Klaros-Testmanagement Tutorial

Version 4.12.7

Publication date April 17 2020

Klaros-Testmanagement Tutorial

by Sabrina Gidley, Fabian Klaffke, Klaus Mandola, Patrick Reilly, Tobias Schmitt, and Torsten Stolpmann

Version 4.12.7

Publication date April 17 2020
Copyright © 2009-2020 verit Informationssysteme GmbH

Abstract

This document serves as a tutorial for the Klaros-Testmanagement application. It gives an example tour through the application and the provided functionality.

Table of Contents

List of Figures

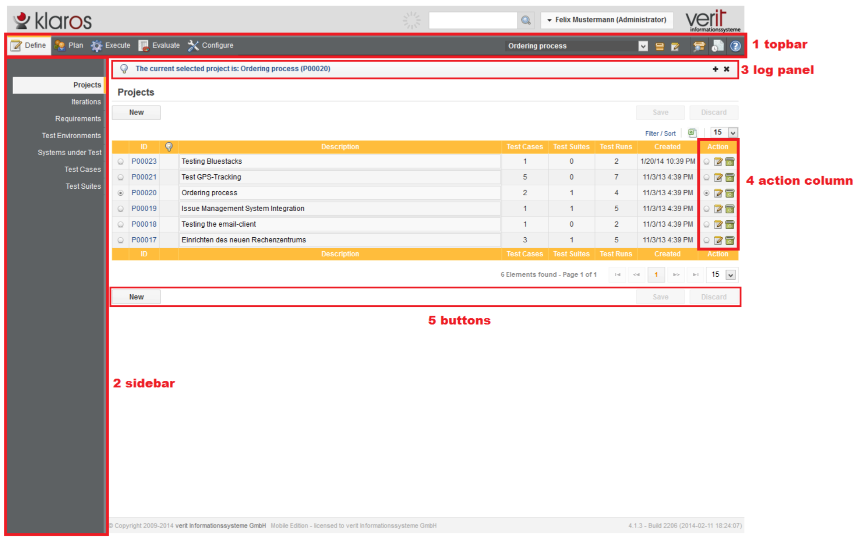
List of Tables

Chapter 1. Introduction

Klaros-Testmanagement is an easy to use web based application which helps to organize test projects. It manages test cases, test suites, information about systems under test and test environments in which tests have been run. When a test case or test suite has been executed, its result, including the information about the system under test and its test environment, is stored in a database. This enables full traceability of all test runs. Multiple reports enable a detailed overview of the progress of the project at any time.

1.1. Navigation

Throughout this tutorial you will find instructions for navigating through Klaros-Testmanagement. Figure 1.1 shows a typical page with the navigation areas numbered.



The current selected project is: Ordering process (P00020)

Projects

	ID		Description	Test Cases	Test Suites	Test Runs	Created	Action
	P00023		Testing Bluestacks	1	0	2	1/20/14 10:39 PM	
	P00021		Test GPS-Tracking	5	0	7	11/3/13 4:39 PM	
	P00020		Ordering process	2	1	4	11/3/13 4:39 PM	
	P00019		Issue Management System Integration	1	1	5	11/3/13 4:39 PM	
	P00018		Testing the email-client	1	0	2	11/3/13 4:39 PM	
	P00017		Einrichten des neuen Rechenzentrums	3	1	5	11/3/13 4:39 PM	

6 Elements found - Page 1 of 1

© Copyright 2009-2014 verit Informationssysteme GmbH Mobile Edition - licensed to verit Informationssysteme GmbH 4.1.3 - Build 2296 (2014-02-11 18:24:07)

Figure 1.1. Navigation in Klaros-Testmanagement

The User Interface of Klaros-Testmanagement

1. The topbar. This can be used to navigate between the main sections of Klaros-Testmanagement.

2. The sidebar contains the menu entries. If you select a specific section via the topbar, the menu entries for this section are displayed in the sidebar.

3. The log panel displays various information, warnings and error messages.

4. Most tables within Klaros-Testmanagement have an action column like this one. The icons in this column execute actions for the object in the row, e.g. in this case, pressing the 📝 icon of the project P00020 will open the details page of the *Ordering Process* project.

5. Many pages in Klaros-Testmanagement have buttons like this to perform actions like creating new objects, or saving or discarding changes. These buttons are located at the bottom of the screen to the left or right hand side.

Chapter 2. Quick Start Guide

This section of the tutorial covers the creation of a project, several test cases and a test suite. In addition, the test cases and the test suite will be executed and as a last step, test reports will be generated from the test results. The example project in this guide involves both hardware and software components. For this example, we will consider a printer as our system under test.

Our printer has both hardware and software components which must be tested, as well as various versions of each to be tested in different environments. A good example here are the printer drivers, which have different versions for Windows and Linux, and must also be tested in different versions of each.

The different sorts of testing that must be done require different teams of testers. For our tutorial we will consider two different teams, the hardware team and the software team. The different teams have different issue management systems to track defects. The hardware team uses JIRA, while the software team uses Mantis.

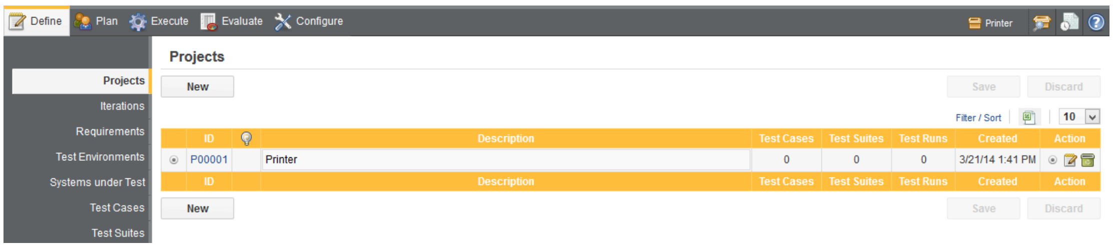
2.1. Setup

In order to execute and evaluate test cases, you will have to create the test cases itself and also all related test artifacts (e.g. the test environments in which the test cases will be executed) in Klaros-Testmanagement. In this chapter we will create all artifacts which are necessary for the execution of a single test case.

2.1.1. Creating a Project

Projects are the base test artifacts in Klaros-Testmanagement. All test-related artifacts, e.g. test cases or test results, are associated with a single project. A project must therefore be selected in order to engage in any test-related activities. When you first log in to Klaros-Testmanagement there will be no projects available so you will have to create one.

1. Press the New button to create a new project.



Projects

	ID		Description	Test Cases	Test Suites	Test Runs	Created	Action
●	P00001		Printer	0	0	0	3/21/14 1:41 PM	

Figure 2.1. The Projects page

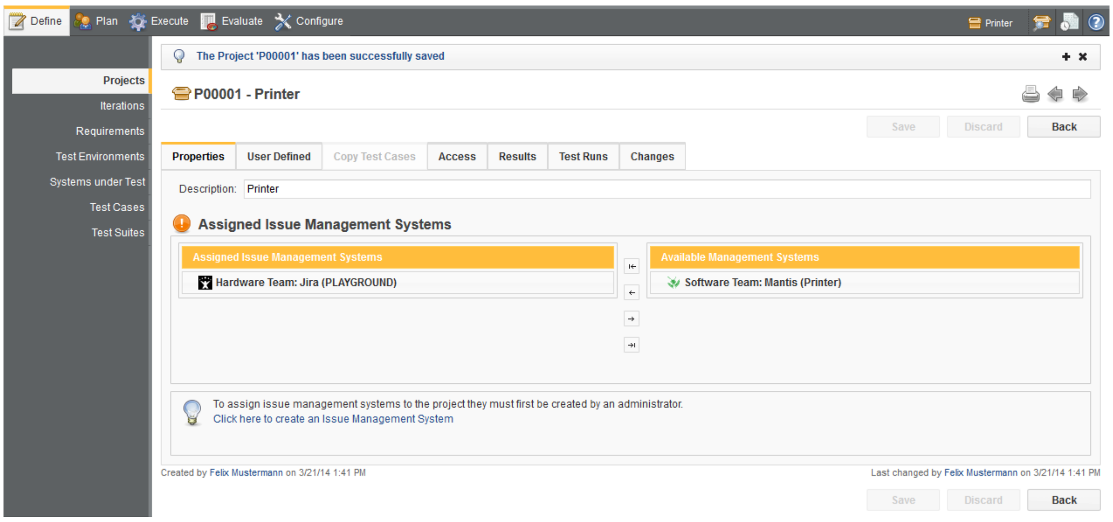
2. Enter **`Printer`** as the project description.

3. Press the Save button.

   Select the newly created project by clicking on the radio button next to its id. You will notice that more options have become available here in the **Define** section, and that some of the buttons in the topbar are no longer grayed out. For now however, we want to edit the project that we have just created.

2.1.1.1. Assigning Issue Management Systems

1. Press the icon to go to the *Project Details* page.
2. Click on the *Integration* and then on the *Issue Management* tab.
3. Select the issue management system you want to add by clicking on it, and then click on the arrow pointing left.



The Project 'P00001' has been successfully saved

P00001 - Printer

Description: Printer

Assigned Issue Management Systems

Hardware Team: Jira (PLAYGROUND)

Software Team: Mantis (Printer)

To assign issue management systems to the project they must first be created by an administrator.
Click here to create an Issue Management System

Created by Felix Mustermann on 3/21/14 1:41 PM

Last changed by Felix Mustermann on 3/21/14 1:41 PM

Figure 2.2. The Properties tab in the Project page

4. Press the Save button.

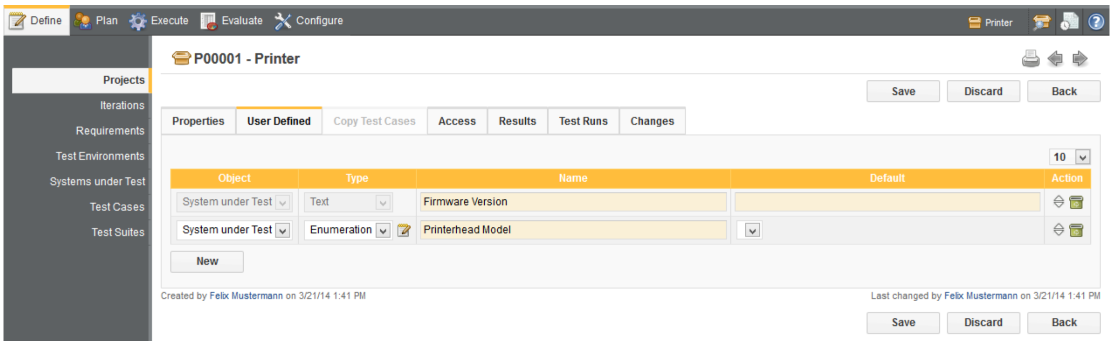
2.1.1.2. Creating User-Defined Properties

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

Now is a good time to create some user defined properties for the project to allow for more accurate searching of elements in the project. The process of creating user defined properties is explained in Section 3.3, “ User Defined Properties ”.

Since the SUTs in our example are printers, there are a number of properties that will make searching through them easier. One of these is the firmware version, an alpha-numeric value which is changed relatively often.

1. Select the *User Defined* tab.
2. Click the New button.



P00001 - Printer

Object	Type	Name	Default	Action
System under Test	Text	Firmware Version		
System under Test	Enumeration	Printerhead Model		

Created by Felix Mustermann on 3/21/14 1:41 PM

Last changed by Felix Mustermann on 3/21/14 1:41 PM

Figure 2.3. Adding User Defined Properties

3. Select *System Under Test* in the *Object* column.

4. Select *Text* in the *Type* column.

5. Enter `Firmware Version` in the *Name* text field.

6. Press the Save button.

7. A second property we can use to search through our SUTs is the model of printerhead used in the printer. Since our company has only two different printerhead models, it makes sense to make this an enumeration property.

8. Click the New button.

9. Select *System Under Test* in the *Object* column.

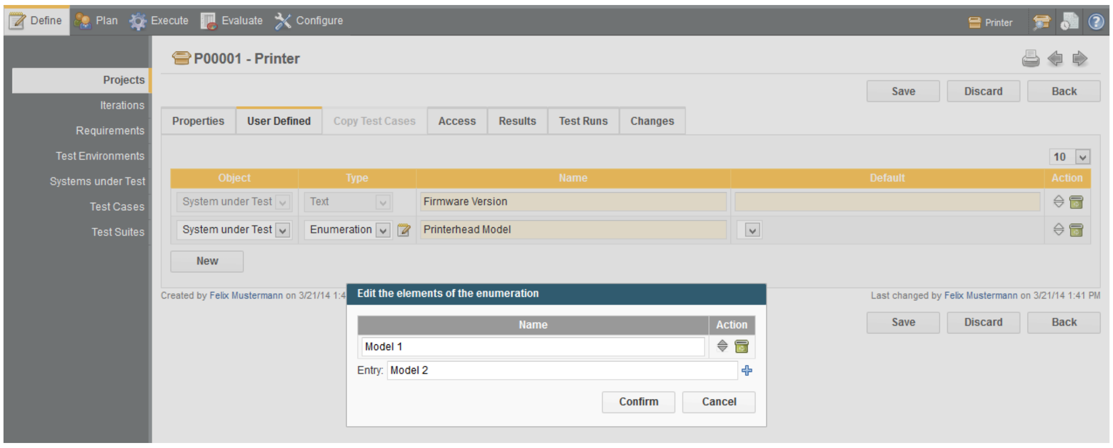
10. Select *Enumeration* ind the *Type*column.

