_item1
            if k\=1 then      /* if not first array, check the array (actual position -
1), if information was already written */
               do
               ltrun_array0=lasttrunarray~at(k-1)
               ltrun_act_item0_pos1=ltrun_array0~at(1)
                  if ltrun_act_item_pos1\=ltrun_act_item0_pos1 then
                  call lineout lasttrun,ltrun_act_item1
               end
      end
   end


--------create Logfileindex.txt if not already existing------------
check3=syssearchpath("path","LogIndex")
   if check3="" then
      do
      pathname1=fpath||op||"LogIndex"
      pathname2=pathname1||op||"LogFileIndex.txt"
      call sysmkdir pathname1
      call lineout pathname2,logfilename
      end
   else call lineout pathname2,logfilename
call copylastlog logpath,lastlogfile
------------------------------------------------------------------------
::routine deleteacttests public
use arg acttests

rc=stream(acttests,"C","close")
call sysfiledelete acttests

::routine copylastlog public      /* copy the last logfile to ...TeRA_files\lastlog.xml
*/
use arg logpath,lastlogfile

rc=stream(logpath,"C","close")

address cmd 'copy' logpath lastlogfile
```

7.8.1.5 TeRA-COMPARELOGS.REX

TeRA-COMPARELOGS.REX generates the second user interface TeRA-COMPARELOGS.HTA.

```
/*
   name:             TeRA-compareLogs.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - provides navigation for comparing log files

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/
```

```
/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                    */
/*                                                                          */
/* This program and the accompanying materials are made available under     */
/* the terms of the Common Public License v1.0 which accompanies this       */
/* distribution. A copy is also available at the following address:         */
/* http://www.opensource.org/licenses/cpl1.0.php                            */
/*                                                                          */
/* Redistribution and use in source and binary forms, with or               */
/* without modification, are permitted provided that the following          */
/* conditions are met:                                                      */
/*                                                                          */
/* Redistributions of source code must retain the above copyright           */
/* notice, this list of conditions and the following disclaimer.            */
/* Redistributions in binary form must reproduce the above copyright        */
/* notice, this list of conditions and the following disclaimer in          */
/* the documentation and/or other materials provided with the distribution. */
/*                                                                          */
/* Neither the name of Rexx Language Association nor the names              */
/* of its contributors may be used to endorse or promote products           */
/* derived from this software without specific prior written permission.    */
/*                                                                          */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS      */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT        */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,    */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,      */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY   */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING  */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS       */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.             */
/*                                                                          */
/*--------------------------------------------------------------------------*/


say "opening compareLogs........."

TeRA_spec=SyssearchPath("path","TeRA.rex")      --get the right directory

bs=countstr("\",TeRA_spec)
sl=countstr("/",TeRA_spec)

   if bs>=1 then op="\"      -- operating system Windows
   if sl>=1 then op="/"      -- operating system Linux

lp=pos("TeRA.rex",TeRA_spec)
file=substr(TeRA_spec,1,lp-2)

name=file||op||"TeRA-compareLogs.hta"
call sysfiledelete name      --delete the old TeRA-compareLogs.hta

qte='"'
qtb="'"

call lineout name,"<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'>"
call lineout name,"<html>"
call lineout name,"<head>"

call lineout name,"<title>TeRA - compare Tests</title>"
call lineout name,"<link rel='stylesheet' href='styles.css' type='text/css'>"
-----------------------------------------------------------------
--start script
-----------------------------------------------------------------

call lineout name,"<script language='object rexx' -
                  src='TeRA_compareLogs_script.rex'></script>"
-----------------------------------------------------------------
--end script
-----------------------------------------------------------------

call lineout name,"</head>"
call lineout name,"<body>"
call lineout name,"<div id='content_container'>"
call lineout name,"<h1>COMPARE LOG FILES</h1>"

filespath=file||op||"logfiles"

FilePattern=filespath||op||"*.xml"  --search for .xml files
flags      ="FSO" /* search files only, in all subdirs, return fully qualified path */
```

```
call SysFileTree FilePattern, "files.", flags

   if files.0=0 then
      do
      call rxmessagebox "no logfiles found - aborting"
      exit -1
      end

mystem.=.stem~new
l=.list~new      -- create a list
do i=1 to files.0      -- add each found file into the list
l~insert(files.i)
mystem.i=files.i
end

j=l~items      --get number of list items for descending list

mystem.0=j      --set index to number of items

call SysStemSort "mystem.","D"      --sort stem descending

call lineout name,"<form name='LogFileForm1' id='LogFileForm1'>" --ascending form
call lineout name,"<table id='table1' style='display:inline;'>"

y=l~makearray      --list => array
arr=y~supplier      --supplier => array

num=l~items

i=1
do while i<=num      --loop through all items in the list (ascending table)
htmlname=LogFile||i
g=arr~item()
backlastpos=lastpos(op,g)
lName=substr(g,(backlastpos+1))
call lineout name,"<tr><td><input type='radio' onclick='call checked' name='a' -
                  id='"htmlname||1"' value='"g"'>" lName"</input></td><td>-
                  <input type='radio' onclick='call checked' name='b' -
                  id='"htmlname||2"' value='"g"'>" lName"</input>-
                  </td></tr>"
arr~next
i=i+1
end

call lineout name,"</table>"
call lineout name,"</form>"

call lineout name,"<form name='LogFileForm2' id='LogFileForm2'>"      --descending form
call lineout name,"<table id='table2' style='display:none;'>"

x=1
do while x<=mystem.0      --loop through all stem items (descending table)
htmlname=Logfile||x
backlastpos=lastpos(op,mystem.x)
lName=substr(mystem.x,(backlastpos+1))
call lineout name,"<tr><td><input type='radio' onclick='call checked' name='a' -
                  id='"htmlname||3"' value='"mystem.x"'>" lName"</input></td>-
                  <td><input type='radio' onclick='call checked' name='b' -
                  id='"htmlname||4"' value='"mystem.x"'>" lName"-
                  </input></td></tr>"
x=x+1
end

call lineout name,"</table>"
call lineout name,"</form>"

call lineout name,"</div>"
call lineout name,"<div id='footer1'>"
call lineout name,"<input type='button' onclick='call uncheckAll' id='uncheck' -
                  name='uncheck' value='uncheck'>"
call lineout name,"<input type='button' onclick='call arrange' id='arrange' -
                  name='arrange' value='descending'>"
call lineout name,"</div>"
call lineout name,"<div id='footer2'>"
call lineout name,"<input type='button' onclick='call compareLogs' id='compareLogs' -
                  name='compareLogs' value='compare logs'>"
call lineout name,"<input type='button' onclick='call showLogFile' id='showLog' -
                  name='showLog' value='show logfile'>"
call lineout name,"</div>"
```

```
call lineout name,"<p type='textarea' size='30' id='output' value="num">"
call lineout name,"</body>"
call lineout name,"</html>"

rc=stream(name,"C","close")      --to close "TeRA-compareLogs.hta"
address cmd name
```

7.8.1.6 TeRA_COMPARELOGS_SCRIPT.REX

Contains the routines for TeRA-COMPARELOGS.HTA.

```
/*
   name:             TeRA-compareLogs_script.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - provides routines for TeRA-compareLogs.hta

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/



::routine showLogFile public
  set=output~value
  LogFileList=.list~of()
  name='LOGFILE'
    do x=1 to 4
      do i=1 to set      --loop through all radio buttons
      check =document~getelementbyid(name||i||x)~checked      --if checked
        if check=1 then
        do
        selection=document~getelementbyid(name||i||x)~value      --then get value
        LogFileList~insert(selection)      --insert value into logfilelist
        end
      end
    end
```

```
  noi=LogfileList~items      --get number of items
    if noi=1 then
    do
    address cmd 'iexplore' selection
    end
    if noi>1 then      --if (# items)>1
    do
    call rxmessagebox 'too many files chosen!','wrong number of files'
    end
    if noi<1 then      --if (# items)<1
    do
    call rxmessagebox 'choose at least one file!','wrong number of files'
    end

::routine checked public
  set=output~value
  name='LOGFILE'
    do i=1 to set      --loop for ascending sort
    check = document~getElementbyId(name||i||1)~checked      /* if this element checked
then check=1 */
      if check=1 then
      document~getElementbyId(name||i||2)~disabled=1      /* if check=1 then disable the
counterpart */
      if check\=1 then
      document~getElementbyId(name||i||2)~disabled=0      /* if check\=1 then enable
counterpart */
    end
    do i=1 to set
    check = document~getElementbyId(name||i||2)~checked
      if check=1 then
      document~getElementbyId(name||i||1)~disabled=1
      if check\=1 then
      document~getElementbyId(name||i||1)~disabled=0
    end
    do i=1 to set      --loop for descending sort
    check = document~getElementbyId(name||i||3)~checked
      if check=1 then
      document~getElementbyId(name||i||4)~disabled=1
      if check\=1 then
      document~getElementbyId(name||i||4)~disabled=0
    end
    do i=1 to set
    check = document~getElementbyId(name||i||4)~checked
      if check=1 then
      document~getElementbyId(name||i||3)~disabled=1
      if check\=1 then
      document~getElementbyId(name||i||3)~disabled=0
    end

::routine uncheckAll public      --onclick uncheck all radio buttons
  set=output~value
  name='LOGFILE'
    do x=1 to 4
      do i=1 to set
      document~getElementbyID(name||i||x)~checked=0
      document~getElementbyID(name||i||x)~disabled=0
      end
    end

::routine arrange public  -- display ascending table or display descending table
  check1=document~getelementbyid('table1')~style~display
  check2=document~getelementbyid('table2')~style~display
    if (check1='inline' & check2='none') then
    do
    document~getelementbyid('table1')~style~display='none'
    document~getelementbyid('table2')~style~display='inline'
    document~getelementbyid('arrange')~value='ascending'
    call uncheckAll
    end
    if (check1='none' & check2='inline') then
    do
    document~getelementbyid('table1')~style~display='inline'
    document~getelementbyid('table2')~style~display='none'
    document~getelementbyid('arrange')~value='descending'
    call uncheckAll
    end

::routine compareLogs public
```

```
  set=output~value      --get number of checkboxes
  LogFileList=.list~of()
  form = document~forms~item(0)
  crlf='0d0a'x    -- CR-LF
  name='LOGFILE'
    do x=1 to 4 --do the loop 4 times for both sides and ascending and descending
      do i=1 to set
      check = document~getelementbyid(name||i||x)~checked
        if check=1 then
        do
        selection=document~getelementbyid(name||i||x)~value
        LogFileList~insert(selection)
        end
      end
    end
  noi=LogfileList~items
    if noi<2 then
    do
    call rxmessagebox 'choose two files!','wrong number of files' --less than two files
    i=set
    end
    if noi>2 then
    do
    call rxmessagebox 'too many files chosen!','wrong number of files' /* too many files
are chosen */
    i=set
    end
  name1=LogFileList~at(0)
  name2=LogFileList~at(1)
    if noi=2 then      --exactly two files chosen
    do
      if rxmessagebox('Compare'crlf name1 crlf' with 'crlf name2,'compare',-
                      'OKCANCEL','question')=1 then
      do
      check=syssearchpath('path','xmlparser.cls')
      bs=countstr('\',check)
      sl=countstr('/',check)
        if bs>=1 then op='\'      --Windows
        if sl>=1 then op='/'      --Linux
      lp=pos('xmlparser.cls',check)
      xmlpath=substr(check,1,lp-2)
      filespath=xmlpath||op||'TeRA_files'||op||'compTests.txt'
      call lineout filespath,name1      --store the first logfile's name
      call lineout filespath,name2      --store the second logfile's name
      rc=stream(filespath,'C','close')
      address cmd 'rexx TeRA-compareTests.rex'      --start the actual testing
      end
    end
```

7.8.1.7 TeRA-COMPARETESTS.REX

TeRA-COMPARETESTS.REX compares the log files, which were selected in TeRA-COMPARELOGS.HTA.

```
/*
   name:            TeRA-compareTests.rex
   author:          Rainer Kegel
   date:            2007-01-02

   purpose:         part of the Test Runner Application for ooRexxUnit
                    - actual comparison of log files

   license:         CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
```

```
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/



say "reading logfiles........."

check=syssearchpath("path","xmlparser.cls")      --get path of "TeRA_files"

bs=countstr("\",check)
sl=countstr("/",check)

   if bs>=1 then op="\"      --Windows
   if sl>=1 then op="/"      --Linux

lp=pos("xmlparser.cls",check)
file=substr(check,1,lp-2)

checkpath=file||op||"TeRA_files"||op||"compTests.txt"
log1=linein(checkpath,1)      -- read which logs are going to be compared

lpos1=lastpos(op,log1)
name=substr(log1,lpos1+1)
pos1=lastpos(".",name)
name1=substr(name,1,pos1-1)      --edit the first name (without path and ".xml")

log2=linein(checkpath,2)      -- read which logs are going to be compared

lpos2=lastpos(op,log2)
name=substr(log2,lpos2+1)
pos2=lastpos(".",name)
name2=substr(name,1,pos2-1)      --edit the second name (without path and ".xml")

c1=name1"___"name2".xml"      --get both name-combinations
c2=name2"___"name2".xml"      --get both name-combinations

pspec=syssearchpath("path","compared_files")      --search folder

g1=pspec||op||c1
g2=pspec||op||c2

res1=syssearchpath("path",g1)      --check whether file is already existing
res2=syssearchpath("path",g2)      --check whether file is already existing

   if res1\="" then
   address cmd 'iexplore.exe' g1      -- if file is existing, open it

   if res2\="" then
   address cmd 'iexplore.exe' g2      -- if file is existing, open it

   if (res1="" & res2="") then      --create file,if not existing
   do
      filename=name1"___"name2".xml"
```

```
      newpath=file||op||"compared_files"

         if pspec="" then      --create folder "compared_files", if not existing
         call sysmkdir newpath

      comppath=newpath||op||filename      --new name of file

      xsl_path=file||op||"complog.xsl"
-------------------------------------------------------------------------------
/* create new xml file */
-------------------------------------------------------------------------------

      call lineout comppath,"<?xml version='1.0'?>"
      call lineout comppath,"<?xml-stylesheet type='text/xsl'  href='"xsl_path"' ?>"
      call lineout comppath,"<testunit>"

      nochanges=0

      do i=1 to 2      --loop for both logs that are compared

         if i=1 then
         log=log1

         if i=2 then
         log=log2

      .local~insert.index=1      --save counter to local environment
      .local~insert.array=.array~new      --save new array to local environment
      .local~index.TeRA=1
      .local~comp.log=.array~new

      say "storing results........."

      parser = .myparser~new()
      errortxt = parser~parse_file(log)      --parse log-xml

         if i=1 then      --fill first array
         do
            attrarr1=.comp.log
            i1=attrarr1~items
         end

         if i=2 then      --fill second array
         do
            attrarr2=.comp.log
            i2=attrarr2~items
         end
      end

/*
search both arrays for changes
*/
      say "creating file........."

      call lineout comppath,"<changes>"
      call lineout comppath,"<headline>CHANGES FOUND</headline>"
      call lineout comppath,"<name1>"name1"</name1>"
      call lineout comppath,"<name2>"name2"</name2>"

      do x=1 to i1      --loop through all items ("attrarr1")
      a1=attrarr1~at(x)      --get actual array (a1=multi dimensional array)
      a1check=a1~at(1)      --get testname
      a1status=a1~at(2)      --get status

         do y=1 to i2      --loop through all items ("attrarr2")
         a2=attrarr2~at(y)      --get actual array (a2=multi dimensional array)
         a2check=a2~at(1)      --get testname
         a2status=a2~at(2)      --get status

         if a1check=a2check then      --compare test name
         do

         nindexTrace=1
         nindexADD=1

            if a1status\=a2status then      /* compare status (if not same => new
section in the in xml "changes") */
            do
                n=a1~items
                m=a2~items
```

