**Contents**

[1 Introduction 2](#_Toc501364256)

[2 Testing and Revision 2](#_Toc501364257)

[2.1 General Implementation Procedure 2](#_Toc501364258)

[2.2 Implementation Example 3](#_Toc501364259)

[2.3 Test Results and Revision 4](#_Toc501364260)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 2017-11-18 | 0.1 | Document created | Gian Raphael Prinz |

# **Introduction**

This document mainly describes the test procedure applied to the «Markup Generator» and the results coming from it. In a second step, it gives a short overview about the design decisions made in the project.

# **Testing and Revision**

All tests implemented and applied to the «Markup Generator». are described within the document MARK-TEST-001. Thus, they are not listed again. Instead, the test procedure and a single implementation example will be described in the following.

## **General Implementation Procedure**

The test classes were implemented by using «AutoTest», a tool within EiffelStudio that allows creating, managing and executing software tests. Multiple test classes consisting of a test set were introduced for structuring the test process. The tests were written manually and not extracted or autogenerated. The anatomy of a test class looks always as follows:

|  |
| --- |
| note  description: "[  Eiffel tests that can be executed by testing tool.  ]"  author: "EiffelStudio test wizard"  date: "$Date$"  revision: "$Revision$"  testing: "type/manual"  class  MY\_TEST\_CLASS  inherit  EQA\_TEST\_SET  feature -- Test routines  my\_test  -- New test routine  do  assert ("not\_implemented", False)  end  end |

The example class MY\_TEST\_CLASS from above inherits from EQA\_TEST\_SET where all test classes have to inherit from. The feature my\_test is a procedure that is part of MY\_TEST\_CLASS accepting no arguments and is thus qualified as single test.

Executing a test can result in three possibilities:

1. The test is successful
2. The test is failing
3. The test result is unresolved

The outcome of a test is displayed in EiffelStudio. If necessary, measures can be smade.

## **Implementation Example**

The following test was described in the test suite. It is about creating documents, the frame of Markup report.

Table 1: Test case about the creation of documents

|  |  |
| --- | --- |
| Test Case ID | T.3.1.1.003 |
| SRS ID of Functionality | R3.2.1.003 |
| Name of Functionality | Creating Documents |
| Classes and Routines | Class: Document, routine: make |
| Set Up | An instance of Document class should be declared. |
| Tear Down | None |
| Test Data | Internal name |
| Oracle | A document object should be created. It should have an internal name that was determined within the creation procedure. No messages should be displayed. Furthermore, no errors should be raised. There is no item or value returned. |

It states that an instance of DOCUMENT should be created consisting of an internal name specified during the creation procedure. Furthermore, no errors or messages should be displayed. It is also not allowed for items or values to be returned. Based on this test specifications and the anatomy of a test class, the test class DOCUMENT\_TEST was implemented.

|  |
| --- |
| note  description: "[  Eiffel tests that can be executed by testing tool.  ]"  author: "EiffelStudio test wizard"  date: "$Date$"  revision: "$Revision$"  testing: "type/manual"  class  DOCUMENT\_TEST  inherit  EQA\_TEST\_SET  create  default\_create  feature {NONE} -- Well known field values  Name: STRING\_8 = "DOCUMENT-A"  Name2: STRING\_8 = ""    feature -- Test routines  document\_test  -- New test routine  local  document: DOCUMENT  document2: DOCUMENT  do  create document.make (Name)  create document2.make (Name2)  assert ("Document name", document.getname.is\_equal (Name))  assert ("Document name", document2.getname.is\_equal (Name2))  end    end -- class DOCUMENT\_TEST |

The test class DOCUMENT\_TEST consists of a single test, called document\_test. It creates two objects of type DOCUMENT, one with a meaningful name and another with an empty string as name. At the end, the names assigned to the objects are validated by comparing them with the string values used during the construction procedure.

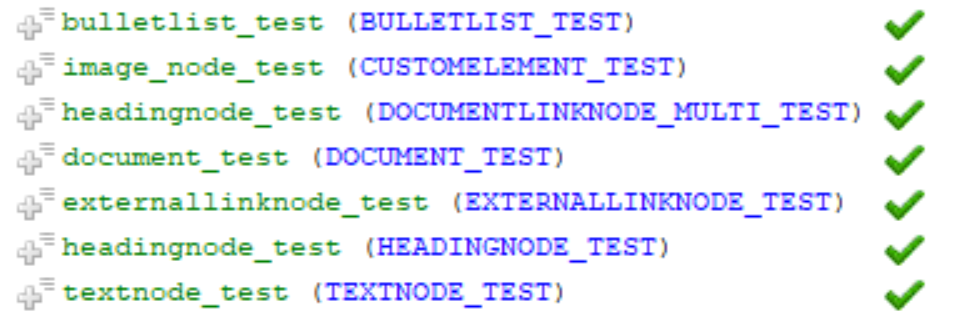


Figure 1: Outcome of the test execution in EiffelStudio

As shown in figure 1, the test document\_test was run successfully. This is indicated by the green hook besides the test. Thus, no revision seems to be necessary.

All other test cases described in document MARK-TEST-001 were implemented analogously.

## **Test Results and Revision**

SOME TEXT HERE…..

# **Design Decisions**

There is no one answer for devising software projects. Devise an (informal) document were you document and justify all your design decisions (this includes design patterns you used, you can gladly take over text from your design documents)