11. Enter `Printerhead Model` in the *Name* text field.

12. Press the icon in the *Type* column to add enumeration values.

13. Enter `Model 1` in the *Entry* text field and press the

14. Enter `Model 2` in the *Entry* field and press the icon.

15. 

P00001 - Printer

Object	Type	Name	Default	Action
System under Test	Text	Firmware Version		
System under Test	Enumeration	Printerhead Model		

Edit the elements of the enumeration

Name	Action
Model 1	

Entry: Model 2

16. Press the Confirm button.

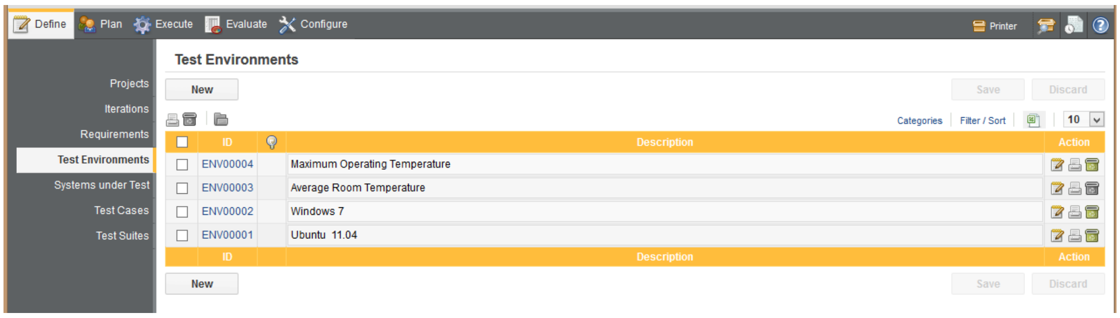
    Press the Save button.

2.1.2. Creating Test Environments

The next step in setting up a Klaros-Testmanagement project is to create some test environments to test in. In Klaros-Testmanagement, a test environment is a description of the extrinsic settings that may influence the test result. This could be the operating system used with the printer or even the physical environment used for the test.

For this tutorial, we will create two software-related and two hardware-related test environments.

1. Press the *Test Environments* link in the sidebar.
2. Press the `New` button.



ID	Description	Action
ENV00004	Maximum Operating Temperature	
ENV00003	Average Room Temperature	
ENV00002	Windows 7	
ENV00001	Ubuntu 11.04	

Figure 2.4. The Test Environment page

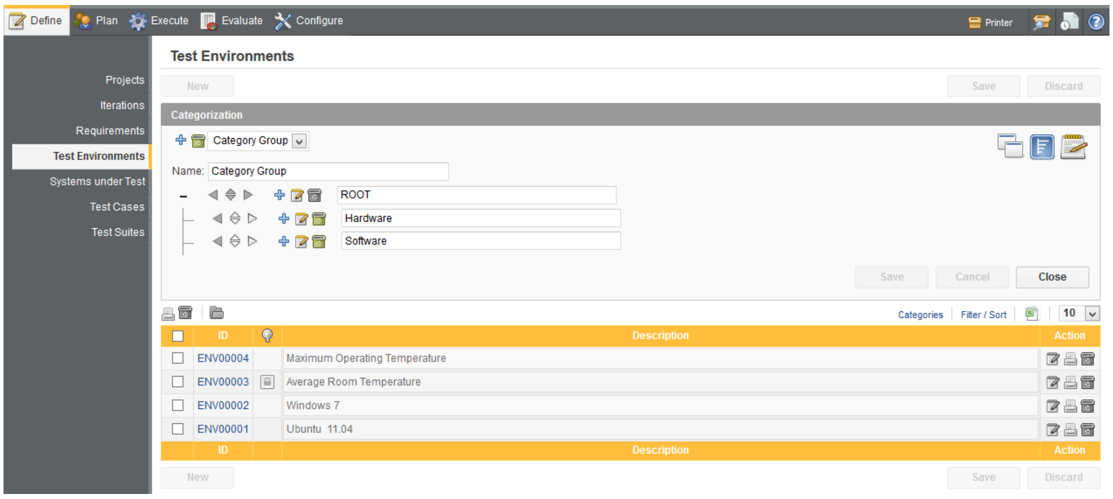
3. Enter `Ubuntu 11.4` in the *Description* text field.
4. Press the `Save` button.
5. Next, create three more Test Environments with the following values in the *Description* text field:
   - `Windows 7`
   - `Average Room temperature`
   - `Maximum Operating Temperature`
6. Press the `Save` button.
7. Press the `Save` button.

2.1.2.1. Assigning Test Environments to Categories

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

Now is a good time to use the categorization feature at this stage. See Section 3.2, “ Categorization ”.

1. Press the *Categories* label to expand the category creation and ???
2. Press the icon.
3. Press the icon to create a new category group.



	ID		Description	Action
☐	ENV00004		Maximum Operating Temperature	
☐	ENV00003		Average Room Temperature	
☐	ENV00002		Windows 7	
☐	ENV00001		Ubuntu 11.04	
	ID		Description	Action

Figure 2.5. Creating Categories

4. Press the ✚ icon on the newly created *ROOT* category (this is the root category of this category group).

5. Enter `Hardware` in the text field of the new category.

6. Press the ✚ icon on the *ROOT* category again.

7. Enter `Software` in the text field of the new category.

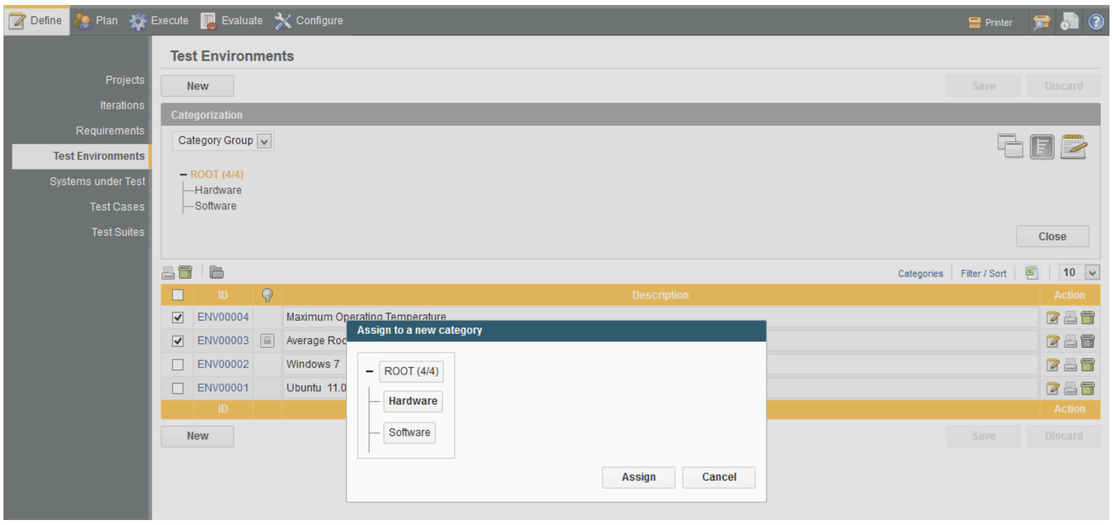
8. Press the Save button.

9. Switch to the tree view by clicking on the icon.

10. Select the checkboxes beside each of the software-related test environments (`Windows 7` and `Ubuntu 10.4`). .

11. Press the 📁 icon.

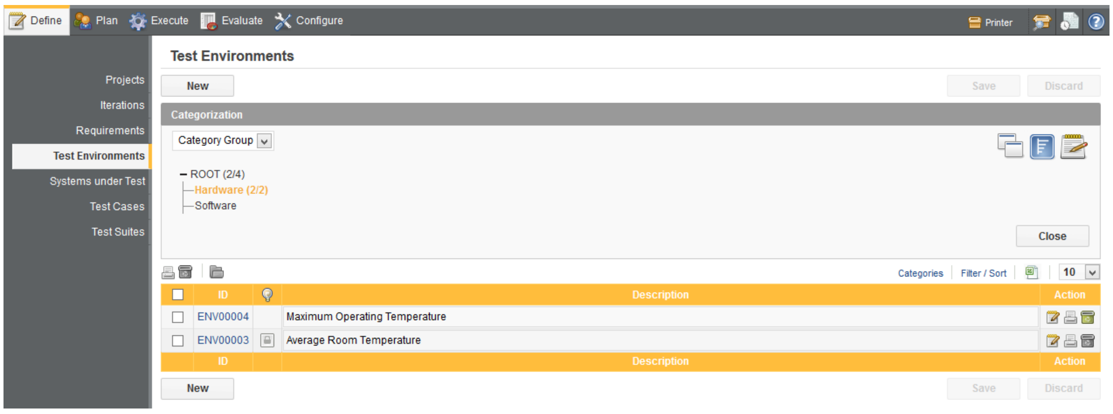
12. Press the `Software` entry in the popup.



	ID		Description	Action
☑	ENV00004		Maximum Operating Temperature	
☑	ENV00003		Average Roo	
☐	ENV00002		Windows 7	
☐	ENV00001		Ubuntu 11.0	
	ID			Action

Figure 2.6. Assigning Test Environments to Categories

13. Press the Assign button.
14. Select the checkbox beside each of the hardware-related test environments.
15. Press the icon.
16. Press the `Hardware` entry in the popup.
17. Press the Assign button.



ID		Description	Action
ENV00004		Maximum Operating Temperature	
ENV00003		Average Room Temperature	

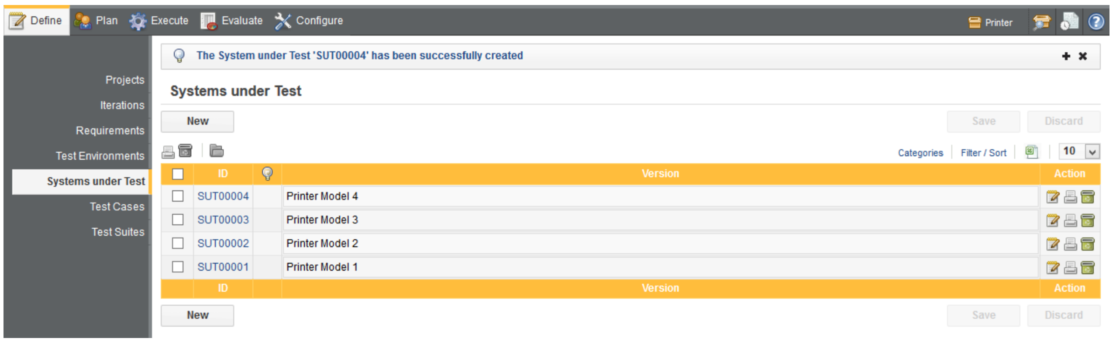
Figure 2.7. The Test Environment Categories

From now on, all test environments can be viewed as normal if the category panel is closed, and opening the category panel allows for selective viewing of the particular categories of test environments.

2.1.2.2. Creating Systems under Test

Next we should create some systems under test. In Klaros-Testmanagement, systems under test represent the versions of the product or software system to be tested. In our example, these will be various printers.

1. Press the **Systems under Test** menu entry in the sidebar.
2. Press the New button.



The System under Test 'SUT00004' has been successfully created

ID		Version	Action
SUT00004		Printer Model 4	
SUT00003		Printer Model 3	
SUT00002		Printer Model 2	
SUT00001		Printer Model 1	

Figure 2.8. The System under Test page

3. Enter the text `Printer Model 1` in the *Version* field.
4. Press the Save button.
5. Repeat the process for three more systems under test:

Versions

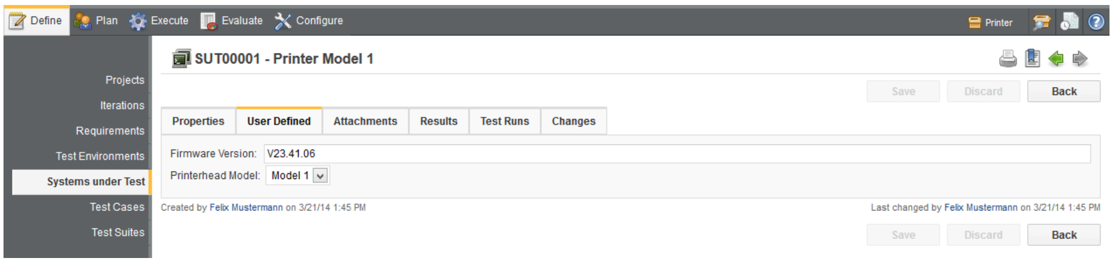
- `Printer Model 2`
- `Printer Model 3`
- `Printer Model 4`

2.1.2.2.1. Assigning User Defined Property Values

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

Please refer to Section 3.3, “ User Defined Properties ” how to create user defined properties and create a String property *Firmware Version* and an enumeration property *Printerhead Model* with the values *Model 1* and *Model 2* for systems under test. Then you can set values for them.