```
                  if n>=m then      --check which array has more items
                  t=n
                  else
                  t=m

               diffstatus1=.array~of()
               dsindex1=1
               diffstatus2=.array~of()
               dsindex2=1

               do q=1 to t      --loop through both arrays at once
                  if q<=n then      --if counter<# of items then write information
                  do
                     get1=a1~at(q)
                     diffstatus1~put(get1,dsindex1)      --insert info
                     dsindex1=dsindex1+1
                  end

                  if q>n then      --if counter>#of items then write empty tag
                  do
                     get1=''
                     diffstatus1~put(get1,dsindex1)
                     dsindex1=dsindex1+1
                  end

                  if q<=m then
                  do
                     get2=a2~at(q)
                     diffstatus2~put(get2,dsindex2)
                     dsindex2=dsindex2+1
                  end

                  if q>m then
                  do
                     get2=''
                     diffstatus2~put(get2,dsindex2)
                     dsindex2=dsindex2+1
                  end
               end

               u=diffstatus1~items
               m=diffstatus2~items
                  if u=m then
                  nui=u
                  if u>m then
                  nui=u
                  if m>u then
                  nui=m

               call lineout comppath,"<tutable>"
               call lineout comppath,"<table1>"      --table for first log file
               do w=1 to nui
               get1=diffstatus1~at(w)
               g1=compare("idx=",get1)
               g2=compare("value:",get1)
               g3=countstr("*-*",get1)
               g4=countstr("]",get1)
               g5=countstr("[",get1)
                  if g1=5 then      --if item is an index item
                  do
                     n1=pos("[",get1)
                     n2=pos("]",get1)
                     attnameadd=substr(get1,n1+1,(n2-1)-n1)
                     call lineout comppath,"<line1 index='"attnameadd"'>-
                                          <complog1 check='d'>"get1"-
                                          </complog1>"      /* attribute 'index'
important to sort the lines */
                  end
                  if g2=8 then      --value item
                  call lineout comppath,"<complog1>"get1"</complog1></line1>"
                  if g3>=1 then      --"traceback" info
                  do
                  call lineout comppath,"<line1 index='TRACEBACKadd' -
                                       nindex='"nindexTrace"'><complog1 -
                                       col='span'>"get1"</complog1></line1>"
                  nindexTrace=nindexTrace+1
                  end
```

```
                  if (g4>=3 & g5>=3 & g2\=8) then      --"Additional" info
                  do
                  call lineout comppath,"<line1 index='ADDITIONALadd' -
                                        nindex='"nindexADD"'><complog1 -
                                        col='span'>"get1"</complog1>-
                                        </line1>"
                  nindexADD=nindexADD+1
                  end
                  if (g1<5 & g2<8 & g3=0 & g4<3 & get1\='') then --testname
                  call lineout comppath,"<line1 index='AAAAAAAAAA'>-
                                        <complog1>"get1"</complog1></line1>"
                  if get1='' then      --test unit name
                  call lineout comppath,"<line1 index='CCCCCCCCCC'><complog1>-
                                         </complog1></line1>"
               end

               call lineout comppath,"</table1>"
               call lineout comppath,"<table2>"
               do w=1 to nui
               get2=diffstatus2~at(w)

               /* checks */
               g1=compare("idx=",get2)
               g2=compare("value:",get2)
               g3=countstr("*-*",get2)
               g4=countstr("]",get2)
               g5=countstr("[",get2)
                  if g1=5 then
                  do
                     n1=pos("[",get2)
                     n2=pos("]",get2)
                     attnameadd=substr(get2,n1+1,(n2-1)-n1)
                     call lineout comppath,"<line2 index='"attnameadd"'>-
                                           <complog2 check='d'>"get2"-
                                           </complog2>"
                  end
                  if g2=8 then
                  call lineout comppath,"<complog2>"get2"</complog2>-
                                        </line2>"
                  if g3>=1 then
                  do
                  call lineout comppath,"<line2 index='TRACEBACKadd' -
                                         nindex='"nindexTrace"'><complog2 -
                                         col='span'>"get2"</complog2>-
                                         </line2>"
                  nindexTrace=nindexTrace+1
                  end
                  if (g4>=3 & g5>=3 & g2\=8) then
                  do
                  call lineout comppath,"<line2 index='ADDITIONALadd' -
                                        nindex='"nindexADD"'><complog2 -
                                        col='span'>"get2"</complog2>-
                                        </line2>"
                  nindexADD=nindexADD+1
                  end
                  if (g1<5 & g2<8 & g3=0 & g4<3 &get2\='') then
                  call lineout comppath,"<line2 index='AAAAAAAAAA'>-
                                        <complog2>"get2"</complog2>-
                                        </line2>"
                  if get2='' then
                  call lineout comppath,"<line2 index='CCCCCCCCCC'>-
                                        <complog2></complog2></line2>"
               end
               call lineout comppath,"</table2>"
               call lineout comppath,"</tutable>"
            end

              if a1status=a2status then      --if status is the same
              do
                 c=1
                 d=1
                 point=0      /* if there are no changes found, point stays =0, else
point=1 */
                 int_arr1=.array~of()
                 int_arr2=.array~of()
                 n=a1~items
                 do q=1 to n      --loop through all items of a1
                 get1=a1~at(q)
```

```
                  get2=a2~at(q)

                     if get1=get2 then      --compare all information
                     do
                        check1=compare("idx=",get1)      --index
                        check2=compare("value:",get1)      --value
                           if check1=5 then      --index item
                           do
                              n1=pos("[",get1)
                              n2=pos("]",get1)
                              attnameadd=substr(get1,n1+1,(n2-1)-n1)
                              int_arr1~put("<line1 index='"attnameadd"'>-
                                          <complog1 check='d'>"get1"-
                                          </complog1>",c)
                              c=c+1
                           end
                           if check2=8 then      --value item
                           do
                              int_arr1~put("<complog1>"get1"</complog1>-
                                          </line1>",c)
                              c=c+1
                           end
                           if (check1\=5 & check2\=8) then      /* additional info,
traceback info or test name */
                           do
                              cstr=countstr("*-*",get1)
                              check3=countstr("[",get1)
                              check4=countstr("]",get1)
                                 if cstr>=1 then
                                 do
                                 int_arr1~put("<line1 index='TRACEBACKadd' -
                                             nindex='"nindexTrace"'>-
                                             <complog1 col='span'>"get1"-
                                             </complog1></line1>",c)
                                 nindexTrace=nindexTrace+1
                                 end
                                 if (check3>=3 & check4>=3) then
                                 do
                                 int_arr1~put("<line1 index='ADDITIONALadd' -
                                             nindex='"nindexADD"'><complog1 -
                                             col='span'>"get1"</complog1>-
                                             </line1>",c)
                                 nindexADD=nindexADD+1
                                 end
                                 if (check3<3 & check4<3 & cstr<1) then
                                 int_arr1~put("<line1 index='AAAAAAAAAA'>-
                                             <complog1 col='span'>"get1"-
                                             </complog1></line1>",c)
                              c=c+1
                           end
                     end

                     if get1\=get2 then      --if information is not the same
                     do
                        point=1
                        check1=compare("idx=",get1)
                        check2=compare("value:",get1)
                           if check1=5 then      --index item
                           do
                              n1=pos("[",get1)
                              n2=pos("]",get1)
                              attnameadd=substr(get1,n1+1,(n2-1)-n1)
                              int_arr1~put("<line1 index='"attnameadd"'>-
                                          <complog1 value='diff' -
                                          check='d'>"get1"</complog1>",-
                                          c)
                              c=c+1
                           end
                           if check2=8 then      --value item
                           do
                              int_arr1~put("<complog1 value='diff'>"-
                                          get1"</complog1></line1>",c)
                              c=c+1
                           end
                           if (check1\=5 & check2\=8) then      /* additional info,
traceback info or test name */
                           do
                              cstr=countstr("*-*",get1)
```

```
                        check3=countstr("[",get1)
                        check4=countstr("]",get1)
                           if cstr>=1 then
                           do
                           int_arr1~put("<line1 index='TRACEBACKadd' -
                                        nindex='"nindexTrace"'><complog1 -
                                        value='diff' col='span'>"get1"-
                                        </complog1></line1>",c)
                           nindexTrace=nindexTrace+1
                           end
                           if (check3>=3 & check4>=3) then
                           do
                           int_arr1~put("<line1 index='ADDITIONALadd' -
                                        nindex='"nindexADD"'><complog1 -
                                        value='diff' col='span'>"get1"-
                                        </complog1></line1>",c)
                           nindexADD=nindexADD+1
                           end
                           if (check3<3 & check4<3 & cstr<1) then
                           int_arr1~put("<line1 index='AAAAAAAAAA'>-
                                        <complog1 value='diff' col='span'>-
                                        "get1"</complog1></line1>",c)
                        c=c+1
                     end
                  end
               end
               do q=1 to n       --loop through all items of a2
               get1=a1~at(q)
               get2=a2~at(q)
                  if get1=get2 then      -- info is the same
                  do
                     check1=compare("idx=",get2)
                     check2=compare("value:",get2)
                        if check1=5 then
                        do
                           n1=pos("[",get2)
                           n2=pos("]",get2)
                           attnameadd=substr(get2,n1+1,(n2-1)-n1)
                           int_arr2~put("<line2 index='"attnameadd"'><complog2 -
                                        check='d'>"get2"</complog2>",d)
                           d=d+1
                        end
                        if check2=8 then
                        do
                           int_arr2~put("<complog2>"get2"</complog2></line2>",d)
                           d=d+1
                        end
                        if (check1\=5 & check2\=8) then
                        do
                           cstr=countstr("*-*",get2)
                           check3=countstr("[",get2)
                           check4=countstr("]",get2)
                              if cstr>=1 then
                              do
                              int_arr2~put("<line2 index='TRACEBACKadd' -
                                           nindex='"nindexTrace"'><complog2 -
                                           col='span'>"get2"</complog2>-
                                           </line2>",d)
                              nindexTrace=nindexTrace+1
                              end
                              if (check3>=3 & check4>=3) then
                              do
                              int_arr2~put("<line2 index='ADDITIONALadd' -
                                           nindex='"nindexADD"'><complog2 -
                                           col='span'>"get2"</complog2>-
                                           </line2>",d)
                              nindexADD=nindexADD+1
                              end
                              if (check3<3 & check4<3 & cstr<1) then
                              int_arr2~put("<line2 index='AAAAAAAAAA'><complog2 -
                                           col='span'>"get2"</complog2>-
                                           </line2>",d)
                           d=d+1
                        end
                        if get1\=get2 then
                        do
                           point=1
```

```
                        check1=compare("idx=",get2)
                        check2=compare("value:",get2)
                           if check1=5 then
                           do
                              n1=pos("[",get2)
                              n2=pos("]",get2)
                              attnameadd=substr(get2,n1+1,(n2-1)-n1)
                              int_arr2~put("<line2 index='"attnameadd"'><complog2-
                                           value='diff' check='d'>"get2"-
                                           </complog2>",d)
                              d=d+1
                           end
                           if check2=8 then
                           do
                              int_arr2~put("<complog2 value='diff'>"get2"-
                                          </complog2>-
                                           </line2>",d)
                              d=d+1
                           end
                           if (check1\=5 & check2\=8) then
                           do
                              cstr=countstr("*-*",get2)
                              check3=countstr("[",get2)
                              check4=countstr("]",get2)
                                 if cstr>=1 then
                                 do
                                 int_arr2~put("<line2 index='TRACEBACKadd' -
                                              nindex='"nindexTrace"'><complog2 -
                                              value='diff' col='span'>"get2"-
                                              </complog2></line2>",d)
                                 nindexTrace=nindexTrace+1
                                 end
                                 if (check3>=3 & check4>=3) then
                                 do
                                 int_arr2~put("<line2 index='ADDITIONALadd' -
                                              nindex='"nindexADD"'><complog2 -
                                              value='diff' col='span'>"get2"-
                                              </complog2></line2>",d)
                                 nindexADD=nindexADD+1
                                 end
                                 if (check3<3 & check4<3 & cstr<1) then
                                 int_arr2~put("<line2 index=-
                                             'AAAAAAAAAA'><complog2-
                                              value='diff' col='span'>"get2"-
                                              </complog2></line2>",d)
                              d=d+1
                           end
                     end
                end

                if point=1 then      --changes found
                do
                   nochanges=1
                   call lineout comppath,"<tutable>"
                   call lineout comppath,"<table1>"
                   m=int_arr1~items
                   do q=1 to m      --create first table
                   get=int_arr1~at(q)
                   call lineout comppath,get
                   end
                   call lineout comppath,"</table1>"
                   call lineout comppath,"<table2>"
                   u=int_arr2~items
                   do q=1 to u      --create second table
                   get=int_arr2~at(q)
                   call lineout comppath,get
                   end
                   call lineout comppath,"</table2>"
                   call lineout comppath,"</tutable>"
                end
             end
             end
          end
       end

       if nochanges=0 then
       do
```

```
               call lineout comppath,"<tutable>"
               call lineout comppath,"<table1>"
               call lineout comppath,"<line1><complog1>no changes found</complog1>-
                                      </line1>"
               call lineout comppath,"</table1>"
               call lineout comppath,"<table2>"
               call lineout comppath,"<line2><complog2>no changes found</complog2>-
                                      </line2>"
               call lineout comppath,"</table2>"
               call lineout comppath,"</tutable>"
            end

         call lineout comppath,"</changes>"

/*
now write the information, which tests run in which logfile (first for log1 and then for
log2)
*/


         call lineout comppath,"<log1>"
         call lineout comppath,"<headline>"name1"</headline>"
         testname_bothindex=1
         testname_onceindex=1

         do x=1 to i1
         a1=attrarr1~at(x)
         a1check=a1~at(1)
         a1status=a1~at(2)
         to_comp=0
         do y=1 to i2
         a2=attrarr2~at(y)
         a2check=a2~at(1)
            if a1check=a2check then
            do
               call lineout comppath,"<testname id='both"||testname_bothindex"' -
                                      value='1'>"a1check"</testname>"
               to_comp=1
               testname_bothindex=testname_bothindex+1
            end
         end

         if to_comp=0 then
         do
            call lineout comppath,"<testname id='once"||testname_onceindex"' -
                                   value='not'>"a1check"</testname>"
            testname_onceindex=testname_onceindex+1
         end
         end
      call lineout comppath,"</log1>"

      call lineout comppath,"<log2>"
      call lineout comppath,"<headline>"name2"</headline>"
      do y=1 to i2
      a2=attrarr2~at(y)
      a2check=a2~at(1)
      a2status=a2~at(2)
      to_comp=0
      do x=1 to i1
      a1=attrarr1~at(x)
      a1check=a1~at(1)
         if a2check=a1check then
         do
            call lineout comppath,"<testname id='both"||testname_bothindex"' -
                                   value='1'>"a2check"</testname>"
            testname_bothindex=testname_bothindex+1
            to_comp=1
         end
      end

      if to_comp=0 then
      do
         call lineout comppath,"<testname id='once"||testname_onceindex"' -
                                value='not'>"a2check"</testname>"
         testname_onceindex=testname_onceindex+1
      end
   end

call lineout comppath,"</log2>"
```

```
qte='"'

call lineout comppath,"</testunit>" --write closing tag

rc=stream(comppath,"C","close") --close new generated comp-file completely
delpath=qte||comppath||qte
address cmd 'iexplore.exe' delpath --open new generated comp-file

end

rc=stream(checkpath,"C","close") --close comptests.txt(info -log1 -log2)
call sysfiledelete checkpath --delete the file

::requires 'xmlparser.cls'


/* This class provides the information of the log files by reading the tags via the
"xmlparser.cls" */
::class myparser subclass xmlparser

::method mystream attribute
::method curtag attribute

::method start_element
  use arg chunk
  self~curtag=chunk~tag
  a=.array~of()
  sub=chunk~tag

     if sub="subtest" then  -- if xml-Tag = subtest
     do
        att=chunk~attr  -- get the attributes (returns one string)
        attarr=att~makearray  --strip down to attributes value and name
        check=attarr~at(1)
        fpos=pos("'",check)
        spos=pos("'",check, fpos+1)
        value=substr(check, fpos+1, (spos-1)-fpos)
        tpos=pos("'",check, spos+1)
        opos=pos("'",check, tpos+1)
        name=substr(check, tpos+1, (opos-1)-tpos)

        .insert.array~put(name,.insert.index) --insert name and value
        .local~insert.index=.insert.index+1
        .insert.array~put(value,.insert.index)
        .local~insert.index=.insert.index+1
   end

::method text
  use arg chunk
     if self~curtag="info" then      --if xml-tag = info
     do      --insert "info"-text
        .insert.array~put(chunk~text,.insert.index)
        .local~insert.index=.insert.index+1
     end

::method end_element
  use arg chunk
  main_index=1
     if chunk~tag="subtest" then --if closing-tag = subtest
     do
        .comp.log~put(.insert.array,.index.TeRA) --insert array into local array
        .local~index.TeRA=.index.TeRA+1
        .local~insert.array=.array~new
        .local~insert.index=1
     end
```

7.8.1.8 LOG_XSL_SCRIPT.REX

Contains the routines for the log file user interface.