1. Press the icon for *Printer Model 1*.
2. Press the *User Defined* tab.
3. Select *Model 1* in the *Printerhead Model* dropdown list.
4. Enter `V23.41.06` in the *Firmware Version* field.
5. Press the Save button.



SUT00001 - Printer Model 1

Firmware Version: V23.41.06

Printerhead Model: Model 1

Created by Felix Mustermann on 3/21/14 1:45 PM

Last changed by Felix Mustermann on 3/21/14 1:45 PM

Figure 2.9. The System under Test page

6. Press the Save button.
7. Repeat the process for the other three systems under test:

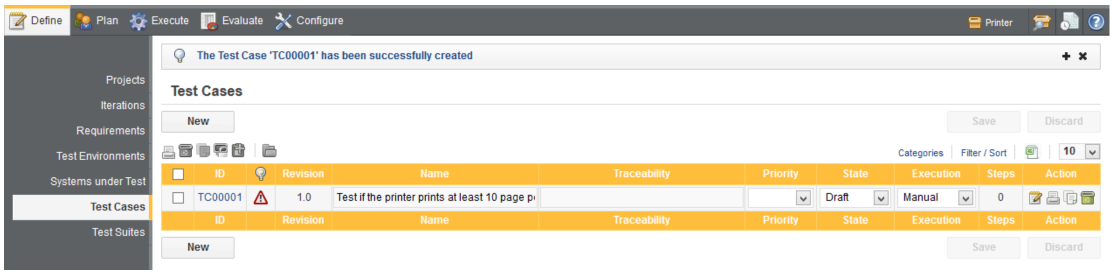
Version	Firmware Version	Printerhead Model
`Printer Model 2`	`V23.41.07B`	`Model 2`
`Printer Model 3`	`V23.41.07B`	`Model 1`
`Printer Model 4`	`V23.41.05`	`Model 2`

2.1.2.3. Creating Test Cases

Now it is time to create some test cases. In Klaros-Testmanagement, test cases represent individual tests to be carried out. We will create test cases for each of the hardware and software teams. First we will create a few hardware test cases. The first test is a verification that the printer upholds our claims about its minimum pages per minute rate.

1. Press the **Test Cases** menu entry in the sidebar.

2. Press the `New` button.



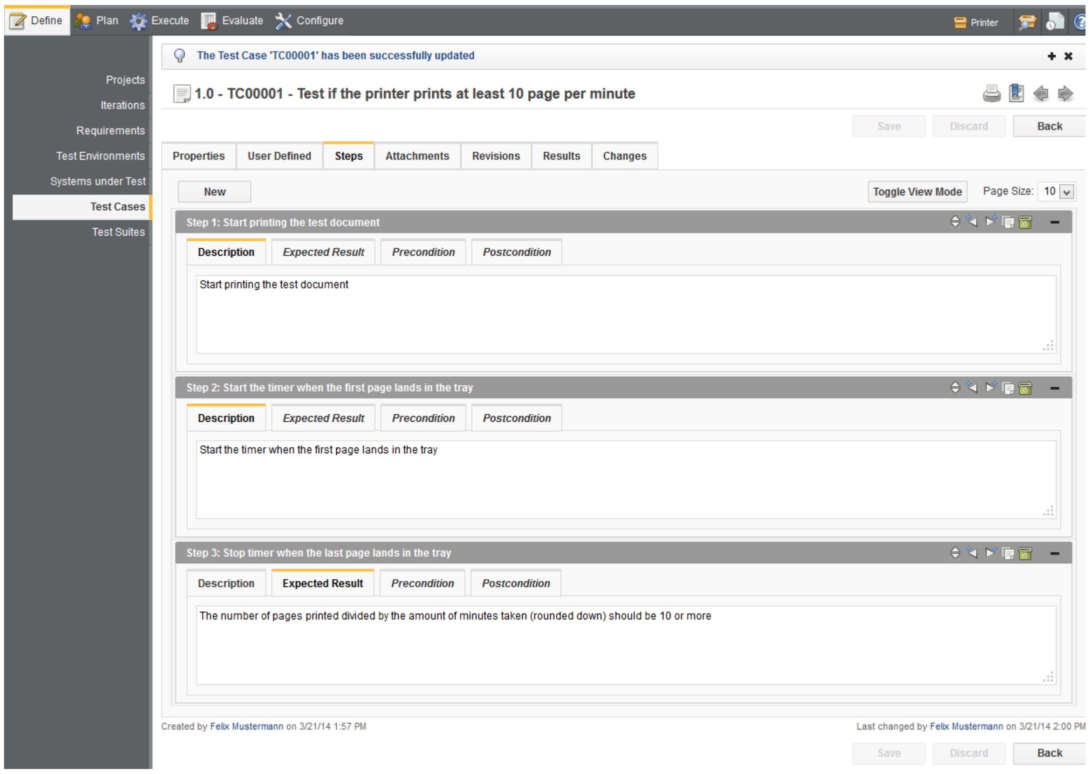
The Test Case 'TC00001' has been successfully created

Test Cases

ID	Revision	Name	Traceability	Priority	State	Execution	Steps	Action
TC00001	1.0	Test if the printer prints at least 10 page p			Draft	Manual	0	

Figure 2.10. The Test Cases page

3. Enter `Test if the printer prints at least 10 page per minute` in the *Name* field.
4. Select *Manual* in the *Execution* field.
5. Press the `Save` button.
6. Press the edit icon on the newly created test case.
7. Select the *Steps* tab.



The Test Case 'TC00001' has been successfully updated

1.0 - TC00001 - Test if the printer prints at least 10 page per minute

Step 1: Start printing the test document

Start printing the test document

Step 2: Start the timer when the first page lands in the tray

Start the timer when the first page lands in the tray

Step 3: Stop timer when the last page lands in the tray

The number of pages printed divided by the amount of minutes taken (rounded down) should be 10 or more

Created by Felix Mustermann on 3/21/14 1:57 PM

Last changed by Felix Mustermann on 3/21/14 2:00 PM

Figure 2.11. The Test Case Step tab in the Test Cases page

8. Create the following Steps:

Description	Expected Result
`Start printing the test document`	
`Start the timer when the first page lands in the tray`	

Description	Expected Result
`Stop timer when the last page lands in the tray`	

9. Press the `Save` button.

10. Next, create the following test cases (each with one test case step):

Name	Execution	Step Description	Step Expected Result
`Prints test page`	*Manual*	`Print the test page using the OS UI`	`Page prints`
`Detects empty`	*Manuel*	`Insert empty`	`Printer display shows "Cartridge empty" warning`

Now you should have three test cases.

2.1.2.4. Creating Test Suites

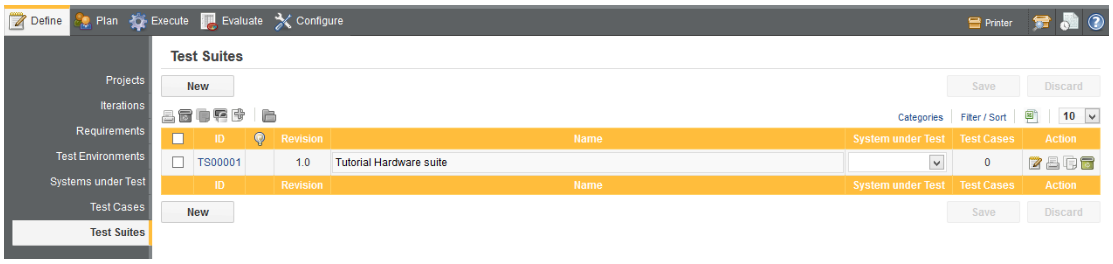
The final step in preparation for testing with Klaros-Testmanagement is the creation of test suites. In Klaros-Testmanagement, test suites represent groups of tests to be carried out in sequence. A test suite contains one or more test cases and can be executed as a unit, which allows test results to be grouped together.

Note

A test suite can contain the same test case multiple times.

1. Press the *Test Suites* link in the sidebar.

2. Press the `New` button.



Test Suites

	ID		Revision	Name	System under Test	Test Cases	Action
☐	TS00001		1.0	Tutorial Hardware suite		0	

Figure 2.12. The Test Suites page

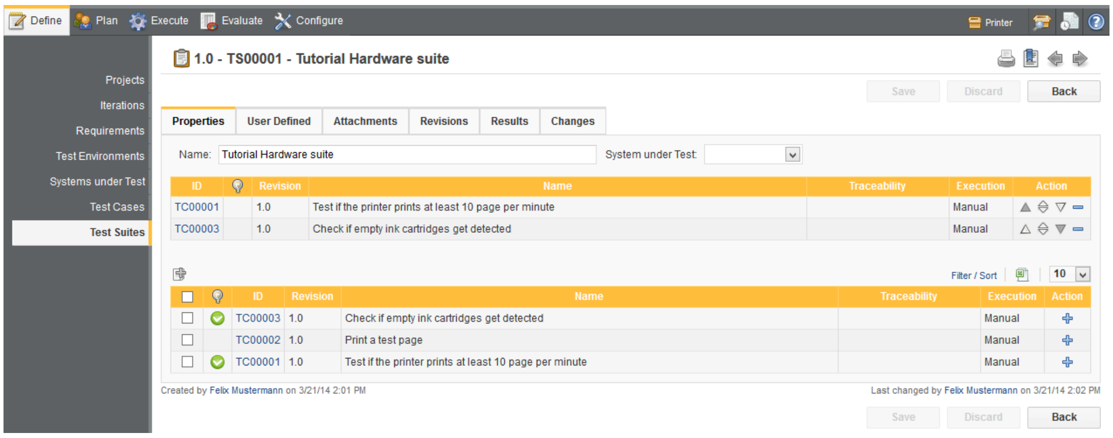
3. Enter `Tutorial Hardware Suite` in the *Name* field.

4. Press the `Save` button.

5. Press the icon.

6. Click on the *Properties* tab.

7. Press the icon on the `Detects empty` and the `Test if the printer prints at least 10 pages per minute` test cases.



1.0 - TS00001 - Tutorial Hardware suite

Name: Tutorial Hardware suite   System under Test:

ID		Revision	Name	Traceability	Execution	Action
TC00001		1.0	Test if the printer prints at least 10 page per minute		Manual	
TC00003		1.0	Check if empty ink cartridges get detected		Manual	

		ID	Revision	Name	Traceability	Execution	Action
☐	✓	TC00003	1.0	Check if empty ink cartridges get detected		Manual	+
☐		TC00002	1.0	Print a test page		Manual	+
☐	✓	TC00001	1.0	Test if the printer prints at least 10 page per minute		Manual	+

Created by Felix Mustermann on 3/21/14 2:01 PM   Last changed by Felix Mustermann on 3/21/14 2:02 PM

Figure 2.13. The Test Suite page

8. Press the Save button

Now our project is set up and we can begin testing!

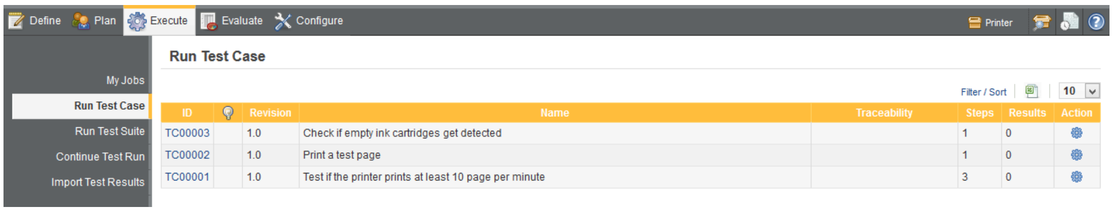
2.1.3. Executing Tests

Now that we have created some test cases and a test suite, we are ready for some testing. Log out of Klaros-Testmanagement and log in as a tester. This is a more limited account which can execute tests and only data linked to the account.

2.1.3.1. Executing Single Test Cases

In this section, we will execute a single test case multiple times with different parameters.

1. If not already selected, select the `Printer` project.
2. Press *Execute* in the topbar.
3. Click on *Run TestCase* in the sidebar.
4. The *Run Test Case* screen should now be shown. This screen displays the available test cases along with an execute icon (⚙) for each. The icon is blue if the test case can be executed in Klaros-Testmanagement, i.e. if *Execution* is set to *Manual*, if the test case has at least one step and its *State* is either *Draft* or *Approved*.



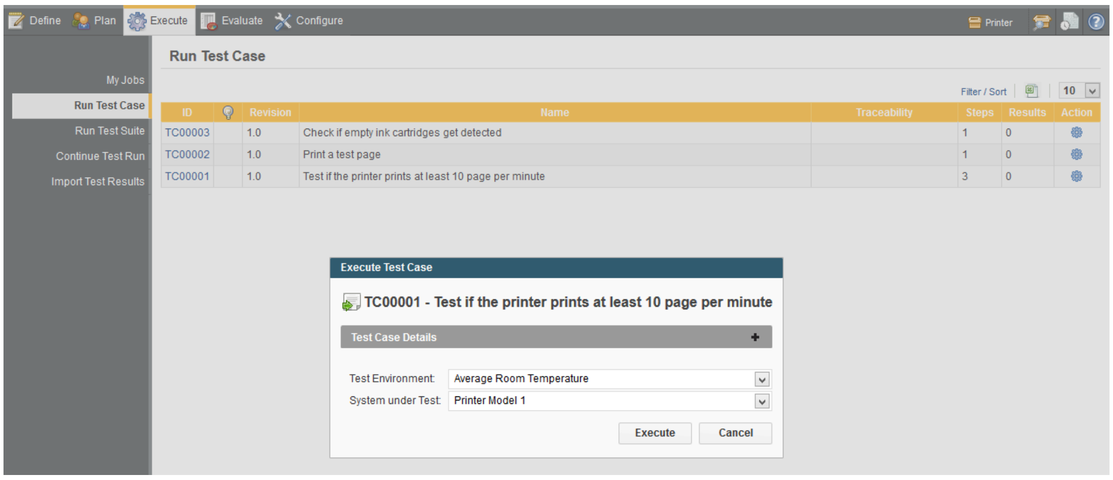
Run Test Case

ID		Revision	Name	Traceability	Steps	Results	Action
TC00003		1.0	Check if empty ink cartridges get detected		1	0	⚙
TC00002		1.0	Print a test page		1	0	⚙
TC00001		1.0	Test if the printer prints at least 10 page per minute		3	0	⚙

Figure 2.14. The Run TestCase page

   Press the ⚙ icon on the `Test if the printer prints at least 10 pages per minute` test case

5. On the next screen, select `Average Room Temperature` as the *Test Environment*



Run Test Case

ID		Revision	Name	Traceability	Steps	Results	Action
TC00003		1.0	Check if empty ink cartridges get detected		1	0	
TC00002		1.0	Print a test page		1	0	
TC00001		1.0	Test if the printer prints at least 10 page per minute		3	0	

Execute Test Case

TC00001 - Test if the printer prints at least 10 page per minute

Test Case Details

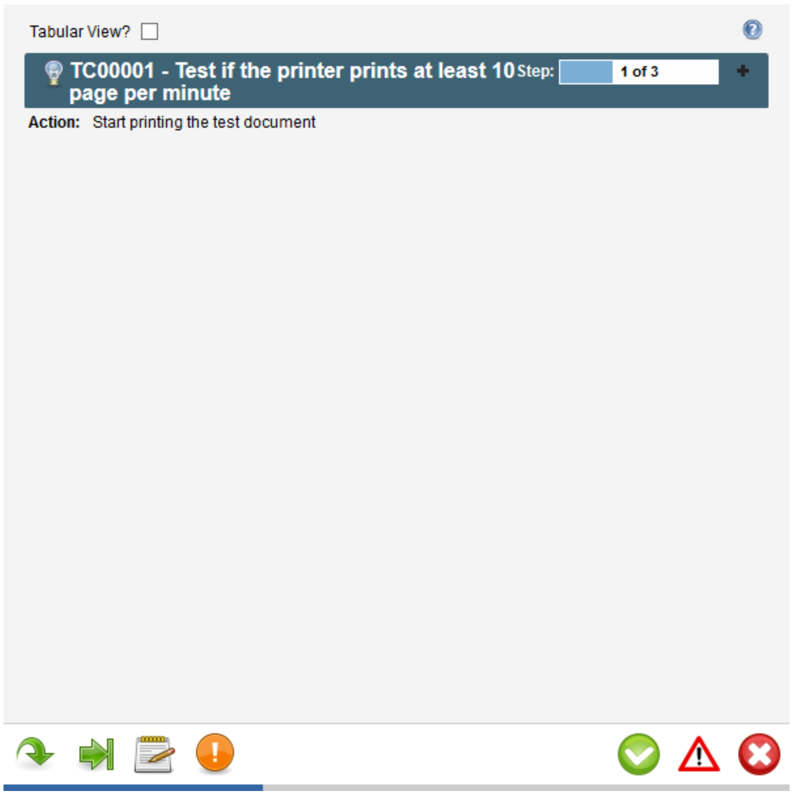
Test Environment:	Average Room Temperature
System under Test:	Printer Model 1

Figure 2.15. The Run Test Case page

6. Select `Printer Model 1` as the *System Under Test*.

7. Press the Execute button.

8. A popup will appear (if it doesn't, be sure to disable the popup blocker in your browser or add the Klaros-Testmanagement address to its exceptions). This is the manual test runner, which guides testers through the testing process. You will see an overview of the test case on the first page of the manual test runner.



Tabular View?

TC00001 - Test if the printer prints at least 10 page per minute Step: 1 of 3

Action: Start printing the test document

Figure 2.16. The Test Case Runner

9. The manual test runner will now show the first step. *Action* shows what we as a tester are supposed to do. Let's assume our printer will start printing the test document with no problems, so the first step is passed.

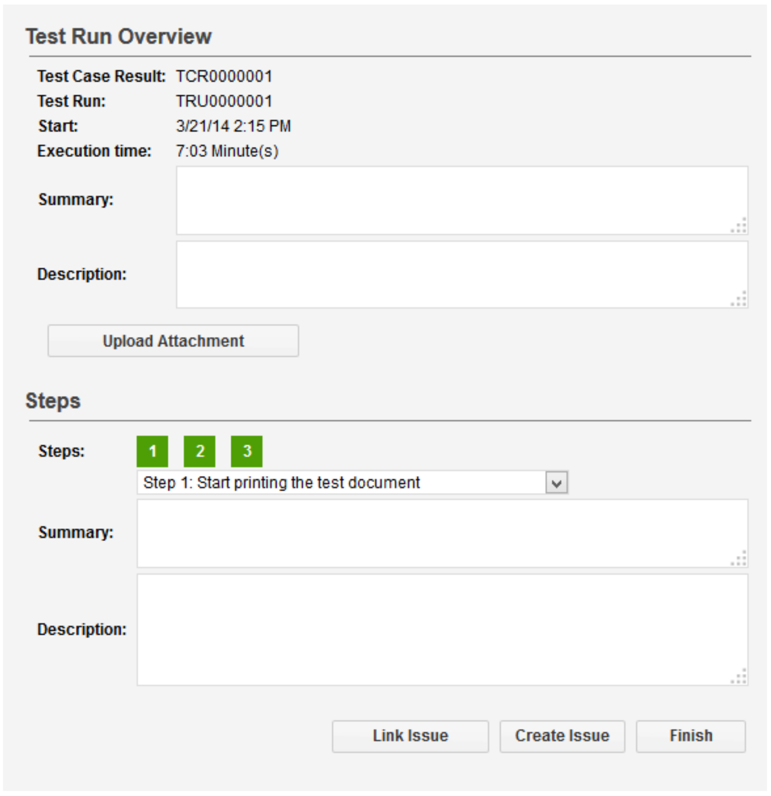
   Press the `Passed` button.

10. The second step will now be shown. Mark the rest of the steps as passed.

11. Now the *Test Run* overview will be displayed. Since all steps were passed without complication, there is no need to comment or create an issue.

    Please confirm the following dialog with `Yes`.

    Press the `Finish` button.



Test Run Overview

Test Case Result: TCR0000001
Test Run: TRU0000001
Start: 3/21/14 2:15 PM
Execution time: 7:03 Minute(s)
Summary:
Description:

Steps

Steps: 1 2 3
Step 1: Start printing the test document
Summary:
Description:

Figure 2.17. The Results Overview in the Manual Test Runner

12. Press **Run Test Case** in the sidebar.

13. Now we will execute the same test case using a different system under test and test environment.

    Press the ✚ icon on the `Test if the printer prints at least 10 pages per minute` test case.

14. On the next screen, select *Maximum Operating Temperature* as the *Test Environment*.

15. Select *Printer Model 2* as the *System Under Test*.

16. Press the `Execute` button.

17. Press the `Start` button.

18. Press the `Passed` button.

19. Press the `Passed` button.

20. Printer Model 2 uses the Model 2 printerheads, which have an overheating problem, so this test step will fail in an environment with a high ambient temperature.

    Press the `Failure` button.

21. Enter **`Too many pause cycles due to overheating`** in the *Summary* field.

    These comments will help reproducing the failure.

22. Press the `Finish` button.

23. If you have set up an issue management system, you can create or link an issue from this screen. The process is described in Section 3.6, “ Issues ”.

24. Confirm the following dialog by clicking the `Yes` button.

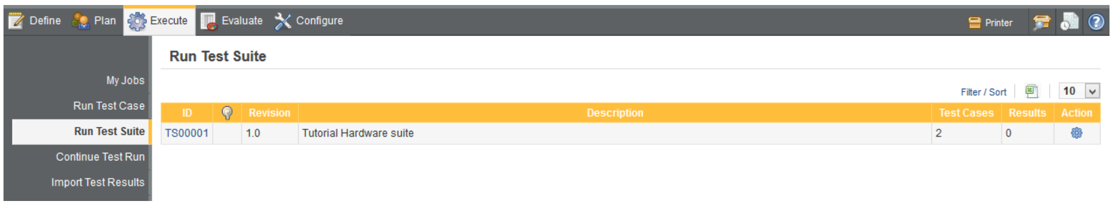
25. Press the `Finish` button.

2.1.3.2. Executing Test Suites

Next we will execute our test suite. The process is very similar to executing a test case, with an additional overview screen at the beginning.

1. Press **Run Test Suite** in the sidebar.

2. Press the ⚙ icon for *Tutorial Hardware suite*.

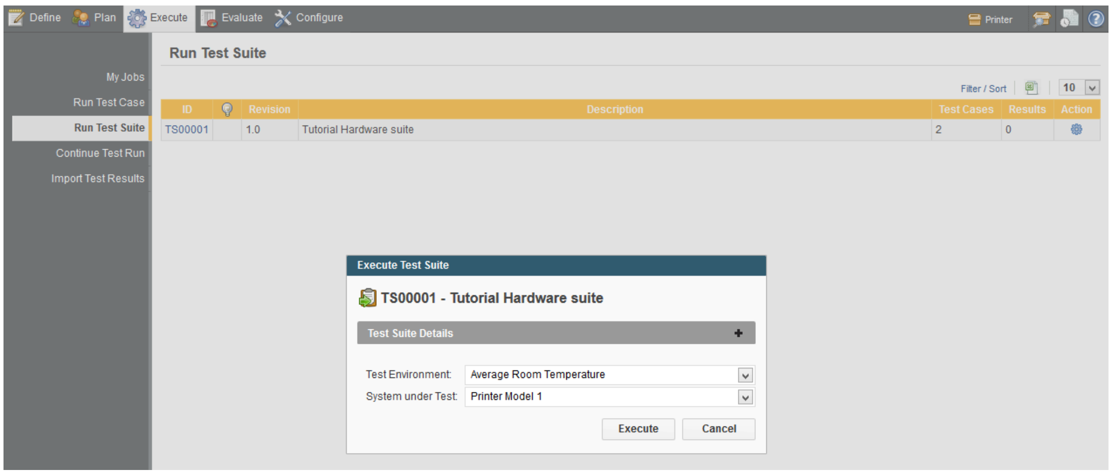


Run Test Suite

ID		Revision	Description	Test Cases	Results	Action
TS00001		1.0	Tutorial Hardware suite	2	0	⚙

Figure 2.18. The Run Test Suite page

3. Select **`Average Room Temperature`** as the *Test Environment*



Run Test Suite

ID		Revision	Description	Test Cases	Results	Action
TS00001		1.0	Tutorial Hardware suite	2	0	⚙

Execute Test Suite

TS00001 - Tutorial Hardware suite

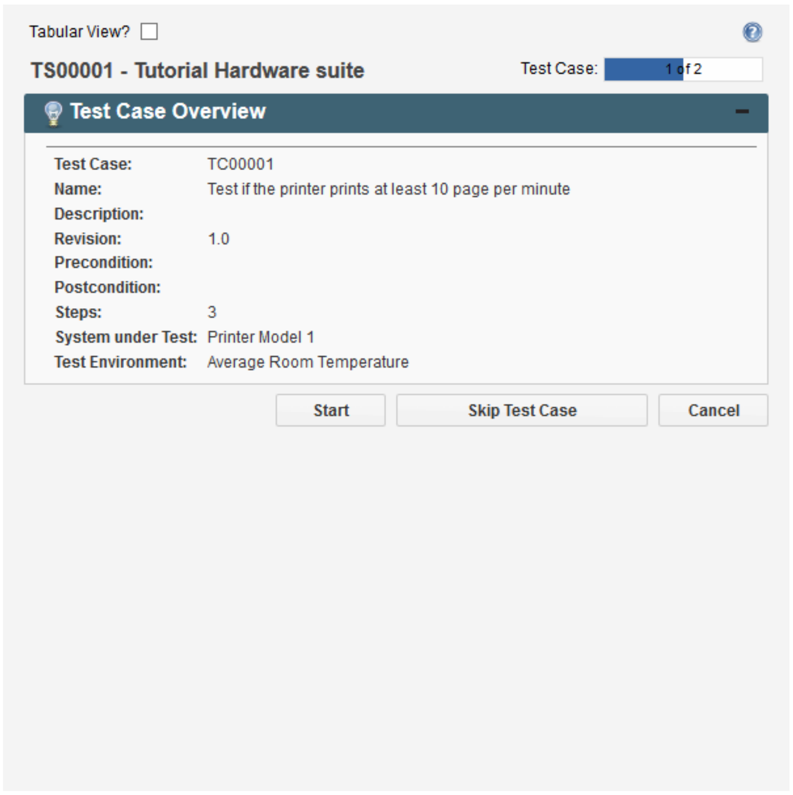
Test Suite Details

Test Environment: Average Room Temperature

System under Test: Printer Model 1

Figure 2.19. The Run Test Suite page

4. Select *Printer Model 1* as the *System under Test*.
5. Press the Execute button.
6. Press the Start button in the manual test runner.