```
/*
   name:             log_xsl_script.rex
   author:           Rainer Kegel
   date:             2007-01-02

   purpose:          part of the Test Runner Application for ooRexxUnit
                     - provides routines for log.xsl

   license:          CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/



  ::routine okay public
/* routine to hide or show the tables with the class "okay" in the log files */
  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".okay" then
    do
    len=i
    check1=document~stylesheets[0]~rules[i]~style~display
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonokay")~value="show okay"
      end
      if check1="none" then
      do
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonokay")~value="hide okay"
      end
    end
  end

  ::routine failure public
```

```
/* routine to hide or show the tables with the class "failure" in the log files */
  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".failure" then
    do
    len=i
    check1=document~stylesheets[0]~rules[i]~style~display
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonfailure")~value="show failure"
      end
      if check1="none" then
      do
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonfailure")~value="hide failure"
      end
    end
  end

  ::routine error public
/* routine to hide or show the tables with the class "error" in the log files */
  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".error" then
    do
    len=i
    check1=document~stylesheets[0]~rules[i]~style~display
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonerror")~value="show error"
      end
      if check1="none" then
      do
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonerror")~value="hide error"
      end
    end
  end
```

7.8.1.9 COMPLOG_XSL_SCRIPT.REX

Contains the routines for the compared log files user interface.

```
/*
   name:            complog_xsl_script.rex
   author:          Rainer Kegel
   date:            2007-01-02

   purpose:         part of the Test Runner Application for ooRexxUnit
                    - provides routines for complog.xsl

   license:         CPL 1.0 (Common Public License v1.0, see below)

*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
```

```
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/



  ::routine buttons public
  /* to display and hide buttons */
  checkboth=document~getelementbyid("both1")
  checkonce=document~getelementbyid("once1")
    if checkboth=.nil then
    document~getelementbyid("buttonrunboth")~style~display="none"
    if checkonce=.nil then
    document~getelementbyid("buttonrunonce")~style~display="none"

  ::routine changes public
  /* to display and hide the section "Changes" */
  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".changes" then
    do
    len=i
    check1=document~stylesheets[0]~rules[i]~style~display
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonchanges")~value="show changes"
      end
      if check1="none" then
      do
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonchanges")~value="hide changes"
      end
    end
  end

  ::routine log1 public
  /* to display and hide the section "log1" */
  r_both=0
  r_once=0
  checkboth=document~getelementbyid("both1")
  checkonce=document~getelementbyid("once1")
    if checkboth\=.nil then
    r_both=1
    if checkonce\=.nil then
    r_once=1

  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".log1" then
    do
    check1=document~stylesheets[0]~rules[i]~style~display
    len=i
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonlog1")~value="show log1"
      end
      if check1="none" then
      do
```

```
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonlog1")~value="hide log1"
      end
    lenb=document~stylesheets[0]~rules~length
    do x=0 to lenb-1
    checkb=document~stylesheets[0]~rules[x]~selectortext
      if checkb=".log2" then
      do
      log2check=document~stylesheets[0]~rules[x]~style~display
      lenb=x
      checkdisplay=document~stylesheets[0]~rules[i]~style~display
        if (checkdisplay="none") then
          if (log2check="none") then
          do
          document~getelementbyid("buttonrunboth")~style~display="none"
          document~getelementbyid("buttonrunonce")~style~display="none"
          end
          if checkdisplay="inline" then
            if r_both=1 then
            document~getelementbyid("buttonrunboth")~style~display="inline"

          if checkdisplay="inline" then
            if r_once=1 then
            document~getelementbyid("buttonrunonce")~style~display="inline"

          if log2check="inline" then
            if r_both=1 then
            document~getelementbyid("buttonrunboth")~style~display="inline"

          if log2check="inline" then
            if r_once=1 then
            document~getelementbyid("buttonrunonce")~style~display="inline"
      end
    end
    end
  end

  ::routine log2 public
  /* to display and hide the section "log2" */
  r_both=0
  r_once=0
  checkboth=document~getelementbyid("both1")
  checkonce=document~getelementbyid("once1")
    if checkboth\=.nil then
    r_both=1
    if checkonce\=.nil then
    r_once=1

  len=document~stylesheets[0]~rules~length
  do i=0 to len-1
  check=document~stylesheets[0]~rules[i]~selectortext
    if check=".log2" then
    do
    check1=document~stylesheets[0]~rules[i]~style~display
    len=i
      if check1="inline" then
      do
      document~stylesheets[0]~rules[i]~style~display="none"
      document~getelementbyid("buttonlog2")~value="show log2"
      end
      if check1="none" then
      do
      document~stylesheets[0]~rules[i]~style~display="inline"
      document~getelementbyid("buttonlog2")~value="hide log2"
      end
    lenb=document~stylesheets[0]~rules~length
    do x=0 to lenb-1
    checkb=document~stylesheets[0]~rules[x]~selectortext
      if checkb=".log1" then
      do
      log1check=document~stylesheets[0]~rules[x]~style~display
      lenb=x
      checkdisplay=document~stylesheets[0]~rules[i]~style~display
        if (checkdisplay="none") then
          if (log1check="none") then
          do
          document~getelementbyid("buttonrunboth")~style~display="none"
          document~getelementbyid("buttonrunonce")~style~display="none"
```

```
        end
        if checkdisplay="inline" then
          if r_both=1 then
          document~getelementbyid("buttonrunboth")~style~display="inline"
        if checkdisplay="inline" then
          if r_once=1 then
          document~getelementbyid("buttonrunonce")~style~display="inline"
        if  log1check="inline" then
          if r_both=1 then
          document~getelementbyid("buttonrunboth")~style~display="inline"
        if  log1check="inline" then
          if r_once=1 then
          document~getelementbyid("buttonrunonce")~style~display="inline"
    end
  end
  end
end

::routine runboth public
/* to display and hide the button "runboth" */
len=document~stylesheets[0]~rules~length
do i=0 to len-1
check=document~stylesheets[0]~rules[i]~selectortext
  if check=".runboth" then
  do
  len=i
  check1=document~stylesheets[0]~rules[i]~style~display
    if check1="inline" then
    do
    document~stylesheets[0]~rules[i]~style~display="none"
    document~getelementbyid("buttonrunboth")~value="show runboth"
    end
    if check1="none" then
    do
    document~stylesheets[0]~rules[i]~style~display="inline"
    document~getelementbyid("buttonrunboth")~value="hide runboth"
    end
  end
end

::routine runonce public
/* to display and hide the button "runonce" */
len=document~stylesheets[0]~rules~length
do i=0 to len-1
check=document~stylesheets[0]~rules[i]~selectortext
  if check=".runonce" then
  do
  len=i
  check1=document~stylesheets[0]~rules[i]~style~display
    if check1="inline" then
    do
    document~stylesheets[0]~rules[i]~style~display="none"
    document~getelementbyid("buttonrunonce")~value="show runonce"
    end
    if check1="none" then
    do
    document~stylesheets[0]~rules[i]~style~display="inline"
    document~getelementbyid("buttonrunonce")~value="hide runonce"
    end
  end
end
```

7.8.2 Cascading Style Sheets

The following chapters deal with the Cascading Style Sheets of TeRA.

7.8.2.1 STYLES.CSS

STYLES.CSS is the style sheet for TeRA.HTA and TeRA-COMPARELOGS.HTA.

```
/*  name:        styles.css                                                 */
/*  author:      Rainer Kegel                                               */
/*  date:        2007-01-02                                                 */
/*                                                                          */
/*  purpose:     part of the Test Runner Application for ooRexxUnit         */
/*               - provides general structures                              */

/*  license:     CPL 1.0 (Common Public License v1.0, see below)            */


/* ------------------------------------------------------------------------ */
/*                                                                          */
/*  Copyright (c) 2007 Rainer Kegel. All rights reserved.                   */
/*                                                                          */
/*  This program and the accompanying materials are made available under    */
/*  the terms of the Common Public License v1.0 which accompanies this      */
/*  distribution. A copy is also available at the following address:        */
/*  http://www.opensource.org/licenses/cpl1.0.php                           */
/*                                                                          */
/*  Redistribution and use in source and binary forms, with or              */
/*  without modification, are permitted provided that the following         */
/*  conditions are met:                                                     */
/*                                                                          */
/*  Redistributions of source code must retain the above copyright          */
/*  notice, this list of conditions and the following disclaimer.           */
/*  Redistributions in binary form must reproduce the above copyright       */
/*  notice, this list of conditions and the following disclaimer in         */
/*  the documentation and/or other materials provided with the distribution.*/
/*                                                                          */
/*  Neither the name of Rexx Language Association nor the names             */
/*  of its contributors may be used to endorse or promote products          */
/*  derived from this software without specific prior written permission.   */
/*                                                                          */
/*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS     */
/*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT       */
/*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS       */
/*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT*/
/*  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,   */
/*  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED*/
/*  TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,     */
/*  OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY  */
/*  OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING */
/*  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS      */
/*  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.            */
/*                                                                          */
/* ------------------------------------------------------------------------ */


/* styling rules for elements “a” and “span” */
a, span
{
        font-family:Arial,sans-serif;
}

dt
{
        border-width: 1px;
        border-style: solid;
        border-color: #dddddd;
        background-color:#BDC6DE;
        padding: 5px;
        font-family:Arial,sans-serif;
}
```

```
h1
{
        color: #616161;
        text-align:center;
}

input
{
        font-family:Arial,sans-serif;
}

#acl table
{
        display: none;
}

table.own td
{
        background-color: #ddd;
        font-family:Arial,sans-serif;
}

table.inherit td
{
        background-color: #eee;
        font-family:Arial,sans-serif;
}

table
{
        border-collapse: collapse;
        font-family:Arial,sans-serif;
}

th, td
{

        padding: 5px;
        width: 500px;
        font-family:Arial,sans-serif;
}

#logo
{
        position:absolute;
        top:10px;
        left:10px;
}


#TUL1
{
        position:absolute;
        top:150px;
        left:10px;
        width:98%;
}

#invert, #checkAll, #uncheckAll, #runtests, #showresult, #comptests, #openAll,
#closeAll, #showLog, #uncheck, #compareLogs, #arrange
{
        width:100px;
}

#username,
{
        width:39%;
        font-family:Arial,sans-serif;
}

#comment
{
        width:60%;
        font-family:Arial,sans-serif;
}

body
{
        margin:0;
        padding:0;
        background-color:#e0e0e0;
```

```
}

#content_container
{
        padding-bottom:5em;
}

#footer1
{
        position:fixed;
        border-top-width:0.05em;
        border-top-color:#404040;
        border-top-style:solid;
        bottom:1px;
        background: #707070;
        text-align:left;
        padding:5px;
        width:100%;
}

#footer2
{
        position:fixed;
        bottom:1px;
        background:#505050;
        text-align:left;
        padding:5px;
        width:100%;
}

/* for internet explorer */

* html, * html body
{
        margin:0;
        padding:0;
        height:100%;
        overflow:hidden;
}

* html #content_container
{
        padding:0;
        height:88%;
        overflow:auto;
}

* html #footer1
{
        background:#c1c1c1;
        border-top-width:0.05em;
        border-top-color:#404040;
        border-top-style:solid;
        height:6%;
        text-align:left;
        padding-top:1%;
        width:100%;
}

* html #footer2
{
        background:#b9b9b9;
        height:6%;
        text-align:left;
        padding-top:1%;
        padding-bottom:1%;
}

.but1
{
        position:absolute;
        left:82%;
        width:6em;
}

.but2
{
        position:absolute;
        left:91%;
        width:6em;
```

```
}

.link_tu
{
        color:black;
        text-decoration:none;
}
```

7.8.2.2 LOG.CSS

LOG.CSS contains the styling rules for the log files.

```
/*  name:        log.css                                                   */
/*  author:      Rainer Kegel                                              */
/*  date:        2007-01-02                                                */

/*  purpose:     part of the Test Runner Application for ooRexxUnit
                      - provides the structures for log files              */

/*  license:     CPL 1.0 (Common Public License v1.0, see below)           */


/* ---------------------------------------------------------------------- */
/*                                                                        */
/*  Copyright (c) 2007 Rainer Kegel. All rights reserved.                 */
/*                                                                        */
/*  This program and the accompanying materials are made available under  */
/*  the terms of the Common Public License v1.0 which accompanies this    */
/*  distribution. A copy is also available at the following address:      */
/*  http://www.opensource.org/licenses/cpl1.0.php                         */
/*                                                                        */
/*  Redistribution and use in source and binary forms, with or            */
/*  without modification, are permitted provided that the following       */
/*  conditions are met:                                                   */
/*                                                                        */
/*  Redistributions of source code must retain the above copyright        */
/*  notice, this list of conditions and the following disclaimer.         */
/*  Redistributions in binary form must reproduce the above copyright     */
/*  notice, this list of conditions and the following disclaimer in       */
/*  the documentation and/or other materials provided with the distribution.  */
/*                                                                        */
/*  Neither the name of Rexx Language Association nor the names           */
/*  of its contributors may be used to endorse or promote products        */
/*  derived from this software without specific prior written permission. */
/*                                                                        */
/*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS   */
/*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT     */
/*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS     */
/*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  */
/*  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,     */
/*  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED  */
/*  TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,       */
/*  OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY    */
/*  OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING   */
/*  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS        */
/*  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.          */
/*                                                                        */
/* ---------------------------------------------------------------------- */


/* styling rule for class "okay" in log files */
.okay
{
        display:inline;
        background:#e0e0e0;
}

/* styling rule for class "failure" in log files */
.failure
{
        display:inline;
        background:#FFFF6B;
}

/* styling rule for class "error" in log files */
.error
{
```

```
        display:inline;
        background:#FF9473;
}

/* styling rule for class "blank" in log files */
.blank
{
        display:inline;
}

/* styling rule for element "th" in log files */
th
{
        width:15%;
}

/* styling rule for class "testsuite" in log files */
.testsuite
{
        width:100%;
        background:yellow;
}

.testcase
{
        width:100%;
        background:#BDC6DE;
}

.space
{
        height:20px;
}

.var1
{
        font-size:13px;
        font-family:arial,sans-serif;
        background:#e0e0e0;
        margin:0;
        padding:0;
        height:100%;
        overflow:hidden;
}

.var2
{
        padding:0;
        height:94%;
        overflow:auto;
}

.var3
{
        background:#b9b9b9;
        height:6%;
        text-align:left;
        padding-top:1%;
        padding-bottom:1%;
}

.var4
{
        font-weight:bold;
        font-size:13px;
        text-align:left;
        width:20%;
}

.var5
{
        background:white;
        text-align:left;
        border-width:1px;
        padding:1px;
        border-style:inset;
        border-color:#a1a1a1;
}

.var6
{
```

```
        background:white;
        text-align:left;
        border-width:1px;
        padding:1px;
        border-style:inset;
        border-color:#a1a1a1;
}

.var7
{
        text-align:left;
        font-weight:bold;
        font-size:12px;
        width:30%;
}

.var8
{
        text-align:left;
        font-size:12px;
}

.var9
{
        text-align:left;
        font-size:13px;
        width:70px;
}

.var10
{
        text-align:left;
        width:7%;
}

.var11
{
        text-align:left;
        width:10%;
}

.var12
{
        text-align:left;
        font-weight:bold;
}

.var13
{
        text-align:left;
        font-size:10px;
}

.var14
{
        font-size:11px;
        font-family:Verdana, Courier;
}

.var15
{
        text-align:left;
        font-size:13px;
        width:10%;
}

.var16
{
        text-align:left;
        font-size:13px;
        width:50px;
}

.var17
{
        text-align:left;
        font-size:13px;
        width:12%;
}

.var18
```

```
{
        text-align:left;
        font-size:13px;
        width:10px;
}
```

7.8.2.3 COMPLOG.CSS

COMPLOG.CSS contains the styling rules for the compared log files.