TS00001 - Tutorial Hardware suite

Test Case Overview

Test Case:	TC00001
Name:	Test if the printer prints at least 10 page per minute
Description:	
Revision:	1.0
Precondition:	
Postcondition:	
Steps:	3
System under Test:	Printer Model 1
Test Environment:	Average Room Temperature

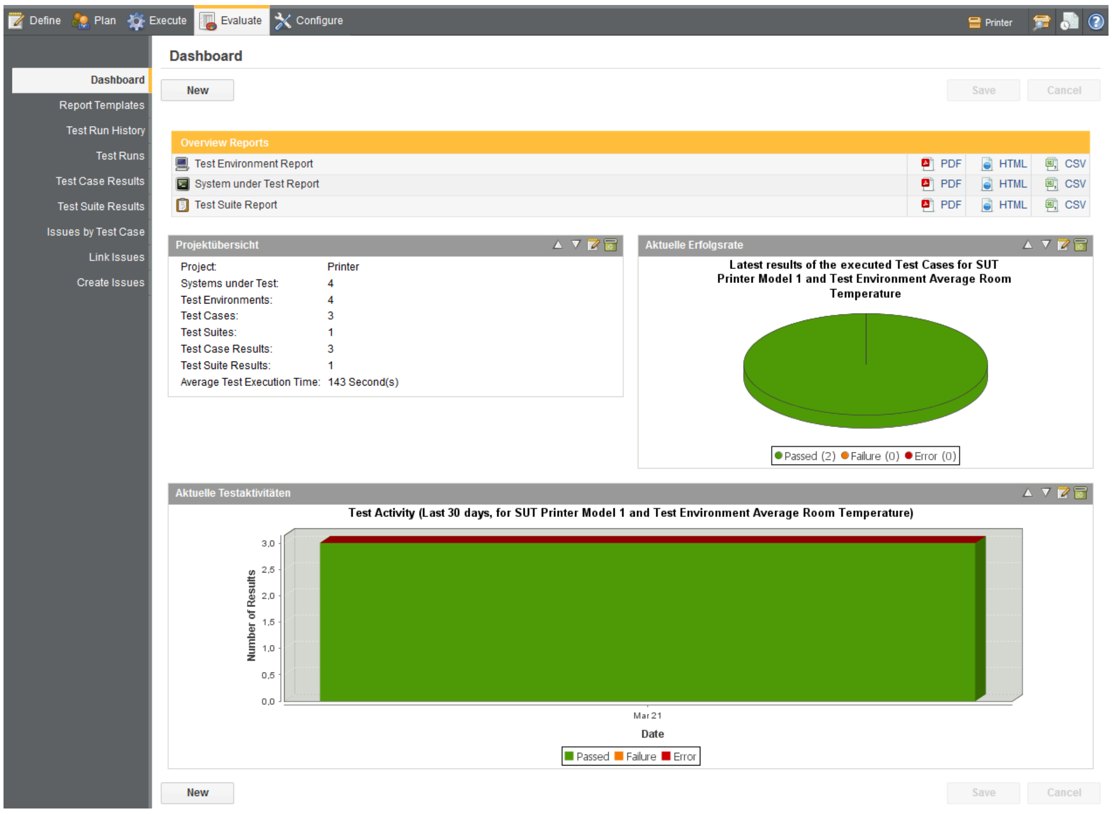
Figure 2.20. The test suite runner

7. Press the *passed* icon for all steps.
8. Confirm the following dialog by clicking the Yes button.
9. Repeat for the second test case.
10. Press the Finish button.

2.1.4. Results and Reports

Now it's time to view the test results and reports. Klaros-Testmanagement contains a feature-rich test evaluation section which is best viewed using a manager account.

1. Log into Klaros-Testmanagement using a manager account.
2. Select the *Printer* project if it is not already selected.
3. Press **Evaluate** in the topbar.
4. Now the dashboard should be displayed, showing some of the default reports for the *Printer* project. These show relevant details like the number of test cases and test suites in the Project, the overall numbers of passed and failed test runs, and the testing activity in the last 30 days.



Dashboard

Overview Reports			
Test Environment Report	PDF	HTML	CSV
System under Test Report	PDF	HTML	CSV
Test Suite Report	PDF	HTML	CSV

Projektübersicht

Project:	Printer
Systems under Test:	4
Test Environments:	4
Test Cases:	3
Test Suites:	1
Test Case Results:	3
Test Suite Results:	1
Average Test Execution Time:	143 Second(s)

Aktuelle Erfolgsrate

Latest results of the executed Test Cases for SUT Printer Model 1 and Test Environment Average Room Temperature

● Passed (2) ● Failure (0) ● Error (0)

Aktuelle Testaktivitäten

Test Activity (Last 30 days, for SUT Printer Model 1 and Test Environment Average Room Temperature)

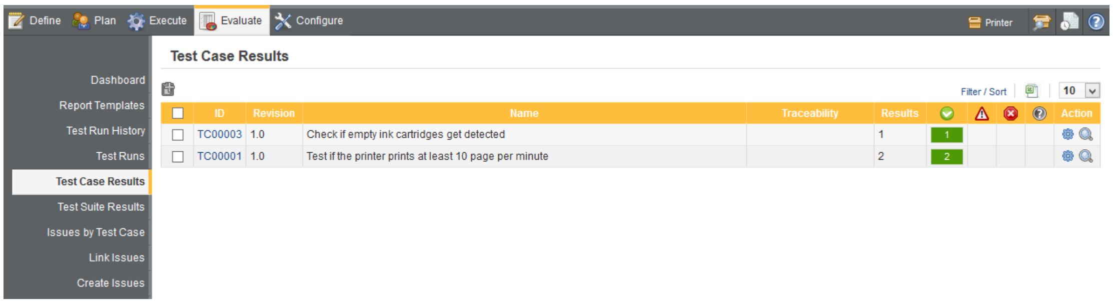
Figure 2.21. The Dashboard

5. The Latest Success Rate report can be changed by clicking on the Icon. Then you can select the test system and the test environment to be displayed.

6. You can also view individual test results for test cases and test suites in the **Evaluate** section.

   Press the **Test Case Results** menu entry in the sidebar.

7. Here you will see the test cases which have been executed in this project, along with a count of test runs with their results.



Test Case Results

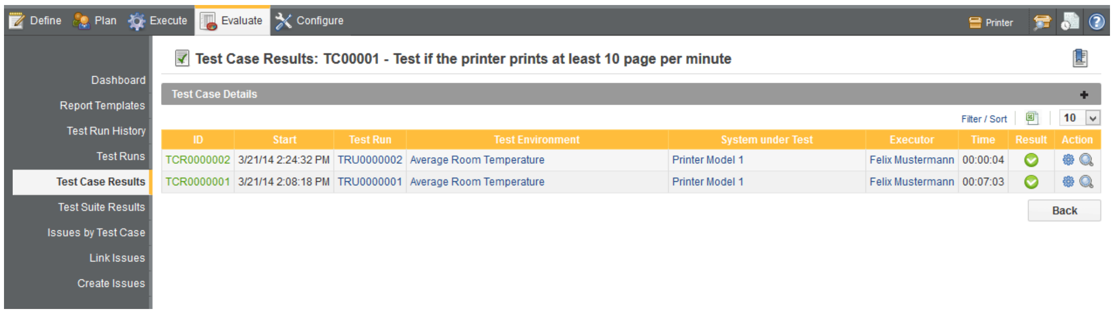
	ID	Revision	Name	Traceability	Results	✓	⚠	✗	?	Action
☐	TC00003	1.0	Check if empty ink cartridges get detected		1	1				
☐	TC00001	1.0	Test if the printer prints at least 10 page per minute		2	2				

Figure 2.22. The Test Case Results page

8. Press the icon for the *Test if the printer prints at least 10 pages per minute* test case.

9. You will now see a screen summarizing each of the test runs for this test case.

   Press the icon for one of the test runs.



Test Case Results: TC00001 - Test if the printer prints at least 10 page per minute

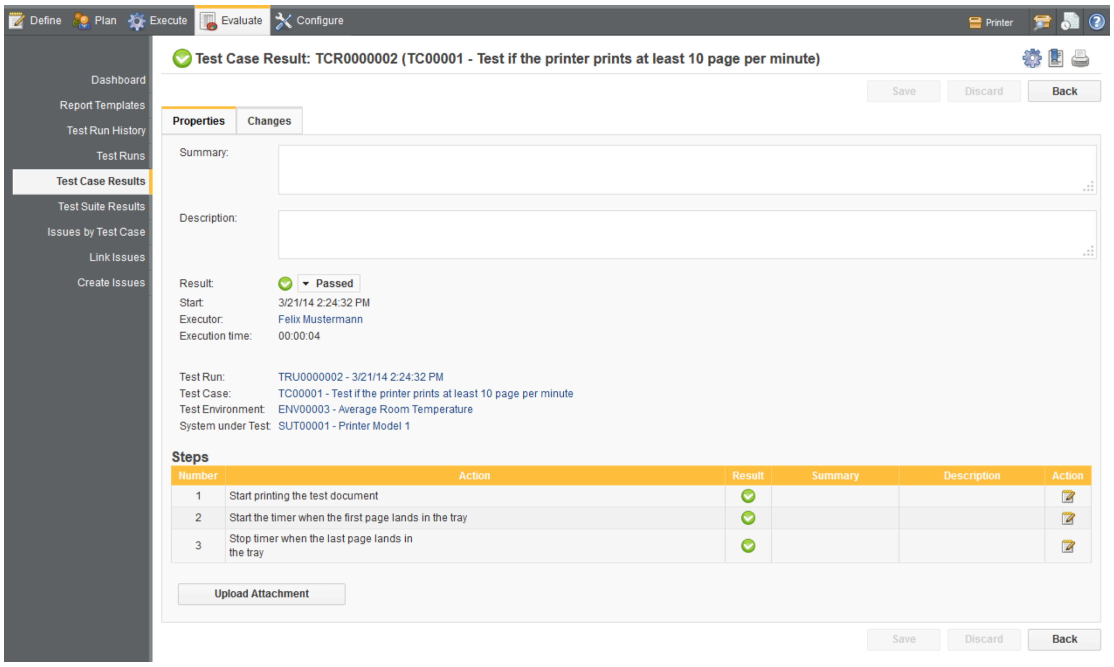
Test Case Details

ID	Start	Test Run	Test Environment	System under Test	Executor	Time	Result	Action
TCR0000002	3/21/14 2:24:32 PM	TRU0000002	Average Room Temperature	Printer Model 1	Felix Mustermann	00:00:04	✓	
TCR0000001	3/21/14 2:08:18 PM	TRU0000001	Average Room Temperature	Printer Model 1	Felix Mustermann	00:07:03	✓	

Figure 2.23. The Test Case Result page

10. This screen shows a breakdown of the results of each step in the test case, as well as the summary and description the tester has entered for each of them.

    Press *Test Results - Test Suite Results* in the sidebar.