```
/*  name:        complog.css                                                */
/*  author:      Rainer Kegel                                               */
/*  date:        2007-01-02                                                 */

/*  purpose:     part of the Test Runner Application for ooRexxUnit         */
/*               - provides general structures                              */

/*  license:     CPL 1.0 (Common Public License v1.0, see below)            */


/* -------------------------------------------------------------------------- */
/*                                                                            */
/*  Copyright (c) 2007 Rainer Kegel. All rights reserved.                     */
/*                                                                            */
/*  This program and the accompanying materials are made available under      */
/*  the terms of the Common Public License v1.0 which accompanies this        */
/*  distribution. A copy is also available at the following address:          */
/*  http://www.opensource.org/licenses/cpl1.0.php                             */
/*                                                                            */
/*  Redistribution and use in source and binary forms, with or                */
/*  without modification, are permitted provided that the following           */
/*  conditions are met:                                                       */
/*                                                                            */
/*  Redistributions of source code must retain the above copyright            */
/*  notice, this list of conditions and the following disclaimer.             */
/*  Redistributions in binary form must reproduce the above copyright         */
/*  notice, this list of conditions and the following disclaimer in           */
/*  the documentation and/or other materials provided with the distribution.  */
/*                                                                            */
/*  Neither the name of Rexx Language Association nor the names               */
/*  of its contributors may be used to endorse or promote products            */
/*  derived from this software without specific prior written permission.     */
/*                                                                            */
/*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS       */
/*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT         */
/*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS         */
/*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  */
/*  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,     */
/*  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED  */
/*  TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,       */
/*  OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY    */
/*  OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING   */
/*  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS        */
/*  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.              */
/*                                                                            */
/* -------------------------------------------------------------------------- */


/* styling rules for the class 'changes' in the compared log files */
.changes
{
        display:inline;
        border:1px;
        border-style:solid;
        border-color:black;
}

/* styling rules for the class 'log1' in the compared log files */
.log1
{
        display:inline;
        width:1000px;
        border:1px;
        border-style:solid;
        border-color:black;
```

```
}

/* styling rules for the class 'log2' in the compared log files */
.log2
{
        display:inline;
        width:1000px;
        border:1px;
        border-style:solid;
        border-color:black;
}

/* styling rules for the class 'runboth' in the compared log files */
.runboth
{
        display:inline;
        background:#E0E0E0;
}

/* styling rules for the class 'runonce' in the compared log files */
.runonce
{
        display:inline;
        background:#C6B5DE;
}

/* styling rules for the element 'body' in the compared log files */
body
{
        font-family:arial, sans-serif;
        background:#e0e0e0;
        margin:0;
        padding:0;
        height:100%;
        overflow:hidden;
}

/* styling rules for the id 'content_comtainer' in the compared log files */
#content_container
{
        padding-bottom:5em;
}

/* styling rules for the id 'footer' in the compared log files */
#footer
{
        position:fixed;
        border-top-width:0.05em;
        border-top-color:#404040;
        border-top-style:solid;
        bottom:1px;
        background: #707070;
        text-align:left;
        padding:5px;
        width:100%;
}

/* styling rules for the id 'footer' in the compared log files */
* html #content_container
{
        padding:0;
        height:94%;
        overflow:auto;
}

/* styling rules for the id 'footer' in the compared log files */
* html #footer
{
        background:#c1c1c1;
        border-top-width:0.05em;
        border-top-color:#404040;
        border-top-style:solid;
        height:6%;
        text-align:left;
        padding-top:1%;
        width:100%;
}

/* styling rules for the ids in the compared log files */
#buttonchanges, #buttonlog1, #buttonlog2, #buttonrunboth, #buttonrunonce,
{
```

```
        width:100px;
}

/* styling rules for the ids 'leg2' and 'leg1' in the compared log files */
#leg1, #leg2
{
        width:150px;
        height:23px;
        text-align:center;
        border-style:solid;
        border-color:black;
        border-width:0.05em;
}

.var1
{
        text-align:left;
        background:#BDC6DE;
        width:1000px;
}

.var2
{
        text-align:left;
        font-weight:bold;
        width:50%;
}

.var3
{
        valign:top;
        width:50%;
}

.yellow
{
        background:#FFFF6B;
        font-size:12px;
}

.width
{
        width:23%;
}
```

7.8.3 CLASSES

The following chapters deal with the two classes, which are used by TeRA.

7.8.3.1 ooRexxUnit.CLS

ooRexxUnit.CLS is the class, which contains all assertion methods, creates `TestCases` and `TestSuites` and `TestResults`.[1]

```
#!/usr/bin/rexx
/*
/* this is a ooRexx line comment, will get ignored by pre-processor which analyzes this
header */
/* enclosed between the very first block comment, i.e. everything between the first "/*"
.. "*/" */
/*
   name:             ooRexxUnit.cls
   author:           Rony G. Flatscher, Rick McGuire
   date:             2005-08-07
   version:          1.0.1
   changed1:         2005-08-07, --rgf, moved assertion routines to a class "Assert",
                                 --which serves as a superclass for TestCase: this is
                                 --the junit-approach
                     2005-08-20, --rgf, added license, assertCount, corrected some
                                 --little bugs
                     2005-08-21, --rgf, added public routine
                                 --"makeTestSuiteFromFileList()"
                     2005-08-30, --rgf, changed some comments from "ooRexx" to
                                 --"ooRexxUnit"
                     2005-10-09, --rgf, only count assertions, if successful
                     2005-10-14, --rgf, changed failure-message string slightly to
                                 --improve understandibility
                     2005-10-27, --rgf, added ability to report which assertion failed
                                 --for what reason (suggested by Walter Pachl)
                     2005-10-28, --rgf, added @assertFailure attribute to lead-in the
                                 --report on the assertion failure, removed angle-
                                 --brackets from that added text
                     2006-03-25, --rgf, removed 'private' attribute from TestResult's
                                 --ooRexx only attributes
                                 --“logQueue" and "TestCaseTable"
                     2006-04-11, Rick McGuire, added condition-handling to Assert and
TestCase.run()
                     2006-05-17, --rgf, altered Rick's code slightly,
                                 --expectSyntax(errorCode) and
                                 --expectCondition(conditionName)
                     2006-10-17, --rgf, escape non-printable chars in error/failure
                                 --string information to
                                 --a Rexx style string literal (concatenating
                                 --hexadecimal strings);
                                 --use new syntax error # 93.964 (application error)
                                 --instead of
                                 --error # 40.1, which had to be used prior to ooRexx
                                 --3.1.1
                     2006-11-05, --rgf, - moved initialisation of
                                 --"defaultTestResultClass" to the class'
                                 --constructor; *but* this also needs the definition of
                                 --the "TestResult"
                                 --class to be moved physically before the "TestCase"
                                 --class (otherwise the class is not known yet and the
                                 --string ".TESTREUSLT" is stored
                                 --instead of the class object!!);
                                 --error message encodings now always add a colon to the
                                 --word "ERROR"; the
                                 --the error message after the eye catcher string "--->"
                                 --is not enquoted
                                 --in square brackets anymore
```

[1] Source: http://oorexx.cvs.sourceforge.net/oorexx/oorexxunit/OOREXXUNIT.CLS?view=log.

```
                                --failure message encodings, if given, now use the
                                --string "--->"
                                --as an eye catcher for parsing and is not enquoted in
                                --square brackets anymore
                    2006-11-27, --rgf, - changed Assert[Not]Same to show ObjectID in
                                --failure message to ease comparison
                    2006-11-28, --rgf, - corrected logic to intercept any condition
                                --from running a test case
                    2006-12-13, --rgf, - changed wording from "expected condition...was
                                --not received" to
                                --“expected condition...was not raised"
                                --made sure that "makeTestSuiteFromFileList()" will
                                --create an own test suite per
                                --test class for which mandatory test methods got
                                --listed
                    2006-12-14, --rgf, added hashbang line
                    2006-12-26/27/28, --rgf, - removed "bWalterPachl" flag (to show
                                      --reason of failure) as
                                      --reason of failures are always shown; even if no
                                      --failure message
                                      --is supplied the assertion methods will still
                                      --supply the "@assertionFailure"
                                      --string to indicate expected and (not matching)
                                      --received values
                                      --enhanced "assertEquals" to work on ordered and
                                      --unordered collections:
                                      --unordered collections (of any type) are
                                      --regarded to be equal, if both contain
                                      --the same number of the same index/value pairs;
                                      --ordered collections (of any type) are regarded
                                      --to be equal, if the items
                                      --in the MAKEARRAY object are the same in the
                                      --same order
                    2006-12-30, --rgf, added method assertNotEquals to Assert class
                                --(ooRexxUnit only, makes it easier
                                --to test for unequal collection values)

   language-level:   6.0
   needs:            ooRexx 3.1 or later (introduced syntax error # 93.964, which is
exploited)
-- determines the minimum ooRexx language level (6.00 = ooRexx, IBM Object REXX)

   purpose:          Supply the base classes for a JUnit compliant testing framework for
ooRexx

   remark:           Wherever possible the JUnit class and method names are used to help
ease
                     the understanding.

   license:          CPL 1.0 (Common Public License v1.0, see below)

   link:             http://www.junit.org
                     http://junit.sourceforge.net/doc/cookbook/cookbook.htm
                     http://junit.sourceforge.net/doc/cookstour/cookstour.htm


-- there may be any number of subcategories, most important listed first, second
important second, ...
-- no need to append numbers, but may be easier to realize the category level easily

-- this is the main categorization
   category0:        ooRexxUnit

-- this is the next concrete categorization
   category1:        framework
*/

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2005-2006 Rexx Language Association. All rights reserved.    */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.opensource.org/licenses/cpl1.0.php                              */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
```

```
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/

.local~chars.NonPrintable=xrange("00"x, "1F"x) || "FF"x  -- define non-printable chars



/* ************************************************************************** */
/* ************************************************************************** */
::class "TestResult" public
::method init
  expose fErrors fFailures fRunTests fStop fTestRuns TestCaseTable logQueue fAssertions
      -- initialize object variables
  fErrors=.queue~new
  fFailures=.queue~new
  logQueue=.queue~new  -- ooRexxUnit only, logs all

  fAssertions=0
  fRunTests=0
  fStop=.false

  TestCaseTable=.table~new

-- ::method TestCaseTable attribute private
::method logQueue attribute              -- ooRexxUnit only
::method TestCaseTable attribute         -- ooRexxUnit only

::method addError       /* ooRexxUnit only, allows to be overriden/intercepted by
subclasses */
  expose fErrors TestCaseTable logQueue
  use arg aTestCase, co

  fErrors~queue(co)     /* enqueue the condition object; cf. entry
"OOREXXUNIT.CONDITION" */
  logQueue~queue(co)
  TestCaseTable[aTestCase]~queue("   " co~ooRexxUnit.Condition) -- "--->" pp(msg))


::method addFailure     /* ooRexxUnit only, allows to be overriden/intercepted by
subclasses */
  expose fFailures TestCaseTable logQueue
  use arg aTestCase, co

  fFailures~queue(co)   /* enqueue the condition object; cf. entry
"OOREXXUNIT.CONDITION" */
  logQueue~queue(co)
  TestCaseTable[aTestCase]~queue("   " co~ooRexxUnit.Condition)

::method assertCount    -- ooRexxUnit only
  expose fAssertions
  return fAssertions

::method endTest        -- informs that the supplied test was completed
  expose TestCaseTable fStop logQueue fAssertions
  use arg aTestCase

  dateTime=pp(date("s") time("L"))
  TestCaseTable[aTestCase]~queue(dateTime": endTest")
  dir=.directory~new ~~setentry("OOREXXUNIT.CONDITION", dateTime":" pp("endTest") -
      makeTestCaseString(aTestCase) )
  logQueue~queue(dir)
  fAssertions=fAssertions+aTestCase~assertCount
  fStop=.false          -- reset indicator
```

```
::method errorCount     -- return # of errors
  expose fErrors
  return fErrors~items

::method errors         -- return error queue
  expose fErrors
  return fErrors


::method failureCount   -- return # of failures
  expose fFailures
  return fFailures~items

::method failures       -- return failure queue
  expose fFailures
  return fFailures


::method run            -- convenience method to run given TestCase
  use arg aTestCase
  return aTestCase~run(self)

::method runCount       -- gets the number of run tests
  expose fRunTests
  return fRunTests

::method shouldStop     -- return value
  expose fStop
  return fStop

::method startTest
  expose TestCaseTable fStop fRunTests logQueue
  use arg aTestCase

  if TestCaseTable~hasindex(aTestCase)=.false then -- already a queue created for it?
     TestCaseTable[aTestCase]=.queue~new

  dateTime=pp(date("s") time("L"))
  TestCaseTable[aTestCase]~queue(dateTime": startTest")
  dir=.directory~new ~~setentry("OOREXXUNIT.CONDITION", dateTime":" pp("startTest") -
      makeTestCaseString(aTestCase))
  logQueue~queue(dir)

  fStop=.false          -- reset indicator
  fRunTests=fRunTests+1 -- increase run counter

::method stop           --  mark that the test run should stop
  expose fStop
  fStop=.true

::method wasSuccessful  -- returns whether the entire test was successful or not
  expose fErrors fFailures

  return (fErrors~items+fFailures~items)=0



/* ******************************************************************************
*/
/* ******************************************************************************
*/
::class "Assert" public

::method version attribute class

:: method init class
  expose version
  version=101.20070112

::method init
  expose fAssertions          -- count assertions, ooRexxUnit only
  fAssertions=0

::method assertCount          -- ooRex only
  expose fAssertions          -- count assertions, ooRexxUnit only
  return fAssertions

::method fAssertions attribute private    -- ooRexxUnit only
```

```
   -- assertions will raise a user error, if they do not hold
::method assertEquals
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=2 then             -- no failure message supplied
  do
     bEquals=(arg(1)=arg(2))
         -- values are not equal, but both are collections, test for equality
     if \bEquals then
     do
         bEquals=isCollEqual(arg(1), arg(2))
     end

     if bEquals then
     do
        fAssertions=fAssertions+1
        return -- assertion holds
     end
     self~fail("@assertFailure assertEquals: expected="formatObjectInfo(arg(1))", -
     actual="formatObjectInfo(arg(2))"."||"09"x)
  end

  bEquals=(arg(2)=arg(3))
         -- values are not equal, but both are collections, test for equality
  if \bEquals then
  do
      bEquals=isCollEqual(arg(2), arg(3))
  end

  if bEquals then
  do
     fAssertions=fAssertions+1
     return    -- assertion holds
  end

  sTmp="@assertFailure assertEquals: expected="formatObjectInfo(arg(2))", -
        actual="formatObjectInfo(arg(3))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg



::method assertFalse
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=1 then             -- no failure message supplied
  do
     if arg(1)=.false then
     do
        fAssertions=fAssertions+1
        return   -- assertion holds
     end
     self~fail("@assertFailure assertFalse: expected=[0], actual="ppp(arg(1))"."||"09"x)
  end

  if arg(2)=.false then
  do
     fAssertions=fAssertions+1
     return      -- assertion holds
  end

  sTmp="@assertFailure assertFalse: expected=[0], actual="ppp(arg(2))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg


   -- assertions will raise a user error, if they do not hold
::method assertNotEquals      -- ooRexxUnit only, 2006-12-30
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=2 then             -- no failure message supplied
  do
     bEquals=(arg(1)=arg(2))
         -- values are not equal, but both are collections, test for equality
     if \bEquals then
     do
         bEquals=isCollEqual(arg(1), arg(2))
     end

     if \bEquals then
     do
        fAssertions=fAssertions+1
```

```
        return -- assertion holds
     end
     self~fail("@assertFailure assertNotEquals: expected="formatObjectInfo(arg(1), "\= -
     ")", actual="formatObjectInfo(arg(2))"."||"09"x)
  end

  bEquals=(arg(2)=arg(3))
          -- values are not equal, but both are collections, test for equality
  if \bEquals then
  do
      bEquals=isCollEqual(arg(2), arg(3))
  end

  if \bEquals then
  do
     fAssertions=fAssertions+1
     return    -- assertion holds
  end

  sTmp="@assertFailure assertNotEquals: expected=["formatObjectInfo(arg(2), "\= ")", -
  actual="formatObjectInfo(arg(3))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg



::method assertNotNull
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=1 then             -- no failure message supplied
  do
     if .nil<>arg(1) then
     do
        fAssertions=fAssertions+1
        return    -- assertion holds
     end
     self~fail("@assertFailure assertNotNull: expected=[\= [.nil]],-
     actual=[.nil]."||"09"x)
  end

  if .nil<>arg(2) then
  do
     fAssertions=fAssertions+1
     return       -- assertion holds
  end

  sTmp="@assertFailure assertNotNull: expected=[\= [.nil]], actual=[.nil]."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg


::method assertNotSame
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=2 then             -- no failure message supplied
  do
     if (arg(1)==arg(2))=.false then
     do
        fAssertions=fAssertions+1
        return  -- assertion holds
     end

     self~fail("@assertFailure assertNotSame: expected="formatObjectInfo(arg(1), "\== -
     ")", actual="formatObjectInfo(arg(2))"."||"09"x)

     /* self~fail("@assertFailure assertNotSame: not expected="ppp(arg(1)~string
"<hashValue:" getEscapedhashValue(arg(1))">")", received="ppp(arg(2)~string
"<hashValue:" getEscapedhashValue(arg(2))">")"."||"09"x) */
  end

  if (arg(2)==arg(3))=.false then
  do
     fAssertions=fAssertions+1
     return     -- assertion holds
  end

  /* sTmp="@assertFailure assertNotSame: not expected="ppp(arg(2)~string "<hashValue:"
getEscapedhashValue(arg(2))">")", received="ppp(arg(3)~string "<hashValue:"
getEscapedhashValue(arg(3))">")"."||"09"x */

  sTmp="@assertFailure assertNotSame: expected="formatObjectInfo(arg(2), "\== ")", -
       actual="formatObjectInfo(arg(3))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg
```

```
::method assertNull
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=1 then             -- no failure message supplied
  do
     if .nil=arg(1) then
     do
        fAssertions=fAssertions+1
        return     -- assertion holds
     end
     self~fail("@assertFailure assertNull: expected=[.nil], -
                actual="ppp(arg(1))"."||"09"x)
  end

  if .nil=arg(2)then
  do
     fAssertions=fAssertions+1
     return         -- assertion holds
  end

  sTmp="@assertFailure assertNull: expected=[.nil], actual="ppp(arg(2))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg


::method assertSame
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=2 then             -- no failure message supplied
  do
     if (arg(1)==arg(2))=.true then
     do
        fAssertions=fAssertions+1
        return-- assertion holds
     end
     -- self~fail("@assertFailure assertSame: expected="ppp(arg(1)~string "<hashValue:"
getEscapedhashValue(arg(1))">")", received="ppp(arg(2)~string "<hashValue:"
getEscapedhashValue(arg(2))">")"."||"09"x)
     self~fail("@assertFailure assertSame: expected="formatObjectInfo(arg(1))", -
     actual="formatObjectInfo(arg(2))"."||"09"x)
  end

  if (arg(2)==arg(3))=.true then
  do
     fAssertions=fAssertions+1
     return   -- assertion holds
  end

  sTmp="@assertFailure assertSame: expected="formatObjectInfo(arg(2))", -
       actual="formatObjectInfo(arg(3))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg


::method assertTrue
  expose fAssertions          -- count assertions, ooRexxUnit only
  if arg()=1 then             -- no failure message supplied
  do
     if arg(1)=.true then
     do
        fAssertions=fAssertions+1
        return    -- assertion holds
     end
     self~fail("@assertFailure assertTrue: expected=[1], actual="ppp(arg(1))"."||"09"x)
  end

  if arg(2)=.true then
  do
     fAssertions=fAssertions+1
     return       -- assertion holds
  end

  sTmp="@assertFailure assertTrue: expected=[1], actual="ppp(arg(2))"."||"09"x
  self~fail(sTmp || arg(1))      -- fail with msg


::method fail
       -- ooRexx: need to raise a syntax error, because USER exception needs to be
propageted
  if arg()=0 then msg=""
             else msg=arg(1)

   -- use application definable syntax error (since ooRexx 3.1), supply description for
```

```
   -- test methods to be able to find out that the ooRexxUnit framework raised it
  RAISE syntax 93.964 array (msg) description ("ooRexxUnit.cls - source of syntax -
  exception 'FAIL' method invocation in class 'ASSERT'.")


::method conditionExpected attribute

::method clearCondition
  expose conditionExpected
  conditionExpected = .false

::method expectSyntax
  expose conditionExpected conditionName errorCode
  parse arg errorCode       -- retrieve errorCode
  conditionName="SYNTAX"
  conditionExpected=.true

  -- self~expectCondition( "SYNTAX", arg(1) )

::method expectCondition
  expose conditionExpected conditionName
  use arg conditionName           /* only name of condition is expected, can be two
words, e.g. "USER SOMETHING" */
  conditionExpected = .true


::method checkCondition
  expose conditionExpected conditionName errorCode fAssertions
  use arg receivedCondition

  if receivedCondition~condition == conditionName then
  do
      if conditionName == "SYNTAX" then
      do
          if errorCode <> receivedCondition~code then
          do
              return .false
          end
      end
      fAssertions=fAssertions+1        /* asserted that expected conditionName has
occurred! */
      return .true
  end
  return .false


   /* Method that will fail, if a condition was expected, but had not been raised. */
::method check4ConditionFailure
  expose conditionExpected conditionName errorCode

  if conditionExpected=.true then
  do
     if conditionName="SYNTAX" then
        tmpStr=conditionName errorCode    /* supply the expected errorCode with the
expected exception */
     else
        tmpStr=conditionName

     sTmp="@assertFailure check4ConditionFailure: expected condition" -
          pp(tmpStr) "was not raised."||"09"x

     self~fail(sTmp)               -- fail with msg
  end


/* ******************************************************************************
*/
/* ******************************************************************************
*/
::class "TestCase" subclass "Assert" public

::method init class
  self~defaultTestResultClass=.TestResult -- set default: use TestResult class
  self~TestCaseInfo=.directory~new
  forward class (super)

::method defaultTestResultClass class attribute -- ooRexxUnit only
```

```
::method TestCaseInfo class attribute


::method init        -- constructor
  expose fName fCountTestCases   -- name of Testcase (method) to carry out
  parse arg fName

  fCountTestCases=1  -- default: individual test
  self~TestCaseInfo=.directory~new -- directory to contain information on test

  self~init:super    -- let superclass initialize


::method TestCaseInfo attribute                -- ooRexxUnit only


::method createResult      -- creates a default TestResult object
  return self~class~defaultTestResultClass~new

::method "countTestCases=" private  -- set method
  expose  fCountTestCases
  use arg fCountTestCases


::method countTestCases -- return nr. of test cases (methods) in this class
  expose fCountTestCases
  return fCountTestCases


::method run            -- will get implemented in subclasses
  expose fName
  use arg aTestResult

   -- make sure an instance of .TestResult is used
  if arg()=0 then aTestResult=self~createResult
             else aTestResult=arg(1)

  aTestResult~startTest(self)       -- remember test started
  self~setUp                        -- make sure setup is invoked before test
  call doTheTest self, fName, aTestResult  -- carry out the testmethod
  self~tearDown                     -- make sure tearDown is invoked after test
  aTestResult~endTest(self)         -- remember test ended

  return aTestResult

doTheTest: procedure    -- make sure exceptions are trapped locally
   use arg self, strM, aTestResult

   /* user exception not interceptable at this stage anymore, hence using SYNTAX
exceptions */
   -- signal on user AssertionFailedError name exceptionHandler

  signal on any name exceptionHandler
/*
   -- signal on any
   -- signal on novalue
   signal on error      name exceptionHandler
   signal on failure    name exceptionHandler
   signal on halt       name exceptionHandler
   signal on nomethod   name exceptionHandler
   -- signal on nostring   name exceptionHandler
   signal on notready   name exceptionHandler
   signal on syntax     name exceptionHandler
*/

-- say "self="pp(self) "strM="pp(strM)"..." "self~hasMethod(strM)="self~hasMethod(strM)

   .message~new(self, strM)~send -- create the message object and send it

   self~check4ConditionFailure   -- check, if a condition was expected and if so, fail
   return aTestResult

exceptionHandler:

   co=condition("O")    -- get the condition directory object

   if self~conditionExpected=.true then
   do
       if self~checkCondition(co) then
          return aTestResult
   end
```

```
   condition=condition("C")      -- get the condition
   additional=condition("A")     -- get additional msg, if any

   strAdditional=""     -- message(s)
   if additional~class=.array then
   do
      items=additional~items
      do i=1 to items
         strAdditional=strAdditional || additional[i]
         if i<items then strAdditional=strAdditional || "0d0a"x  -- add a CRLF
      end
      strAdditional=strAdditional
   end

   if co~code=93.964 then     /* o.k. an own raised exception (using new ooRexx 3.1
application defined syntax error) */
   do
      /* tmpString=pp(date("S") time("L"))":" pp("failure")~left(11) self~string ||
iif(strAdditional="", "", " --->" strAdditional) */
      tmpString=pp(date("S") time("L"))":" pp("failure") self~string ||
iif(strAdditional="", "", " --->" strAdditional)
      co~setentry("OOREXXUNIT.CONDITION", tmpString)  /* add ooRexxUnit-infos with
condition object */
      aTestResult~addFailure(self, co)    -- save the condition object
   end
   else     -- unexpected/untested failure, ie. an "error"
   do
      tmpInfo=co~condition

      -- if tmpInfo="SYNTAX" then tmpInfo=tmpInfo "ERROR"

      if co~hasentry("CODE") then
         tmpInfo=tmpInfo co~code

      tmpInfo="condition" pp(tmpInfo) "raised unexpectedly." || "09"x

      if .nil<>co~message then
         tmpInfo=tmpInfo || co~message   -- add error message

      /* tmpString=pp(date("S") time("L"))":" pp("error")~left(11) self~string "--->"
tmpInfo */
      tmpString=pp(date("S") time("L"))":" pp("error") self~string "--->" tmpInfo
      co~setentry("OOREXXUNIT.CONDITION", tmpString)  /* add ooRexxUnit-infos with
condition object */
      aTestResult~addError(self, co)      -- save the condition object
   end

   return aTestResult   -- rgf, 2006-04-08


::method getName     -- returns the name for this TestCase
  expose fName
  return fName

::method setName     -- set the name for this TestCase
  expose fName
  parse arg fName

::method string      -- create a string representation, counterpart to Javas toString()
  return  makeTestCaseString(self)


::method setUp       /* will get implemented in subclasses (allows to create a test-
environment) */
  NOP                -- indicate that emptiness is intended

::method tearDown    /* will get implemented in subclasses (allows to remove a test-
environment) */
  NOP                -- indicate that emptiness is intended



/* ******************************************************************************
*/
/* ******************************************************************************
*/
::class "TestSuite" subclass TestCase public
::method init
  expose fTestList
```

```
  forward class (super) continue
  fTestList=.queue~new

      -- a class object, use reflection and create test cases
  if arg(1)~class=.class then
  do
     testCaseClass=arg(1)
     fTestMethods.=self~class~getTestMethods(testCaseClass)
     do i=1 to fTestMethods.0
        self~addTest(testCaseClass~new(fTestMethods.i))
     end
  end

::method getTestMethods class -- use reflection to retrieve testmethods, sort
alphabetically
  use arg class

  fTestMethods.0=0      -- set index to 0
   -- now get the test methods, i.e. methods starting with "TEST"
  methSupplier=class~methods(.nil) /* only get methods of the receiver class (=
testClass) */
  do while methSupplier~available -- iterate over supplied methods
     name=methSupplier~index
     if name~left(4)~translate="TEST" then   -- a test method in hand
     do
        i=fTestMethods.0+1
        fTestMethods.i= name      -- index should be uppercase for sorting
        fTestMethods.0=i
     end
     methSupplier~next
  end

  call sysStemSort fTestMethods. -- sort test methods into ascending order
  return fTestMethods.           -- return stem



::method addTest
  expose fTestList
  use arg aTestCase
  fTestList~queue(aTestCase)
  self~countTestCases = self~countTestCases+1


::method run
  expose fTestList
  use arg aTestResult

   -- make sure an instance of .TestResult is used
  if arg()=0 then aTestResult=self~createResult
             else aTestResult=arg(1)

  aTestResult~startTest(self)       -- remember test started
  self~setUp                        -- make sure setup is invoked before testSuite runs
  do aTestCase over fTestList while aTestResult~shouldStop=.false
     aTestCase~run(aTestResult)
  end
  self~tearDown                     -- make sure tearDown is invoked after testSuite ran
  aTestResult~endTest(self)         -- remember test ended

  return aTestResult

::method countTestCases -- return nr. of test cases (methods) in this class
  expose fTestList
  return fTestList~items



/* ******************************************************************************
*/
/* ******************************************************************************
*/
-- routines

/* ******************************************************************************
*/
/* ******************************************************************************
*/
::routine iif public -- utility routine
```

```
  if arg(1)=.true then return arg(2)
                  else return arg(3)


/* ********************************************************************************
*/
/* ********************************************************************************
*/
::routine pp public  -- "pretty print" ;) encloses string value in square brackets
  return "[" || arg(1)~string || "]"

/* ********************************************************************************
*/
/* ********************************************************************************
*/
/* encloses string value in square brackets, escapes non-printable chars as Rexx
   concatenated Rexx hex strings
*/
::routine ppp public  -- "printable pretty print" ;)
  parse arg string    -- retrieve string value of argument

  if verify(string, .chars.NonPrintable, "Match")>0 then
     return "[" || escapeString(string) || "]" -- escape non-printable characters

  return "[" || string || "]"

::routine escapeString        public   -- escape non-printable characters in string
  parse arg str
  tmpStr=.mutableBuffer~new

  do forever while str<>""
     start=verify(str, .chars.nonPrintable, "Match")
     if start>0 then    -- non-printing char found, look for printable char after it
     do
           /* find non-matching position, deduct one to point to last non-printable
chars in string */
        end=verify(str, .chars.nonPrintable, "Nomatch", start)-1
        if end=-1 then   /* no non-matching (=ending) position found: rest is non-
printable */
           end=length(str)

        if start>1 then -- printable chars before section with non-printable chars ?
        do
           chunk=enQuote(substr(str, 1, start-1))
           if tmpStr~length<>0 then tmpStr~~append(" || ")~~append(chunk)
                               else tmpStr~append(chunk)
        end

           -- extract non-printable chars, encode them as a Rexx hex string
        chunk=enQuote(substr(str, start, end-start+1)~c2x) || "x"

        if tmpStr~length<>0 then tmpStr~~append(" || ")~~append(chunk)
                            else tmpStr~append(chunk)

           -- extract non-processed part of string
        str=substr(str, end+1)   -- get remaining string
     end
     else   -- only printable chars available respectively left
     do
        if tmpStr~length<>0 then tmpStr~~append(" || ")~~append(enquote(str))
                            else tmpStr~append(str)
        leave         -- str=""
     end
  end
  return tmpStr~string

-- enQuote: procedure
::routine enQuote             public
  return '"' || arg(1) || '"'

/*
-- return hash value as a Rexx hex literal
::routine getEscapedhashValue  public
  use arg o
  return enQuote(o~"=="~c2x)"x"
*/

-- create
::routine formatObjectInfo public
  use arg o, hint
  if arg(2, 'omitted') then hint=""
```

```
  return pp(hint || ppp(o~string)", hashValue="enQuote(o~"=="~c2x)"x")

-- ppp(arg(1)~string "<hashValue:" getEscapedhashValue(arg(1))">")



   /* function returning .true, if both collections can be regarded to be the same,
      .false else
      logic: - if both values have the method SUBSET, assume unordered collection;
               proceed with test, if both values also possess the method SUPPLIER;
               each value can be of any collection type: returns .true only, if the
               same number of the same index/value pairs is present in both collections,
               .false else
               [there is a short-circuit logic if either value is of .set/.bag type
                in which case the test for equality is carried out directly with the
                help of the SUBSET method]

             - if above does not apply, but both values have the MAKEARRAY method assume
               an ordered collection: values of MAKEARRAY must occur in the same
sequence
               (and both need to have the same number of items) to return .true, .false
else
   */
::routine IsCollEqual public
  use arg expected, received