Test Case Result: TCR0000002 (TC00001 - Test if the printer prints at least 10 page per minute)

Summary:	
Description:	
Result:	Passed
Start:	3/21/14 2:24:32 PM
Executor:	Felix Mustermann
Execution time:	00:00:04
Test Run:	TRU0000002 - 3/21/14 2:24:32 PM
Test Case:	TC00001 - Test if the printer prints at least 10 page per minute
Test Environment:	ENV00003 - Average Room Temperature
System under Test:	SUT00001 - Printer Model 1

Steps

Number	Action	Result	Summary	Description	Action
1	Start printing the test document	✓			
2	Start the timer when the first page lands in the tray	✓			
3	Stop timer when the last page lands in the tray	✓			

Figure 2.24. The Test Run page

This page shows results of all test cases which have been run in a similar style to the *Test Case Results* page. Have a look at the results of the *Tutorial Hardware suite*.

This rounds up our quick start guide. The following chapter contains in-depth how-tos for most of the available actions in Klaros-Testmanagement.

Chapter 3. How-Tos

This section of the tutorial consists of how-to guides for certain actions in Klaros-Testmanagement. We will continue to use the printer example project from the quick start guide.

It is recommended to complete the quick-start guide first if you wish to follow these guides step-by-step.

3.1. User Role Management

In this section we will learn how project access works and how to limit the access to our example project to specific users.

The default installation of Klaros-Testmanagement contains the three users `admin`, `manager` and `tester` (see the Klaros-Testmanagement documentation here and here for more information about user roles). If you create a new project (see Section 2.1.1, " Creating a Project ") all users of Klaros-Testmanagement have access to this project. To limit the access to this project to a single users or several specific users, you can use project specific roles, which are part of Klaros-Testmanagement Enterprise Edition.

3.1.1. Limiting Project Access

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

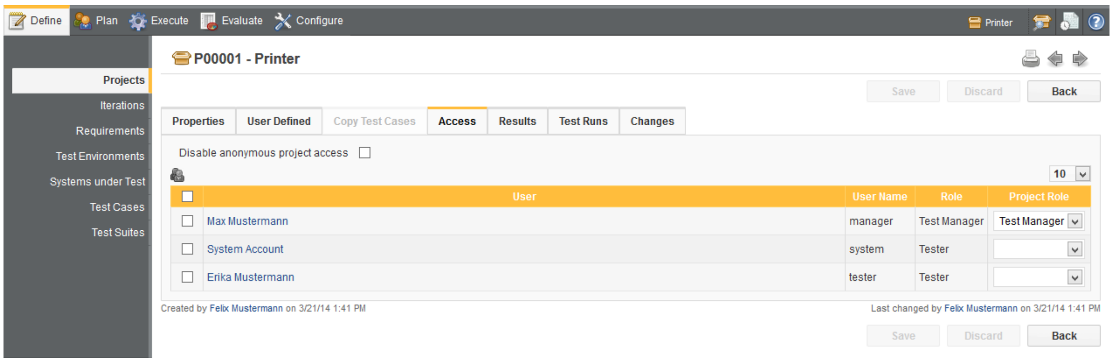
In this section we will limit access to the *Printer* project. See here for more information about project specific roles.

Assigning project specific roles

Project specific roles do not need to match the role of the user. For example, a user with the role *test manager* can be a *tester* in one project and a *test manager* in another project.

1. Login to Klaros-Testmanagement using the *Manager* account.
2. Select the **Define** section in the topbar.
3. Press the *P00001* label, or the Printer project if you already created more projects.
4. Select the *Access* tab.

5\.



P00001 - Printer

Disable anonymous project access ☐

10

☐	User	User Name	Role	Project Role
☐	Max Mustermann	manager	Test Manager	Test Manager
☐	System Account	system	Tester	
☐	Erika Mustermann	tester	Tester	

Created by Felix Mustermann on 3/21/14 1:41 PM

Last changed by Felix Mustermann on 3/21/14 1:41 PM

Figure 3.1. The Access tab

6. Click on the Assign button.

7. In the following dialog select the checkbox in front of *Max Mustermann* and select *Test Manager* from the *Project Role* dropdown list (if not already selected)

8. Confirm the changes by clicking on the Apply button.

The only user allowed to view the *Printer* project now is the user named *manager*. Administrators can still view the project, since they are excluded from project access rules. On the projects page, you can see which projects have access rules defined by the icon in the *Additional information* column.

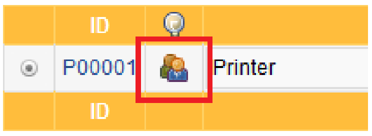


Figure 3.2. A Project with Limited Access

1. Login to Klaros-Testmanagement using the *Manager* account.

2. Select the **Define** section in the topbar.

3. Press the *P00001* label, or the Printer project in case you have created more projects.

4. Select the *Access* tab.

5. In the *Erika Mustermann* column, select *Tester* in the *Project Role* dropdown list.

   Click on the Assign button.

6. In the following dialog select the checkbox in front of *Erika Mustermann* and select *Tester* from the *Project Role* dropdown list (if not already selected)

7. Confirm the changes by clicking on the Apply button.

Important

If you use project specific roles, you need to assign at leat one test manager per project.

3.2. Categorization

Feature only available in Klaros-Testmanagement Enterprise Edition

Important

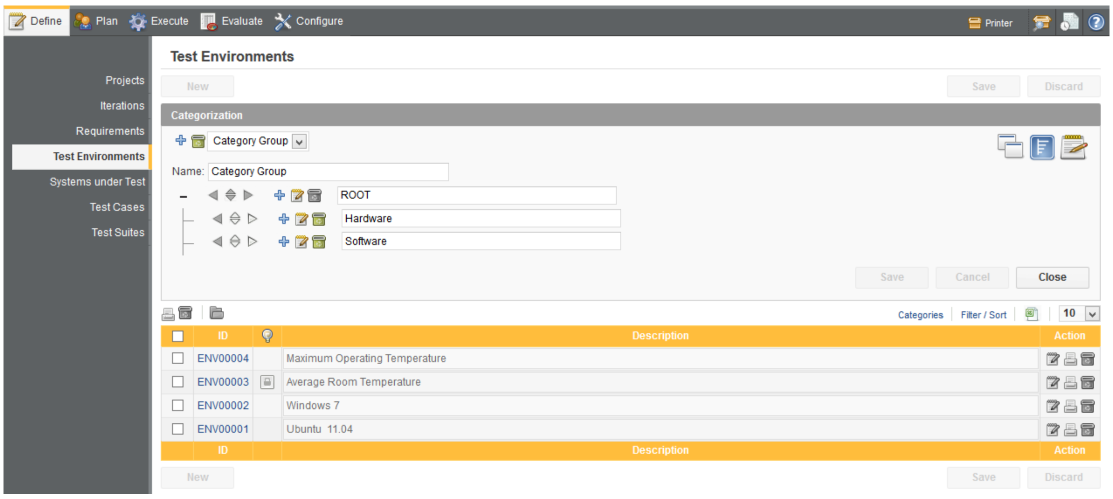
In order to use the examples in this guide it is recommended that you complete Section 2.1.2, “ Creating Test Environments ” of the quick-start guide.

In our example project, we have separate test environments and test cases for hardware- and software-related criteria. A useful way to separate these within in Klaros-Testmanagement Enterprise Edition is to use the categorization feature.

Test environments, systems under test, test cases, test suites, requirements and iterations can all be assigned to categories in different category groups. Each object can only be assigned to one category in each category group, but there is no limit on the amount of category groups that may be created.

This how-to describes the process of assigning the test environments created in Section 2.1.2, “ Creating Test Environments ” of the quick-start guide into hardware- and software-related categories.

1. Login to Klaros-Testmanagement using the *Manager* account.
2. Select the *Printer* project.
3. Select the **Test Environments** menu entry in the sidebar.
4. Press the *Categories* label to expand the category creation and selection panel.
5. Press the icon.
6. Press the icon to create a new category group.



Test Environments

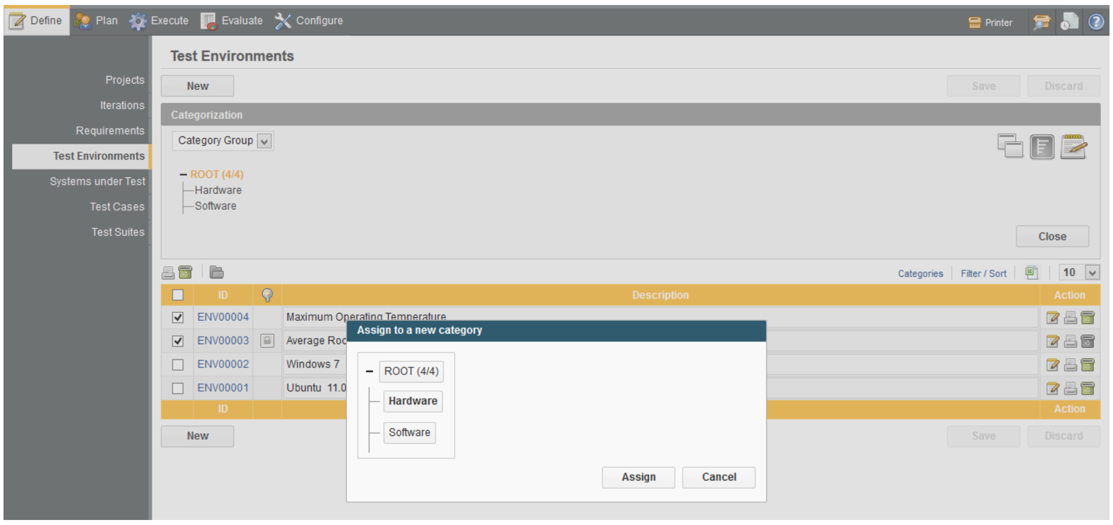
Name: Category Group

ROOT
Hardware
Software

ID		Description	Action
ENV00004		Maximum Operating Temperature	
ENV00003		Average Room Temperature	
ENV00002		Windows 7	
ENV00001		Ubuntu 11.04	

Figure 3.3. Creating Categories

7. Press the icon on the newly created *ROOT* category (this is the root category of this category group).
8. Enter `Hardware` in the text field of the new category.
9. Press the icon on the *ROOT* category again.
10. Enter `Software` in the text field of the new category.
11. Press the Save button.
12. Select the checkboxes beside each of the software-related test environments (`Windows 7` and `Ubuntu 10.4`). .
13. Press the icon.
14. Press the `Software` entry in the popup.



Test Environments

- ROOT (4/4)
  - Hardware
  - Software

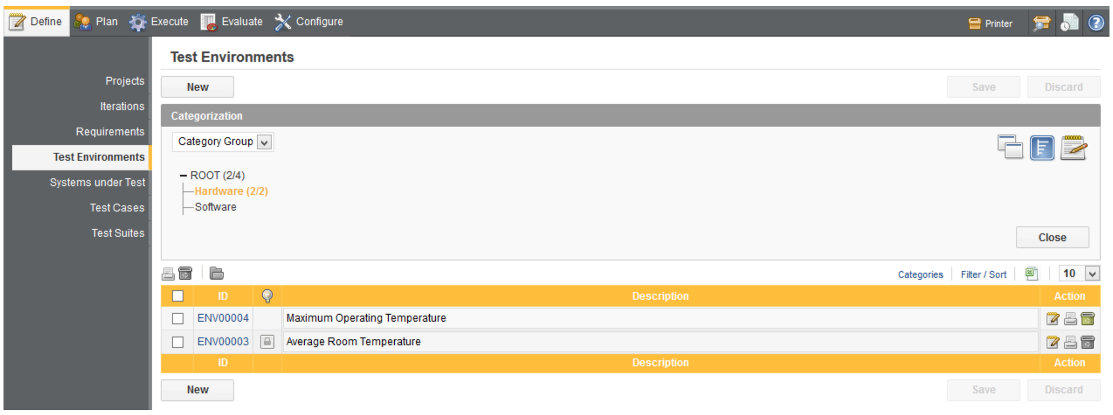
	ID		Description	Action
✓	ENV00004		Maximum Operating Temperature	
✓	ENV00003		Average Roc	
	ENV00002		Windows 7	
	ENV00001		Ubuntu 11.0	
	ID			Action

Assign to a new category

- ROOT (4/4)
  - Hardware
  - Software

Figure 3.4. Assigning Test Environments to Categories

15. Press the Assign button.
16. Select the checkbox beside each of the hardware-related test environments.
17. Press the icon.
18. Press the `Hardware` entry in the popup.
19. Press the Assign button.



ID	Description	Action
ENV00004	Maximum Operating Temperature	
ENV00003	Average Room Temperature	

Figure 3.5. The Test Environment Categories

From now on, all test environments can be viewed as normal if the category panel is closed, and opening the category panel allows for selective viewing of the particular categories of test environments.

3.3. User Defined Properties

Feature only available in Klaros-Testmanagement Enterprise Edition

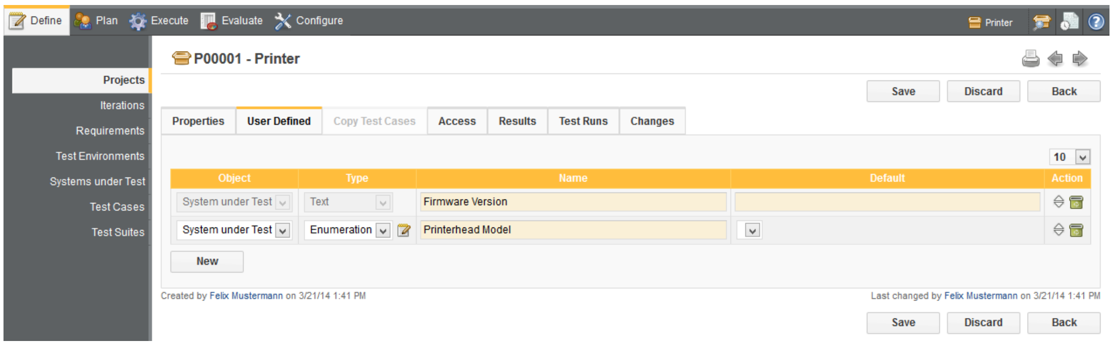
Important

In order to use the examples in this guide it is recommended that you complete Section 2.1.1, " Creating a Project " of the quick-start guide.

User defined properties are customizable fields which can be added to the various artifacts of a project e.g. to make it easier to find specific elements or to reference external systems or documents. In this how-to, we will set up some user defined properties for our systems under test (or SUTs).

Since the SUTs in our example are printers, there are a number of properties that will make searching through them easier. One of these is the firmware version, an alpha-numeric value which is changed relatively often.

1. Login to Klaros-Testmanagement using the *Manager* account.
2. Select the *Printer* project.
3. Press the icon.
4. Select the *User Defined* tab.
5. Click the New button.



Object	Type	Name	Default	Action
System under Test	Text	Firmware Version		
System under Test	Enumeration	Printerhead Model		

Created by Felix Mustermann on 3/21/14 1:41 PM

Last changed by Felix Mustermann on 3/21/14 1:41 PM

Figure 3.6. Adding User Defined Properties

6. Select *System Under Test* in the *Object* column.

7. Select *Text* in the *Type* column.

8. Enter `Firmware Version` in the *Name* text field.

9. A second property we can use to search through our SUTs is the model of printerhead used in the printer. Since our company has only two different printerhead models, it makes sense to make this an enumeration property.

   Click the New button.

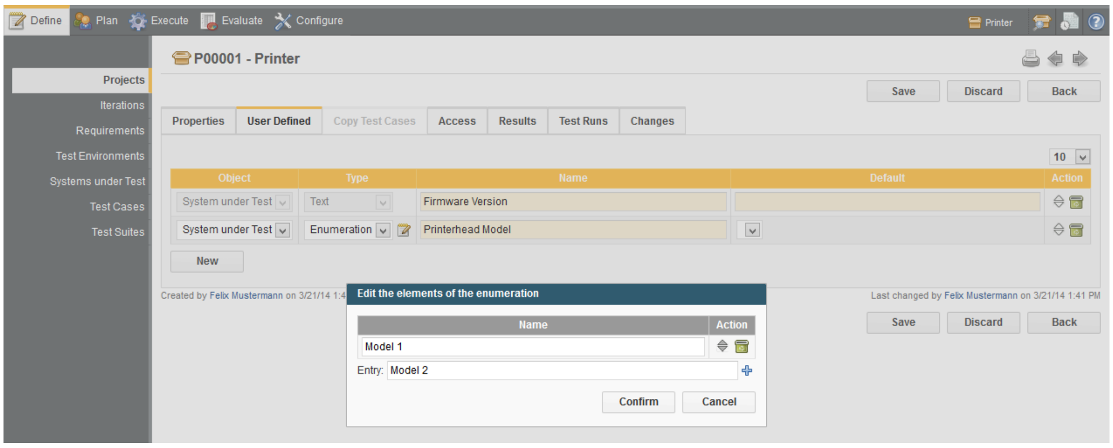
10. Select *System Under Test* in the *Object* column.

11. Select *Enumeration* in the *Type* column.

12. Enter `Printerhead Model` in the *Name* text field.

13. Press the icon in the *Type* column to add enumeration values.

14. Enter `Model 1` in the *Entry* text field and press the icon.



Object	Type	Name	Default	Action
System under Test	Text	Firmware Version		
System under Test	Enumeration	Printerhead Model		

Created by Felix Mustermann on 3/21/14 1:4

Last changed by Felix Mustermann on 3/21/14 1:41 PM

Edit the elements of the enumeration

Name	Action
Model 1	

Entry: Model 2

15. Enter `Model 2` in the *Entry* field and press the icon.

16. Press the Confirm button.

17. Press the Save button.

3.4. Pausing and Resuming Test Runs

Important

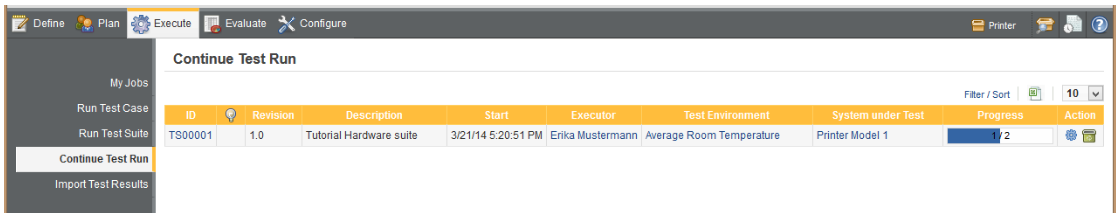
In order to use the examples in this guide it is recommended that you complete Section 2.1.2.4, “ Creating Test Suites ” of the quick-start guide.

Klaros-Testmanagement supports pausing and resuming of test suite runs. If the manual test runner is cancelled after at least one test case has been completed, the test run is automatically saved in the *Continue Test Runs* section.

1. Login to Klaros-Testmanagement using the *Tester* account.
2. Select the *Printer* project.
3. Navigate to the **Execute** section using the topbar.
4. Select the **Run Test Suite** menu entry in the sidebar.
5. Select the *Tutorial Hardware* test suite.
6. Complete the first test case as explained in Section 2.1.3.2, “ Executing Test Suites ”.
7. Press the Cancel button.

   The manual test runner will now close, returning you to the main Klaros-Testmanagement window.
8. Select the **Continue Test Run** menu entry in the sidebar.

   The interrupted test suite run will now be displayed, showing that one of the two test cases in the test suite have been completed.
9. You can click the ⚙ icon to continue executing the test suite or the 🗑 icon to delete this test suite run and the associated results.



Continue Test Run

ID		Revision	Description	Start	Executor	Test Environment	System under Test	Progress	Action
TS00001		1.0	Tutorial Hardware suite	3/21/14 5:20:51 PM	Erika Mustermann	Average Room Temperature	Printer Model 1	1 / 2	⚙ 🗑

Figure 3.7. Continuing a Test Run

3.5. Configuring Issue Management System

In order to set up Klaros-Testmanagement to use issue management systems, you must log in as an administrator.

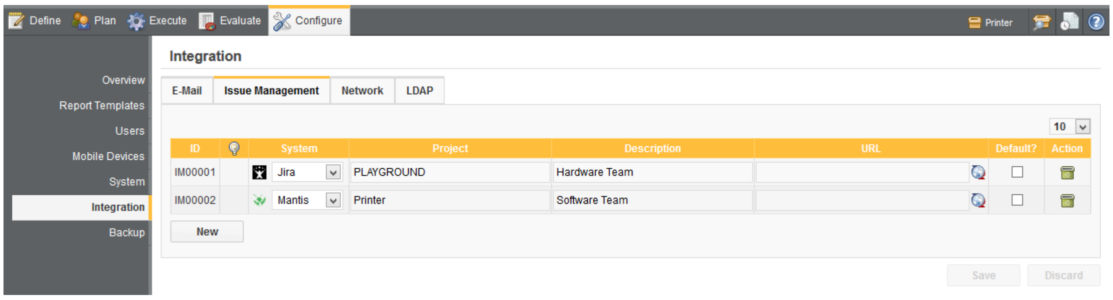
Note

Three user accounts are set up by default during the installation of Klaros-Testmanagement an administrator account, a manager account and a tester account.

Username	Password
admin	admin
manager	manager
tester	tester

Table 3.1. User Roles

1. Log in as administrator and navigate to **Configure** - **Integration** - *Issue Management*.
2. Press the New button.



ID		System	Project	Description	URL	Default?	Action
IM00001		Jira	PLAYGROUND	Hardware Team		☐	
IM00002		Mantis	Printer	Software Team		☐	

Figure 3.8. The Issue Management page

3. Select JIRA in the *System* drop-down list.
4. Enter `PRINTER` in the *Project* text field (or the name of a test project, e.g. `PLAYGROUND`)
5. Enter `Hardware Team` in the *Description* text field.
6. (Optional) If you have a JIRA system to test with, enter the address in the *URL* text field and press the Validate button.
7. Press the Save button to save your new issue management system.

Repeat the process, selecting Mantis instead of JIRA and using `Software Team` as the description.

Now these two issue management systems can be assigned to projects within Klaros-Testmanagement.

This concludes the role of the administrator account in our quick start guide. Log out of Klaros-Testmanagement now and log in using a manager account.

3.6. Issues

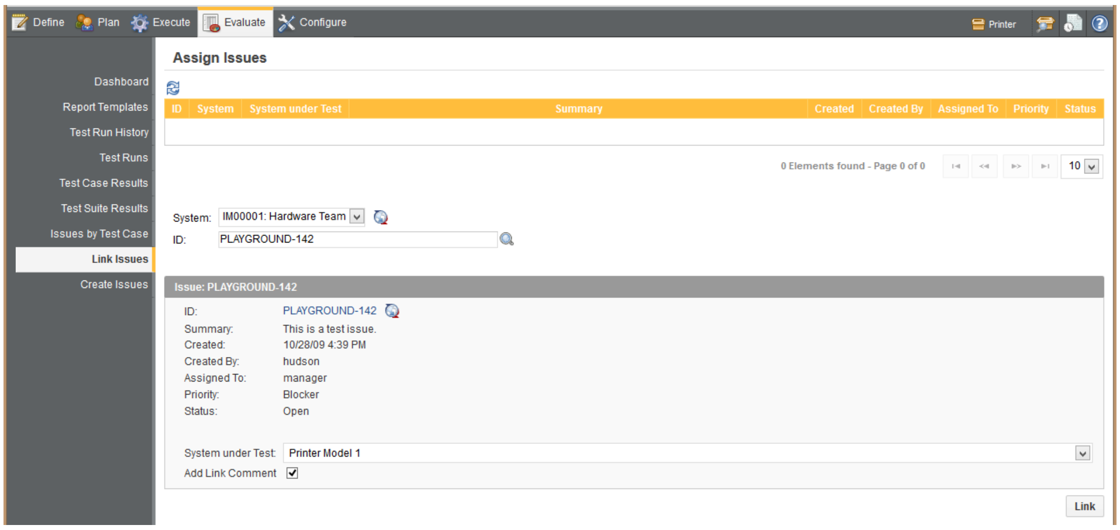
Important

In order to use the examples in this guide it is recommended that you complete Section 2.1.3.1, “ Executing Single Test Cases ” of the quick-start guide.

Klaros-Testmanagement supports the creation of issues in issue management systems (IMS) (see Section 3.5, “ Configuring Issue Management System ”) as well as linking them with test cases. This can be achieved either by creating an issue in the IMS from within Klaros-Testmanagement, or by using the *Link Issues* feature to link pre-existing issues and test cases.

Both of these actions can be carried out from within the manual test runner or in the *Evaluate - Link Issues* section of Klaros-Testmanagement. Each *Results Overview* page in the manual test runner has the Link Issue and Create Issue buttons.