      -- both collections with "SUPPLIER", if so, sequence should not matter
  if expected~hasmethod("SUBSET") & received~hasmethod("SUBSET") then
  do
     if expected~hasmethod("SUPPLIER") & received~hasmethod("SUPPLIER") then
     do
         /* if both are either of type .set and/or .bag, then test directly with SUBSET
for equality */
        mS=.set~of(.set, .bag)
        if mS~hasindex(expected~class) & mS~hasindex(received~class) then
        do
           if expected~subset(received) then       /* equal, if each is a subset of the
other */
           do
              if received~subset(expected) then
                 return .true
           end

           return .false
        end

            /* create a bag rendering of both collections such, that the resulting
               bags can be used to test whether each is a subset of the other, hence
               can be regarded to be the same collection; the encoding takes the hex-
value
               of the index object and concatenates it with a blank with the hex-value
of
               the item object, creating a uniquely identifiable bag element
            */
        eBag=.bag~new               -- create a bag rendering for the expected value
        eSupp=expected~supplier
        do while eSupp~available
           eBag~put( eSupp~index~"==" eSupp~item~"==" )
           eSupp~next
        end

        rBag=.bag~new               -- create a bag rendering for the received value
        rSupp=received~supplier
        do while rSupp~available
           rBag~put( rSupp~index~"==" rSupp~item~"==" )
           rSupp~next
        end

            /* test whether both collections can be regarded to be the same: if both
bags are
               subsets of each other, both values are regarded to be the same
            */
        if eBag~subset(rBag) then
        do
           if rBag~subset(eBag) then
              return .true
        end
        return .false
     end
  end
```

```
      -- maybe both collections with MAKEARRAY, if so, sequence matters
  if expected~hasmethod("MAKEARRAY") & received~hasmethod("MAKEARRAY") then
  do
     eArr=expected~makearray
     rArr=received~makearray
     if eArr~items<>rArr~items then
        return .false

     do i=1 to eArr~items
        if eArr[i]<>rArr[i] then
           return .false
     end
     return .true
  end

  return .false



/* ******************************************************************************
*/
/* ******************************************************************************
*/
::routine addN       -- if string starts with a vowel, then "n" is returned, "" else
  parse arg name

  if pos(name~left(1), "aeiouAEIOU")>0 then return "n"
  return ""

/* ******************************************************************************
*/
/* ******************************************************************************
*/
::routine makeTestCaseString -- string to represent an instance of a TestCase
  use arg aTestCase
  className=aTestCase~class~id   -- get class name
  return "testCase:" pp(aTestCase~getName) "(a" || addN(className) className || "@" || -
                aTestCase~"=="~c2x")"


/* ******************************************************************************
*/
/* ******************************************************************************
*/
   -- parse file-info into the supplied directory object
   /*
      uses the information about the program in the very first block-comment at the top:

      - keyword":" text
           if keyword starts with "changed", "purpose", "remark", "link", "category"
then
           entry is a queue and text will get enqueued at the end it; the first four
letters
           are used for matching these words

      - arrLines:

   */
::routine makeDirTestInfo public
   use arg aTestCaseClass, arrLines

   tmpDir=aTestCaseClass~TestCaseInfo   -- get directory object to add infos to

   keyWord=""
   tOut=xrange("A","Z")||xrange("a","z")
   tIn =xrange("A","Z")||xrange("a","z")||xrange()

   do i=1 to arrLines~items while arrLines[i]<>"*/"
      if arrLines[i]~strip~left(2)="--" then iterate    -- ignore comment

         -- a keyWord already set and this line has no new keyword, than append it
      if pos(":", arrLines[i])=0 then
      do
         if keyWord<>"" then  -- alreay a keyWord found, append line to it
         do
            tmpDir~entry(keyWord)~queue(arrLines[i])
         end
         iterate
      end
```

```
      parse value arrLines[i] with name ":" rest

      keyWord=name~translate(tOut, tIn)~space(0)   -- a keyWord change ?

      if tmpDir~hasEntry(keyWord)=.false then
         tmpDir~setentry(keyWord, .queue~new)   -- create a new queue for this keyword

      tmpDir~entry(keyWord)~queue(rest~strip)      -- add line
   end



   -- simple dumping of the testResult data
::routine simpleDumpTestResults public
   use arg aTestResult, title

   if arg()>1 & title<>"" then
   do
      say title
      say
   end

   say "nr of test runs:            " aTestResult~runCount
   say "nr of successful assertions:" aTestResult~assertCount

   say "nr of failures:             " aTestResult~failureCount
   if aTestResult~failureCount>0 then
   do
      do co over aTestResult~failures
         say "  " co~ooRexxUnit.condition
      end
   end

   say "nr of errors:               " aTestResult~errorCount
   if aTestResult~errorCount>0 then
   do
      do co over aTestResult~errors
         say "  " co~ooRexxUnit.condition
      end
   end


   /* create a testSuite object by calling the supplied testCaseFileList; needs testCase
programs */
   -- modelled after the example programs
::routine makeTestSuiteFromFileList public
   use arg testCaseFileList, ts

   if arg(2, "Omitted") then  -- no TestSuite object supplied?
      ts=.testSuite~new

      /* make sure, that the tests are not run when CALLing/REQUIREing the testUnit
programs */
   .local~bRunTestsLocally=.false   /* do not run tests, if calling/requiring the
testUnit files */

   do fileName over testCaseFileList
       call (fileName)         -- call file
       testUnitList=result     -- retrieve result (a list of array objects)
       do arr over testUnitList  -- loop over array objects
          classObject   =arr[1]  -- a class object
          mandatoryTests=arr[2]  -- a list

          -- check whether mandatory tests are defined
          bMandatoryTests=(.nil<>arr[2])
          if bMandatoryTests=.true then   -- o.k. not .nil in hand
          do
             bMandatoryTests=(.list=mandatoryTests~class)   -- is there a list in hand
             if bMandatoryTests then
             do
                bMandatoryTests=(mandatoryTests~items>0)    -- are there any entries?
             end
          end

          if bMandatoryTests then   /* mandatory tests available, just use them to
create testCases */
          do
            tsMand=.testSuite~new     -- create a test suite for this test class
```

```
            do testMethodName over mandatoryTests
               tsMand~addTest( classObject~new(testMethodName) )   /* create and add
testCase */
            end
            ts~addTest(tsMand)      /* now add the test suite of mandatory methods to
the overall test suite */
         end
         else    -- no mandatory tests defined, hence use all testmethods
         do
            ts~addTest(.testSuite~new(classObject))  /* creates testCases from all
testmethods */
         end
      end
   end
   return ts      -- return the testSuite object
```

7.8.3.2 XMLPARSER.CLS

XMLPARSER.CSL is used to parse the log files, when comparing them. [cp. Ashl07]

```
/*------------------------------------------------------------------------------*/
/*                                                                              */
/* Description: Very simple class to parse XML.                                 */
/*                                                                              */
/* Copyright (c) 2006 Rexx Language Association. All rights reserved.           */
/*                                                                              */
/* This program and the accompanying materials are made available under         */
/* the terms of the Common Public License v1.0 which accompanies this           */
/* distribution. A copy is also available at the following address:             */
/* http://www.ibm.com/developerworks/oss/CPLv1.0.htm                            */
/*                                                                              */
/* Redistribution and use in source and binary forms, with or                   */
/* without modification, are permitted provided that the following              */
/* conditions are met:                                                          */
/*                                                                              */
/* Redistributions of source code must retain the above copyright               */
/* notice, this list of conditions and the following disclaimer.                */
/* Redistributions in binary form must reproduce the above copyright            */
/* notice, this list of conditions and the following disclaimer in              */
/* the documentation and/or other materials provided with the distribution.     */
/*                                                                              */
/* Neither the name of Rexx Language Association nor the names                  */
/* of its contributors may be used to endorse or promote products               */
/* derived from this software without specific prior written permission.        */
/*                                                                              */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS          */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT            */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS            */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT     */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,        */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED     */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,          */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY       */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING      */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS           */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.                 */
/*                                                                              */
/* Author: W. David Ashley                                                      */
/*                                                                              */
/*------------------------------------------------------------------------------*/


/*------------------------------------------------------------------------------*/
/*                                                                              */
/* Notes:                                                                       */
/*                                                                              */
/* The xmlparser class is a very simple parser for XML files. It is not an      */
/* official 100% compatible XML parser because it has a lot of limitations,     */
/* most of which you could probably care less about.                            */
/*                                                                              */
/* 1. The parser only understands ASCII, which is a valid subset of UTF-8.      */
/*    It does not understand any other encoding except 8-bit ASCII.             */
/* 2. It does not test that the document is well-formed. It assumes that the    */
/*    document is a well-formed XML document.                                   */
```

```
/* 3. The parser does not know how to handle XML processing instructions. It  */
/*    passes those instructions through the passthrough method intact so      */
/*    that the user can try to make sense of them.                            */
/*                                                                            */
/* To use the xmlparser you need to be aware of the following.                */
/*                                                                            */
/* 1. The parser uses a SAX-like interface, but methods of the class are used */
/*    instead of a call-back mechanism. The default methods perform no        */
/*    actions. The user will need to subclass the xmlparser class and         */
/*    override the call-back methods in order to insert their own actions     */
/*    for each XML chunk type.                                                */
/* 2. The call-back methods use the xmlchunk class to pass data to the        */
/*    methods. This is a very simple class and is used as a container for     */
/*    specific types of XML chunks.                                           */
/* 3. Text chunks (CDATA) are passed through the text method intact. If the   */
/*    parser encounters multiple lines of text it invokes the text method     */
/*    for each line individually. It also does not collapse white space chars.*/
/* 4. XML tags are collapsed. This means that if a tag crosses a line         */
/*    boundary then the lines are collapsed together. This is important       */
/*    for processing instruction tags, comment tags and other special tags.   */
/*                                                                            */
/*----------------------------------------------------------------------------*/


/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Class: XMLPARSER                                                           */
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/

::class xmlparser subclass object public


/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Class: XMLPARSER                                                           */
/*        Private methods                                                     */
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/

::method parserver   attribute private -- the version of this parser
::method src         attribute private -- the array of xml lines to be parsed
::method lineidx     attribute private -- the index into the array of xml lines
::method charidx     attribute private -- the index into a xml line
::method errortxt    attribute private -- error text
::method eof         attribute private -- done parsing?


/*----------------------------------------------------------------------------*/
/* Method: create_error                                                       */
/* Description: creates an xmlerror instance.                                 */
/*----------------------------------------------------------------------------*/

::method create_error private
use arg msg, errline, errpos
xmlerror = .xmlerror~new
xmlerror~text = msg
xmlerror~filename = self~filename
xmlerror~line = errline
xmlerror~charpos = errpos
return xmlerror


/*----------------------------------------------------------------------------*/
/* Method: xlatetext                                                          */
/* Description: translate & attributes to their normal characters.            */
/*----------------------------------------------------------------------------*/

::method xlatetext private
use arg text
text = text~changestr('&gt;', '>')
text = text~changestr('&lt;', '<')
text = text~changestr('&amp;', '&') -- always do this one last!
return text


/*----------------------------------------------------------------------------*/
/* Method: current char                                                       */
/* Description: return the current character.                                 */
/*----------------------------------------------------------------------------*/

::method currentchar private
```

```
expose src lineidx charidx
if lineidx > src~items then return .nil
return src[lineidx]~substr(charidx, 1)


/*-----------------------------------------------------------------------------*/
/* Method: getchar                                                             */
/* Description: get a single character from the xml document.                  */
/*-----------------------------------------------------------------------------*/

::method getchar private
expose src lineidx charidx eof
character = src[lineidx]~substr(charidx, 1)
charidx = charidx + 1
if charidx > src[lineidx]~length then do
   lineidx = lineidx + 1
   charidx = 1
   end
if lineidx > src~items then do
   eof = .true
   return ''
   end
return character


/*-----------------------------------------------------------------------------*/
/* Method: getchunk                                                            */
/* Description: returns a chunk of the xml document.                           */
/*-----------------------------------------------------------------------------*/

::method getchunk private
expose src lineidx charidx errortxt eof
errlineidx = lineidx
errcharidx = charidx
chunk = .xmlchunk~new
if self~currentchar() <> '<' then do
   /* we found some CDATA */
   chunk~text = ''
   curline = lineidx
   do while eof = .false & self~currentchar() <> '<'
      -- Do NOT collapse the white space and newlines out of the chunk!
      -- We leave that task up to the client of this class.
      -- Instead, we return each line of text individually.
      chunk~text = chunk~text || self~getchar()
      if curline <> lineidx then leave
      end
   if eof = .true & chunk~text~strip <> '' then do
      errortxt = 'Error line' errlineidx 'column 'errcharidx': EOF within CDATA.'
      self~error(self~create_error(errortxt, errlineidx, errcharidx)) /* call the public
override method */
      return .nil
      end
   chunk~text = self~xlatetext(chunk~text)
   if chunk~text~strip <> '' then
   self~text(chunk) -- call the public override method
   return chunk
   end
/* we found an XML tag, process it */
character = self~getchar() -- skip the '<'
element = ''
curline = lineidx
nestlevel = 0
do while eof = .false
   if element~substr(1, 1) = '!' then do
      -- It is possible for tags to be contained within other tags in XML
      -- processing tags. The next two IF statements take care of that nesting
      -- possibility. It will be up to the user to parse out the contained
      -- tags.
      if self~currentchar() = '<' then nestlevel = nestlevel + 1
      if self~currentchar() = '>' & level > 0 then nestlevel = nestlevel - 1
      end
   element = element || self~getchar()
   if curline <> lineidx then do
      element = element || ' '
      curline = lineidx
      end
   if self~currentchar() = '>' & nestlevel = 0 then leave
   end
if eof = .true then do
   errortxt = 'Error line' errlineidx': EOF within an XML tag.'
```

```
   self~error(self~create_error(errortxt, errlineidx, errcharidx)) /* call the public
override method */
   return .nil
   end
element = element~strip()
select
   when element~substr(1, 1) = '/' then do
      chunk~tag = element~substr(2)
      self~end_element(chunk) -- call the public override method
      end
   when pos('?', element~substr(1, 1)) > 0 then do
      chunk~tag = ''
      chunk~text = element
      self~passthrough(chunk) -- call the public override method
      end
   when pos('!--', element~substr(1, 3)) > 0 then do
      chunk~tag = ''
      chunk~text = element
      self~passthrough(chunk) -- call the public override method
      end
   when pos('!', element~substr(1, 1)) > 0 then do
      chunk~tag = ''
      chunk~text = element
      self~passthrough(chunk) -- call the public override method
      end
   when pos(element~substr(1, 1), xrange('a', 'z') || xrange('A', 'Z')) > 0 then do
      parse var element tag element
      chunk~tag = tag
      /* process the attributes */
      if element~length > 0 then chunk~attr = .directory~new
      do while element~length() > 0
         if pos('=', element~word(1)) > 0 then do
            parse var element attrname '="' attrvalue '"' element
            attrname = attrname~strip()
            attrvalue = attrvalue~strip()
            attrvalue = self~xlatetext(attrvalue)
            chunk~attr[attrname] = attrvalue
            end
         else do
            parse var element attrname element
            if attrname <> '/' then do
               -- do not allow attributes without values!
               errortxt = 'Error line' errlineidx 'column' errcharidx || ,
                          ': Invalid tag attribute' attrname'.'
               self~error(self~create_error(errortxt, errlineidx, errcharidx)) /* call
the public override method */
               /* stop parsing */
               eof = .true
               return .nil
               end
            end
         end
      self~start_element(chunk) -- call the public override method
      if attrname = '/' then do
         endchunk = .xmlchunk~new
         endchunk~tag = tag
         self~end_element(endchunk) -- call the public override method
         end
      end
   otherwise do
      errortxt = 'Error line' errlineidx 'column 'errcharidx': Invalid tag name.'
      self~error(self~create_error(errortxt, errlineidx, errcharidx)) /* call the public
override method */
      /* stop parsing */
      eof = .true
      return .nil
      end
   end
character = self~getchar() -- skip the '>'
return chunk


/*--------------------------------------------------------------------------*/
/*--------------------------------------------------------------------------*/
/* Class: XMLPARSER                                                         */
/*        Public methods                                                    */
/*--------------------------------------------------------------------------*/
/*--------------------------------------------------------------------------*/
```

```
::method filename attribute private -- the XML file name, if known


/*-----------------------------------------------------------------------------*/
/* Method: init                                                                */
/* Description: instance initialization                                        */
/*-----------------------------------------------------------------------------*/

::method init
expose src
if arg() > 0 then raise syntax 93.902 array (0)
self~parserver = '0.3'
self~filename = ''
return


/*-----------------------------------------------------------------------------*/
/* Method: start_element                                                       */
/* Description: called when a start element tag has been encountered.          */
/* Arguments:   an xmlchunk instance.                                          */
/*-----------------------------------------------------------------------------*/

::method start_element
/* this method is designed to be overridden by a subclass */
use arg chunk
return


/*-----------------------------------------------------------------------------*/
/* Method: end_element                                                         */
/* Description: called when an end element tag has been encountered.           */
/* Arguments:   an xmlchunk instance.                                          */
/*-----------------------------------------------------------------------------*/

::method end_element
/* this method is designed to be overridden by a subclass */
use arg chunk
return


/*-----------------------------------------------------------------------------*/
/* Method: text                                                                */
/* Description: called when character data has been encountered.               */
/* Arguments:   an xmlchunk instance.                                          */
/*-----------------------------------------------------------------------------*/

::method text
/* this method is designed to be overridden by a subclass */
use arg chunk
return


/*-----------------------------------------------------------------------------*/
/* Method: passthrough                                                         */
/* Description: called when comment tag or a processing instruction has been  */
/*              encountered.                                                   */
/* Arguments:   an xmlchunk instance.                                          */
/*-----------------------------------------------------------------------------*/

::method passthrough
/* this method is designed to be overridden by a subclass */
use arg chunk
return


/*-----------------------------------------------------------------------------*/
/* Method: error                                                               */
/* Description: called on an error.                                            */
/* Arguments:   an xmlerror instance.                                          */
/*-----------------------------------------------------------------------------*/

::method error
/* this method is designed to be overridden by a subclass */
use arg xmlerror
return


/*-----------------------------------------------------------------------------*/
/* Method: getversion                                                          */
/* Description: return the version of this class.                              */
/*-----------------------------------------------------------------------------*/

::method getversion
```

```
return self~parserver


/*-----------------------------------------------------------------------------*/
/* Method: parse_array                                                         */
/* Description: parse the specified array of XML code.                         */
/*-----------------------------------------------------------------------------*/

::method parse_array
expose src lineidx charidx errortxt eof
if arg() < 1 then raise syntax 93.901 array (1)
if arg() > 1 then raise syntax 93.902 array (1)
use arg src
eof = .false
/* make sure this is an xml document */
if src[1]~pos('<?xml') <> 1 then do
   errortxt = 'Error: Invalid XML document.'
   self~error(self~create_error(errortxt, 1, 1))
   return
   end
/* parse the xml array */
lineidx = 1
charidx = 1
errortxt = ''
do while eof = .false
   chunk = self~getchunk()
   end
return errortxt


/*-----------------------------------------------------------------------------*/
/* Method: parse_file                                                          */
/* Description: parse the specified file of XML code.                          */
/*-----------------------------------------------------------------------------*/

::method parse_file
expose errortxt filename
if arg() < 1 then raise syntax 93.901 array (1)
if arg() > 1 then raise syntax 93.902 array (1)
use arg xmlfile
tfile = .stream~new(xmlfile)
errortxt = tfile~open('read')
if errortxt <> 'READY:' then do
   tfile~close()
   return errortxt
   end
lines = tfile~arrayin()
tfile~close()

errortxt = self~parse_array(lines)
return errortxt


/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLCHUNK                                                             */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

::class xmlchunk subclass object public


/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLCHUNK                                                             */
/*        Private methods                                                      */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLCHUNK                                                             */
/*        Public methods                                                       */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

::method text        attribute        -- the text
::method tag         attribute        -- the xml tag name
::method attr        attribute        -- the tag attributes
```

```
/*-----------------------------------------------------------------------------*/
/* Method: init                                                                */
/* Description: instance initialization                                        */
/*-----------------------------------------------------------------------------*/

::method init
self~text = .nil  -- For the start_element and passthrough methods this contains
                  -- the entire text string enclosed within the '<' and '>'
                  -- brackets. For the text method it contains a single line
                  -- of CDATA text.
self~tag = .nil   -- For  the start_element and end_element methods this is
                  -- the XML element (tag) name. For the end_element method the
                  -- leading '/' character is not a part of this string.
self~attr = .nil  -- For the start_element method this is an ooRexx directory
                  -- class instance. Each attribute and value is contained in
                  -- the ooRexx directory instance.
return


/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLERROR                                                             */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

::class xmlerror subclass object public


/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLERROR                                                             */
/*        Private methods                                                      */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/
/* Class: XMLERROR                                                             */
/*        Public methods                                                       */
/*-----------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------*/

::method text        attribute        -- the error message text, if any
::method filename    attribute        -- the xml file name, if known
::method line        attribute        -- the error line number
::method charpos     attribute        -- the error character position


/*-----------------------------------------------------------------------------*/
/* Method: init                                                                */
/* Description: instance initialization                                        */
/*-----------------------------------------------------------------------------*/

::method init
self~text = ''
self~filename = ''
self~line = 0
self~charpos = 0
return
```

7.8.4 Test Unit

The following chapter shows an example of a test unit for the ooRexxUnit framework.

7.8.4.1 OOREXX.BASE.CLASS.MUTABLEBUFFER.TESTUNIT

OOREXX.BASE.CLASS.MUTABLEBUFFER.TESTUNIT is used to test all methods of the ooRexx class MutableBuffer.

```
/*
        name:           ooRexx.Base.Class.MutableBuffer.testUnit
        author:         Rainer Kegel
        date:           2007-01-02

        purpose:        Test the methods of the class MutableBuffer
        license:        CPL 1.0
        link:

        category:       ooRexx
        category:       Base
        category:       Class
        category:       Mutable Buffer
*/
/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Copyright (c) 2007 Rainer Kegel. All rights reserved.                      */
/*                                                                            */
/* This program and the accompanying materials are made available under       */
/* the terms of the Common Public License v1.0 which accompanies this         */
/* distribution. A copy is also available at the following address:           */
/* http://www.ibm.com/developerworks/oss/CPLv1.0.htm                          */
/*                                                                            */
/* Redistribution and use in source and binary forms, with or                 */
/* without modification, are permitted provided that the following            */
/* conditions are met:                                                        */
/*                                                                            */
/* Redistributions of source code must retain the above copyright             */
/* notice, this list of conditions and the following disclaimer.              */
/* Redistributions in binary form must reproduce the above copyright          */
/* notice, this list of conditions and the following disclaimer in            */
/* the documentation and/or other materials provided with the distribution.   */
/*                                                                            */
/* Neither the name of Rexx Language Association nor the names                */
/* of its contributors may be used to endorse or promote products             */
/* derived from this software without specific prior written permission.      */
/*                                                                            */
/* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS        */
/* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          */
/* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          */
/* FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT   */
/* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,      */
/* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED   */
/* TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,        */
/* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY     */
/* OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING    */
/* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS         */
/* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.               */
/*                                                                            */
/*----------------------------------------------------------------------------*/


testUnitClass=.ooRexx.Base.Class.MutableBuffer.TestUnit  -- change accordingly
/*----------------------------------------------------------------------------
-- ===> adapt the "testUnitList" to your testCase classes; each element in the list is
<===
-- ===> an array object, the first element containing the testCase class object, the
<===
-- ===> second element is a list of test method names which are regarded to be
<===
-- ===> mandatory (if the list remains empty all test methods are mandatory)
<===
```

```
*/
   /* list of array objects, each containing the testUnit class object and an
      optional list of mandatory test case methods name                      */
mandatoryTestMethods=.list~new   -- no mandatory tests for this testCase class
testUnitList=.list~of( .array~of(.ooRexx.Base.Class.MutableBuffer.testUnit,
mandatoryTestMethods) )


-------------------------------------------------------------------------------
---
-- ===> the following code needs not to be individualized                  <===

   -- read top comment, containing infos about this program
arrLines=.array~new
do i=1 to 150 until arrLines[i]="*/"
   arrLines[i]=sourceline(i)
end
      -- supply information for the testClass(es) in this file; the class attribute
      -- "TestCaseInfo" (a directory object, index points to a queue) will store
      -- the parsed infos
aTestUnitClass=testUnitList~at(testUnitList~first)[1] -- get first testClass

   -- will parse the array lines and store result in class object
call makeDirTestInfo aTestUnitClass, arrLines
tmpDir=aTestUnitClass~TestCaseInfo
parse source s   -- op_sys invocationType fullPathToThisFile
tmpDir~setentry("testCase-source", s)

   -- now add this directory to other testCase classes, if any left
do arr over testUnitList
   if arr[1]=aTestUnitClass then iterate  -- already handled
   arr[1]~TestCaseInfo=tmpDir             -- save info in class object
end

/* if this file is CALLed or REQUIRED then define an entry "bRunTestLocally" in .local
 and set it to .false; this way the independent local invocation of the tests is
inhibited */
if .local~hasentry("bRunTestsLocally")=.false then
   .local~bRunTestsLocally=.true /* if this file is executed directly, then run tests
for debugging */

if .bRunTestsLocally=.true then  -- run ALL tests in this test unit
do
   ts=.testSuite~new             -- create a testSuite
   do arr over testUnitList
      -- create a testSuite for the given test case class, use all its testmethods
      ts~addTest( .testSuite~new(arr[1]))
   end
   -- testResult=.testSuite~new(testUnitClass)~run
   testResult=ts~run       -- now run all the tests

   call simpleDumpTestResults testResult
end

   /* return list of array objects containing test case classes and
      optionally list of mandatory test methods                     */
return testUnitList



::requires ooRexxUnit.cls       --load the ooRexxUnit classes

        -- class named exactly like file
::class "ooRexx.Base.Class.MutableBuffer.TestUnit" subclass TestCase public


     --test INIT
::method "test_INIT"

    TestBufferINIT=.mutableBuffer~new
        TestBufferINIT~insert(“TEST4”,,5)
        TestBufferINIT~insert(“TEST3”,5,5)
        TestBufferINIT~insert(“TEST2”,10,5)
        TestBufferINIT~insert(“TEST1”,15,5)

        self~assertSame("subtest1", "TEST4TEST3TEST2TEST1", TestBufferINIT~string)


::method "test_APPEND"

    TestBufferAPP=.mutableBuffer~new
```

```
        TestBufferAPP~insert(“TEST4”)
        TestBufferAPP~insert(“TEST3”)
        TestBufferAPP~insert(“TEST2”)
        TestBufferAPP~insert(“TEST1”)

    TestBufferAPP~append(“TEST “5)
        self~assertSame("subTest1",     "TEST1TEST2TEST3TEST4TEST5", -
        TestBufferAPP~string)

    TestBufferAPP~append(“TEST6”)
        self~assertSame("subTest2", "TEST1TEST2TEST3TEST4TEST5TEST6", -
        TestBufferAPP~string)

    TestBufferAPP~append(“TEST7”)
        self~assertSame("subTest3",     "TEST1TEST2TEST3TEST4TEST5TEST6TEST7", -
        TestBufferAPP~string)

    TestBufferAPP~append(“TEST8”)
        self~assertSame("subTest4",     "TEST1TEST2TEST3TEST4TEST5TEST6TEST7TEST8", -
        TestBufferAPP~string)


::method "test_DELETE" --

    TestBufferDEL=.mutableBuffer~new
        TestBufferDEL~insert(“TEST4”)
        TestBufferDEL~insert(“TEST3”)
        TestBufferDEL~insert(“TEST2”)
        TestBufferDEL~insert(“TEST1”)

    TestBufferDEL~delete(5,1)
        self~assertSame("subTest1", "TESTTEST2TEST3TEST4" , TestBufferDEL~string)

        TestBufferDEL~delete(9,1)
        self~assertSame("subTest2", "TESTTESTTEST3TEST4" , TestBufferDEL~string)

        TestBufferDEL~delete(13,1)
        self~assertSame("subTest3", "TESTTESTTESTTEST4" , TestBufferDEL~string)

        TestBufferDEL~delete(17,1)
        self~assertSame("subTest4", "TESTTESTTESTTEST" , TestBufferDEL~string)

    TestBufferDEL~delete(9,8)
        self~assertSame("subTest5", "TESTTEST" , TestBufferDEL~string)


::method "test_GETBUFFERSIZE" --

    TestBufferGBS=.mutableBuffer~new
        TestBufferGBS~insert(“TEST4”)
        TestBufferGBS~insert(“TEST3”)
        TestBufferGBS~insert(“TEST2”)
        TestBufferGBS~insert(“TEST1”)

        self~assertSame("subtest1", "256", TestBufferGBS~getbuffersize()        )

        TestBufferGBS~setbuffersize(1000)
        self~assertSame("subtest2", "1000", TestBufferGBS~getbuffersize()       )


::method "test_INSERT"

    TestBufferIN=.mutableBuffer~new
        TestBufferIN~insert(“TEST4”)
        TestBufferIN~insert(“TEST3”)
        TestBufferIN~insert(“TEST2”)
        TestBufferIN~insert(“TEST1”)

        self~assertSame("subtest1", "TEST1TEST2TEST3TEST4", TestBufferIN~string)

        TestBufferIN~insert(AAAAA,5)
        self~assertSame("subtest2", "TEST1AAAAATEST2TEST3TEST4", TestBufferIN~string)

        TestBufferIN~insert(BBBBB,,10)
        self~assertEquals("subtest3", "BBBBB     TEST1AAAAATEST2TEST3TEST4", -
        TestBufferIN~string)

::method "test_LENGTH"
```

```
    TestBufferLG=.mutableBuffer~new
        TestBufferLG~insert(“TEST4”)
        TestBufferLG~insert(“TEST3”)
        TestBufferLG~insert(“TEST2”)
        TestBufferLG~insert(“TEST1”)

        self~assertSame("subTest1",     "20", TestBufferLG~length())

        TestBufferLG~insert(“TEST5”)
        self~assertSame("subtest2", "25", TestBufferLG~length())

        TestBufferLG~insert(“TEST6”)
        self~assertSame("subtest3", "30", TestBufferLG~length())


::method "test_OVERLAY"

    TestBufferOV=.mutableBuffer~new
    TestBufferOV~insert(“TEST6”)
    TestBufferOV~insert(“TEST5”)
        TestBufferOV~insert(“TEST4”)
        TestBufferOV~insert(“TEST3”)
        TestBufferOV~insert(“TEST2”)
        TestBufferOV~insert(“TEST1”)

    TestBufferOV~overlay("mutableBuffer",6)
        self~assertSame("subtest1", "TEST1mutableBufferT4TEST5TEST6", -
                        TestBufferOV~string       )

    TestBufferOV~overlay("MUTABLEBUFFER",,20)
        self~assertEquals("subtest2", "MUTABLEBUFFER       TEST5TEST6", -
                          TestBufferOV~string             )


::method "test_SETBUFFERSIZE"

    TestBufferSBS=.mutableBuffer~new
        TestBufferSBS~insert(“TEST4”)
        TestBufferSBS~insert(“TEST3”)
        TestBufferSBS~insert(“TEST2”)
        TestBufferSBS~insert(“TEST1”)

    TestBufferSBS~setBufferSize(25)
        self~assertSame("subtest1", "25", TestBufferSBS~getBufferSize()         )

        TestBufferSBS~insert("CharactersToExpandBuffersize")
        self~assertTrue("subtest2", TestBufferSBS~getBufferSize()~'>'(25)       )
        self~assertTrue("subtest3", TestBufferSBS~getBufferSize()~'='(50)       )


::method "test_STRING"
        TestBufferSTG=.mutableBuffer~new
        TestBufferSTG~insert(“TEST4”)
        TestBufferSTG~insert(“TEST3”)
        TestBufferSTG~insert(“TEST2”)
        TestBufferSTG~insert(“TEST1”)


        self~assertSame("subtest1", "TEST1TEST2TEST3TEST4", TestBufferSTG~string    )

        TestBufferSTG~overlay(AAAAABBBBB,6)
        self~assertSame("subtest2", "TEST1AAAAABBBBBTEST4", TestBufferSTG~string)


::method "test_SUBSTR"

        TestBufferSUB=.mutableBuffer~new
        TestBufferSUB~insert(“TEST4”)
        TestBufferSUB~insert(“TEST3”)
        TestBufferSUB~insert(“TEST2”)
        TestBufferSUB~insert(“TEST1”)

        self~assertSame("subtest1", "TEST2TEST3", TestBufferSUB~Substr(6,10)    )
        self~assertSame("subtest2", "TEST1", TestBufferSUB~Substr(1,5)    )

        TestBufferSUB~overlay("mytest",6)
        self~assertSame("subtest3", "TEST1mytest", TestBufferSUB~Substr(1,11)   )
```

7.8.5 Extensible Style Sheet Language Files

XSL files are used in the course of TeRA to transform log files and compared log files (both XML files) into HTML.

7.8.5.1 LOG.XSL

LOG.XSL transforms the log files (XML files) into HTML.

```
<!--                                                                          -->
<!--   name:        log.xsl                                                   -->
<!--   author:      Rainer Kegel                                              -->
<!--   date:        2007-01-02                                                -->
<!--                                                                          -->
<!--   purpose:     part of the Test Runner Application for ooRexxUnit        -->
<!--                - provides general structures and routines                -->
<!--                                                                          -->
<!--   license:     CPL 1.0 (Common Public License v1.0, see below)           -->
<!--                                                                          -->
<!--                                                                          -->
<!--  ------------------------------------------------------------------------  -->
<!--                                                                          -->
<!--   Copyright (c) 2007 Rainer Kegel. All rights reserved.                  -->
<!--                                                                          -->
<!--   This program and the accompanying materials are made available under   -->
<!--   the terms of the Common Public License v1.0 which accompanies this     -->
<!--   distribution. A copy is also available at the following address:       -->
<!--   http://www.opensource.org/licenses/cpl1.0.php                          -->
<!--                                                                          -->
<!--   Redistribution and use in source and binary forms, with or             -->
<!--   without modification, are permitted provided that the following        -->
<!--   conditions are met:                                                    -->
<!--                                                                          -->
<!--   Redistributions of source code must retain the above copyright         -->
<!--   notice, this list of conditions and the following disclaimer.          -->
<!--   Redistributions in binary form must reproduce the above copyright      -->
<!--   notice, this list of conditions and the following disclaimer in        -->
<!--   the documentation and/or other materials provided with the distribution.  -->
<!--                                                                          -->
<!--   Neither the name of Rexx Language Association nor the names            -->
<!--   of its contributors may be used to endorse or promote products         -->
<!--   derived from this software without specific prior written permission.  -->
<!--                                                                          -->
<!--   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS    -->
<!--   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT      -->
<!--   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS      -->
<!--   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  -->
<!--   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  -->
<!--   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED  -->
<!--   TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,    -->
<!--   OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY -->
<!--   OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING  -->
<!--   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS     -->
<!--   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.           -->
<!--                                                                          -->
<!--  ------------------------------------------------------------------------  -->


<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:template match="/">
      <html>
         <head>
            <link rel='stylesheet' href='log.css' type='text/css' />
            <script language="Object Rexx" src="log_xsl_script.rex"></script>
         </head>
         <body class="var1">
            <div class="var2">
               <xsl:apply-templates />
            </div>
            <div class="var3">
               <input type="button" style="width:100px" value="hide okay" -
               id="buttonokay" alt="show/hide okay" onclick="call okay" />
```

```
            <input type="button" style="width:100px" value="hide failure" –
            id="buttonfailure" alt="show/hide failure" onclick="call failure" />
            <input type="button" style="width:100px" value="hide error" –
            id="buttonerror" alt="show/hide error" onclick="call error" />
         </div>
      </body>
   </html>
</xsl:template>

<xsl:template match="username">
   <table width="50%">
      <tr>
         <td class="var4">USERNAME:</td>
         <td colspan="2"  class="var5"><xsl:value-of select="." /></td>
         <td></td>
      </tr>
   </table>
   <p></p>
</xsl:template>

<xsl:template match="comment">
   <table width="50%">
      <tr>
         <td class="var4">COMMENT:</td>
         <td colspan="2" class="var6"><xsl:value-of select="." /></td>
         <td></td>
      </tr>
   </table>
   <p></p>
</xsl:template>

<xsl:template match="runs">
   <table width="50%">
      <tr>
         <td class="var4">COUNTS:</td>
         <td>
            <table class="var6" width="100%">
               <tr>
                  <xsl:apply-templates select="testruns" />
               </tr>
               <tr>
                  <xsl:apply-templates select="errcounts" />
               </tr>
               <tr>
                  <xsl:apply-templates select="failcount" />
               </tr>
               <tr>
                  <xsl:apply-templates select="runcount" />
               </tr>
            </table>
         </td>
      </tr>
   </table>
   <p></p>
</xsl:template>

<xsl:template match="testruns">
   <td class="var7">TESTRUNS:</td>
   <td class="var8"><xsl:value-of select="." /></td>
</xsl:template>

<xsl:template match="errcounts">
   <td class="var7">ERRORCOUNTS:</td>
   <td class="var8"><xsl:value-of select="." /></td>
</xsl:template>

<xsl:template match="failcount">
   <td class="var7">FAILURECOUNTS:</td>
   <td class="var8"><xsl:value-of select="." /></td>
</xsl:template>

<xsl:template match="runcount">
   <td class="var7">RUNCOUNTS:</td>
   <td class="var8"><xsl:value-of select="." /></td>
</xsl:template>



<xsl:template match="starttestsuite|startdatets">
   <table class="testsuite">
```

```
         <tr>
            <td class="var17">Testsuite started:</td>
            <td class="var18"><xsl:value-of select="." /></td>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="maintestunit">
      <table width="100%">
         <xsl:if test="@id">
            <xsl:attribute name="id">
               <xsl:value-of select="@id" />
            </xsl:attribute>
         </xsl:if>
         <tr>
            <td>
               <xsl:apply-templates />
            </td>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="startend">
      <tr>
         <td class="space"></td>
      </tr>
      <table class="testcase">
         <tr>
            <th colspan="4" class="var9">
               <xsl:apply-templates select="namemain"/>
            </th>
            <th></th><th></th><th></th>
         </tr>
         <tr>
            <xsl:apply-templates select="mainstartdate"/>
            <xsl:apply-templates select="mainstarttime"/>
         </tr>
         <tr>
            <xsl:apply-templates select="mainenddate"/>
            <xsl:apply-templates select="mainendtime"/>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="namemain">
      <xsl:value-of select="." />
   </xsl:template>

   <xsl:template match="mainstartdate">
      <td class="var10">Start:</td>
      <td class="var11"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="mainenddate">
      <td class="var10">End:</td>
      <td class="var11"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="mainstarttime | mainendtime">
      <td style="text-align:left;" width="12%;">
         <xsl:value-of select="." />
      </td>
   </xsl:template>

   <xsl:template match="subtest">
      <table width="100%">
         <xsl:if test="@id">
            <xsl:attribute name="id">
               <xsl:value-of select="@id" />
            </xsl:attribute>
         </xsl:if>
         <xsl:if test="@value">
            <xsl:attribute name="value">
               <xsl:value-of select="@value" />
            </xsl:attribute>
         </xsl:if>
         <xsl:choose>
            <xsl:when test="@value='failure'">
               <xsl:attribute name="class">failure</xsl:attribute>
            </xsl:when>
```

```
            <xsl:when test="@value='error'">
               <xsl:attribute name="class">error</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
               <xsl:attribute name="class">okay</xsl:attribute>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates />
         <tr>
            <td height="10px" bgcolor="#e0e0e0"></td>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="name">
      <tr>
         <th colspan="4" class="var12">
            <xsl:value-of select="." />
         </th>
      </tr>
   </xsl:template>

   <xsl:template match="main">
      <tr>
         <th /><th /><th />
         <th class="var13" colspan="2">
            <xsl:value-of select="." />
         </th>
      </tr>
   </xsl:template>

   <xsl:template match="linestart">
      <tr>
         <td colspan="3">
           <table>
                     <tr>
                       <xsl:apply-templates select="substartdate" />
                       <xsl:apply-templates select="substarttime" />
                     </tr>
                   </table>
         </td>
      </tr>
   </xsl:template>

   <xsl:template match="substarttime">
      <td class="var9"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="substartdate">
      <td class="var16">START:</td><td class="var9"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="lineend">
      <tr>
         <td colspan="3">
           <table>
                     <tr>
                       <xsl:apply-templates select="subenddate" />
                       <xsl:apply-templates select="subendtime" />
                     </tr>
                   </table>
         </td>
      </tr>
   </xsl:template>

   <xsl:template match="subendtime">
      <td class="var9"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="subenddate">
      <td class="var16">END:</td><td class="var9"><xsl:value-of select="." /></td>
   </xsl:template>

   <xsl:template match="status">
      <tr>
         <th class="var9"><xsl:value-of select="." /></th>
      </tr>
   </xsl:template>

   <xsl:template match="infotable">
```

```
      <table width="100%;">
         <xsl:choose>
            <xsl:when test="@value='failure'">
               <xsl:attribute name="class">failure</xsl:attribute>
            </xsl:when>
            <xsl:when test="@value='error'">
               <xsl:attribute name="class">error</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
               <xsl:attribute name="class">okay</xsl:attribute>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates>
            <xsl:sort select="@index"></xsl:sort>
            <xsl:sort select="@n_index"></xsl:sort>
         </xsl:apply-templates>
      </table>
   </xsl:template>

   <xsl:template match="line">
      <tr>
         <xsl:apply-templates />
      </tr>
   </xsl:template>

   <xsl:template match="lineadd">
      <tr>
         <xsl:apply-templates />
      </tr>
   </xsl:template>

   <xsl:template match="info">
      <td style="text-align:left;">
         <xsl:choose>
            <xsl:when test="@col='span'">
               <xsl:attribute name="colspan">3</xsl:attribute>
               <xsl:attribute name="class">var14</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
               <xsl:attribute name="style">font-size:11px;</xsl:attribute>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:value-of select="." />
      </td>
   </xsl:template>

   <xsl:template match="endtestsuite">
      <table class="testsuite">
         <tr>
            <td>Time testsuite ended: <xsl:value-of select="." /></td>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="enddatets">
      <table class="testsuite">
         <tr>
            <td>Date testsuite ended: <xsl:value-of select="." /></td>
         </tr>
      </table>
   </xsl:template>

   <xsl:template match="subtestnumber">
      <table>
         <xsl:if test="@id">
            <xsl:attribute name="id">
               <xsl:value-of select="@id" />
            </xsl:attribute>
         </xsl:if>
         <xsl:if test="@value">
            <xsl:attribute name="value">
               <xsl:value-of select="@value" />
            </xsl:attribute>
         </xsl:if>
      </table>
   </xsl:template>

</xsl:stylesheet>
```

7.8.5.2 COMPLOG.XSL

COMPLOG.XSL transforms the compared log files (XML files) into HTML.

```
<!--                                                                          -->
<!--   name:        complog.xsl                                               -->
<!--   author:      Rainer Kegel                                              -->
<!--   date:        2007-01-02                                                -->
<!--                                                                          -->
<!--   purpose:     part of the Test Runner Application for ooRexxUnit        -->
<!--                - provides general structures and routines                -->
<!--                                                                          -->
<!--   license:     CPL 1.0 (Common Public License v1.0, see below)           -->
<!--                                                                          -->
<!--                                                                          -->
<!--  ------------------------------------------------------------------------  -->
<!--                                                                          -->
<!--   Copyright (c) 2005 Rexx Language Association. All rights reserved.     -->
<!--                                                                          -->
<!--   This program and the accompanying materials are made available under   -->
<!--   the terms of the Common Public License v1.0 which accompanies this     -->
<!--   distribution. A copy is also available at the following address:       -->
<!--   http://www.opensource.org/licenses/cpl1.0.php                          -->
<!--                                                                          -->
<!--   Redistribution and use in source and binary forms, with or             -->
<!--   without modification, are permitted provided that the following        -->
<!--   conditions are met:                                                    -->
<!--                                                                          -->
<!--   Redistributions of source code must retain the above copyright         -->
<!--   notice, this list of conditions and the following disclaimer.          -->
<!--   Redistributions in binary form must reproduce the above copyright      -->
<!--   notice, this list of conditions and the following disclaimer in        -->
<!--   the documentation and/or other materials provided with the distribution.  -->
<!--                                                                          -->
<!--   Neither the name of Rexx Language Association nor the names            -->
<!--   of its contributors may be used to endorse or promote products         -->
<!--   derived from this software without specific prior written permission.  -->
<!--                                                                          -->
<!--   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS    -->
<!--   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT      -->
<!--   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS      -->
<!--   FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT  -->
<!--   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,  -->
<!--   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED  -->
<!--   TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,    -->
<!--   OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY -->
<!--   OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING  -->
<!--   NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS     -->
<!--   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.           -->
<!--                                                                          -->
<!--  ------------------------------------------------------------------------  -->

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:template match="/">
      <html>
         <head>
            <link rel='stylesheet' href='complog.css' type='text/css' />
            <script language="object rexx" src="complog_xsl_script.rex"></script>
         </head>
         <body onload="call buttons">
            <div id="content_container">
               <xsl:apply-templates />
            </div>
            <div id="footer">
               <input type="button" id="buttonchanges" onclick="call changes" -
               value="hide changes"></input>
               <input type="button" id="buttonlog1" onclick="call log1"  value="hide -
               log1"></input>
               <input type="button" id="buttonlog2" onclick="call log2"  value="hide -
               log2"></input>
               <input type="button" id="buttonrunboth" onclick="call runboth"  -
               value="hide runboth"></input>
               <input type="button" id="buttonrunonce" onclick="call runonce"  -
               value="hide runonce"></input>
               <span style="width:10%;"> </span>
               <span id="leg1" style="background:#C6B5DE; font-size:12px;">test case in
this log only</span>
```

```
               <span style="width:10px;"> </span>
               <span id="leg2" style="background:#e0e0e0; font-size:12px;">test case in
both logs</span>
            </div>
         </body>
      </html>
   </xsl:template>

   <xsl:template match="changes">
      <table width="100%">
         <tr>
            <td>
               <xsl:apply-templates />
            </td>
         </tr>
      </table>
      <p/>
   </xsl:template>

   <xsl:template match="headline">
      <tr>
         <th colspan="2" class="var1">
            <xsl:value-of select="." />
         </th>
      </tr>
      <tr></tr>
   </xsl:template>

   <xsl:template match="name1">
      <td class="var2">
         <xsl:value-of select="." />
      </td>
   </xsl:template>

   <xsl:template match="name2">
      <td class="var2">
         <xsl:value-of select="." />
      </td>
      <tr></tr>
      </xsl:template>

   <xsl:template match="tutable">
      <table class="changes">
         <tr>
            <td class="var3">
               <xsl:apply-templates select="table1"></xsl:apply-templates>
            </td>
            <td class="var3">
               <xsl:apply-templates select="table2"></xsl:apply-templates>
            </td>
         </tr>
      </table>
      <p></p>
   </xsl:template>

   <xsl:template match="table1 | table2">
      <table>
         <xsl:apply-templates>
            <xsl:sort select="@index" data-type="text"></xsl:sort>
            <xsl:sort select="@nindex" data-type="number"></xsl:sort>
         </xsl:apply-templates>
      </table>
   </xsl:template>

   <xsl:template match="line1 | line2">
      <tr>
         <xsl:apply-templates />
      </tr>
   </xsl:template>

   <xsl:template match="complog1 | complog2">
      <td style="font-size:11px;">
                  <xsl:if test="@check='d'">
                     <xsl:attribute name="class">width</xsl:attribute>
                  </xsl:if>
         <xsl:choose>
            <xsl:when test="@col='span'">
               <xsl:attribute name="colspan">3</xsl:attribute>
            </xsl:when>
            <xsl:when test="@index='AAAAAAAAAA'">
```

```
               <xsl:attribute name="style">font-weight:bold;</xsl:attribute>
            </xsl:when>
            <xsl:when test="@value='diff'">
               <xsl:attribute name="class">yellow</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
               <xsl:attribute name="bgcolor">#E0E0E0</xsl:attribute>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:value-of select="." />
      </td>
   </xsl:template>

   <xsl:template match="log1">
      <table class="log1">
         <tr>
            <xsl:apply-templates />
         </tr>
      </table>
      <p/>
   </xsl:template>

   <xsl:template match="log2">
      <table class="log2">
         <tr>
            <xsl:apply-templates />
         </tr>
      </table>
      <p/>
   </xsl:template>

   <xsl:template match="testname">
      <tr>
         <xsl:if test="@id">
            <xsl:attribute name="id">
               <xsl:value-of select="@id" />
            </xsl:attribute>
         </xsl:if>
         <xsl:if test="@value">
            <xsl:attribute name="value">
               <xsl:value-of select="@value" />
            </xsl:attribute>
         </xsl:if>
         <xsl:choose>
            <xsl:when test="@value='not'">

               <xsl:attribute name="class">runonce</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>

               <xsl:attribute name="class">runboth</xsl:attribute>
            </xsl:otherwise>
         </xsl:choose>
      <th style="text-align:left; font-size:11px;"><xsl:value-of select="." /></th>
      </tr>
   </xsl:template>

</xsl:stylesheet>
```