3.6.1. Linking Issues



Assign Issues

ID	System	System under Test	Summary	Created	Created By	Assigned To	Priority	Status

0 Elements found - Page 0 of 0 10

System: IM00001: Hardware Team
ID: PLAYGROUND-142

Issue: PLAYGROUND-142

ID:	PLAYGROUND-142
Summary:	This is a test issue.
Created:	10/28/09 4:39 PM
Created By:	hudson
Assigned To:	manager
Priority:	Blocker
Status:	Open

System under Test: Printer Model 1
Add Link Comment ✓

Figure 3.9. Linking Issues

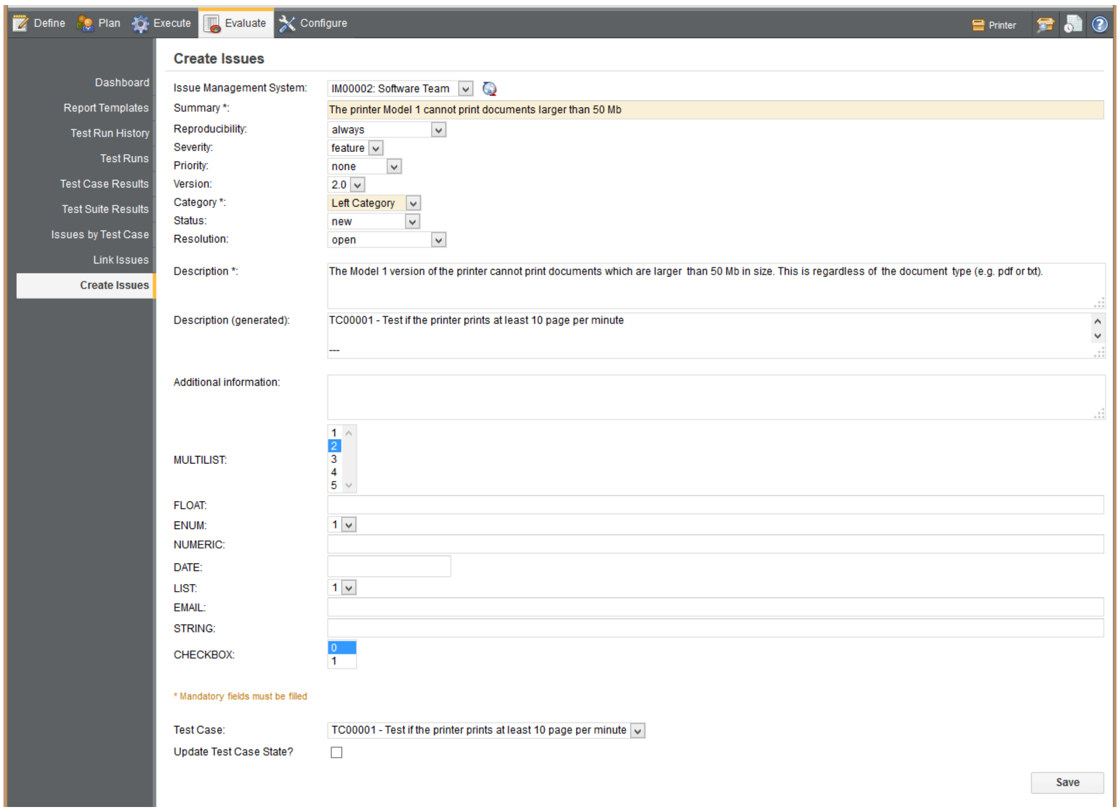
1. Login to Klaros-Testmanagement using the *Tester* account.
2. Click on the **Evaluate** link in the topbar.
3. Select the **Issues** menu entry in the sidebar.
4. Press the icon in the action column of the test case *Test if the printer prints at least 10 pages per minute*.
5. Press the **Create** button.
6. Select the *Hardware Team* issue management system from the *Issue Management System* dropdown list.
7. Enter a valid issue id in the *ID* text field (this id has to match an existing issue in the IMS).
8. Press the icon next to the *ID* field (Please wait a few moments until the issue has been retrieved from the IMS).

9. Select the system under test *Printer Model 1* in the *System under Test* dropdown list.

10. Press the `Link` button.

The list of assigned issues should now contain a single entry.

3.6.2. Creating Issues



Create Issues

Issue Management System:	IM00002: Software Team
Summary *:	The printer Model 1 cannot print documents larger than 50 Mb
Reproducibility:	always
Severity:	feature
Priority:	none
Version:	2.0
Category *:	Left Category
Status:	new
Resolution:	open
Description *:	The Model 1 version of the printer cannot print documents which are larger than 50 Mb in size. This is regardless of the document type (e.g. pdf or txt).
Description (generated):	TC00001 - Test if the printer prints at least 10 page per minute ---
Additional information:	
MULTILIST:	1 [✓ 2] 3 4 5
FLOAT:	
ENUM:	1
NUMERIC:	
DATE:	
LIST:	1
EMAIL:	
STRING:	
CHECKBOX:	[✓ 0] 1

* Mandatory fields must be filled

Test Case:	TC00001 - Test if the printer prints at least 10 page per minute
Update Test Case State?	☐

Figure 3.10. Creating Issues

1. Login to Klaros-Testmanagement using the *Tester* account.

2. Press the **Evaluate** link in the topbar.

3. Select the **Issues** menu entry in the sidebar.

4. Press the **New** button.

5. Select the `Software Team` issue management system from the *Issue Management System* dropdown list.

6. Enter `The printer Model 1 cannot print documents larger than 50 Mb` into the *Summary* field.

7. In the *Description* field, enter `The Model 1 version of the printer cannot print documents which are larger than 50 Mb in size. This is regardless of the document type (e.g. pdf or txt).`

8. Select the test case *Test if the printer prints at least 10 pages per minute* in the *Test Case* dropdown list.

9. Press the *Save* button.

You should now see a message *The issue was successfully created in Mantis with ID XYZ.* in the log panel on top of the screen.

3.7. Revisions

Important

In order to use the examples in this guide it is recommended that you complete Section 2.1.2.3, “ Creating Test Cases ” of the quick-start guide.

Test requirements can change during the lifecycle of a project, due to various internal or external changes. Let's assume that in our *Printer* project for example, 20 pages per minute instead of 10 are the new status quo for printers. Instead of copying and editing the test case *Test if the printer prints at least 10 page per minute* you can simply create a new revision for that test case.

Note

With Klaros-Testmanagement you can create Revisions for test cases, test suites and requirements.

1. Login to Klaros-Testmanagement using the *Manager* account.
2. Select the *Printer* project.
3. Press the **Define** link in the topbar.
4. Select the **Test Cases** menu entry in the sidebar.
5. Press the icon on the test case *Test if the printer prints at least 10 page per minute*.
6. Press the *Revisions* tab.
7. Press the New Revision button.
8. Enter `requirements have changed` in the *Comments* text field.
9. Press the Yes button.
10. Press the *Properties* tab.
11. Change the *Description* of the test case to `Test if the printer prints at least 20 page per minute` .

You have successfully created a new revision of the test case.

Note

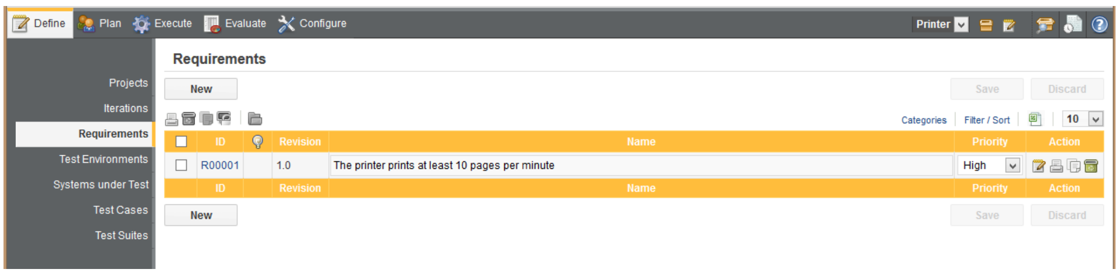
When creating a new revision, this revision gets selected as the active revision for that test case. You can, at any time, switch back to an older revision via the revision tab.

3.8. Requirements

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

Requirements are conditions which must be met in order for the product to be considered ready. A requirement can be linked to several test cases, which must be executed in order to fulfill this requirement.

In the course of this howto, we will create a requirement, link it to a test case and execute this test case.



ID	Revision	Name	Priority	Action
R00001	1.0	The printer prints at least 10 pages per minute	High	

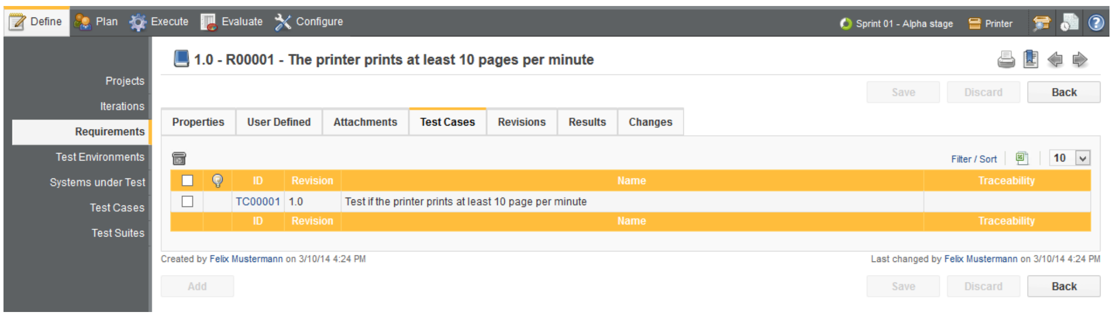
Figure 3.11. The Requirements Page

1. Select the **Define** section in the topbar.
2. Select the **Requirements** menu entry in the sidebar.
3. Press the `New` button.
4. Write `The printer prints at least 10 pages per minute` in the *Name* text field.
5. Select *High* from the *Priority* dropdown list.
6. Press the `Save` button.
7. Press the icon of the requirement *The printer prints at least 10 pages per minute*.
8. Select the *Test Cases* tab.
9. Press the `Add` button.
10. Select the test case *Test if the printer prints at least 10 page per minute*.
11. Press the `Add` button.

Multiple Requirements

You can assign multiple test cases to the same requirement.

You can also assign test cases to multiple requirements!



1.0 - R00001 - The printer prints at least 10 pages per minute

		ID	Revision	Name	Traceability
☐		TC00001	1.0	Test if the printer prints at least 10 page per minute	
		ID	Revision	Name	Traceability

Created by Felix Mustermann on 3/10/14 4:24 PM

Last changed by Felix Mustermann on 3/10/14 4:24 PM

Figure 3.12. The Requirements Details Page

Now we will execute the test case with which the requirement is assigned to.

1. Switch to the **Execute** section in the topbar.
2. Select the **Run Test Case** menu entry in the sidebar.
3. Press the Execute button.
4. Press the Execute button.
5. Press the Passed button.
6. Press the Yes button.

You have now executed a test case that is assigned to a requirement. Let's see the results of this test case on the requirements page.

1. Select the **Define** section in the topbar.
2. Select the **Requirements** menu entry in the sidebar.
3. Press the icon of the requirement *The printer prints at least 10 pages per minute*.
4. Select the *Results* tab.

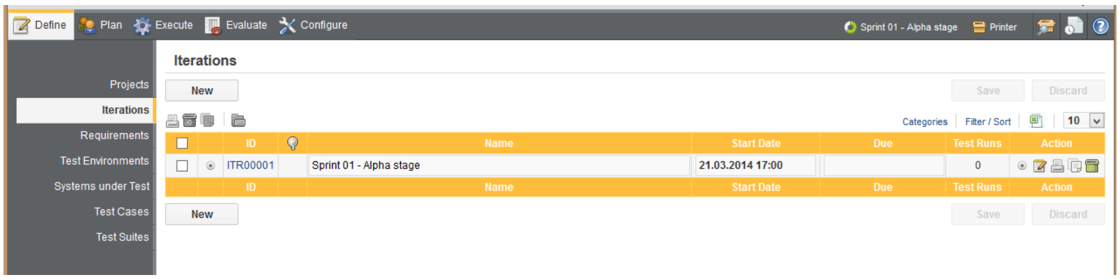
In the results tab, you can see the results of all test cases that are assigned to this requirement. Since we've executed a single test case, there is only one element in this table. Pressing the icon shows you the details of this test case result.

3.9. Iterations

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

An iteration represents a single test cycle in a project. This helps to synchronize the test process with an agile development process (e.g. Scrum). An iteration can represent a milestone for a project. In a software development project for example, an iteration could be *The core functionality is working*.

In this section, we will create an iteration, learn how to activate or deactivate iterations and how to assign requirements to a specific iteration.



ID		Name	Start Date	Due	Test Runs	Action
ITR00001		Sprint 01 - Alpha stage	21.03.2014 17:00		0	

Figure 3.13. The Iterations Page

1. Select the **Define** section in the topbar.
2. Select the **Iterations** menu entry in the sidebar.
3. Press the `New` button.
4. Enter `Sprint 01 - Alpha stage` in the *Name* text field.
5. Press the icon.
6. Enter `All printer models can print a test page without an error.` and `All printers print at least 10 pages per minute.` in the *Success Criteria* text field.
7. Select the *Properties* tab.
8. Enter `All printer models can print a test page without an error.` and `All printers print at least 10 pages per minute.` in the *Success Criteria* text field.
9. Press the `Save` button.

You have now successfully created an iteration. Since this is the only iteration in the project, it gets activated. You can see the active iteration in the topbar (see Section 3.9, “ Iterations ”).

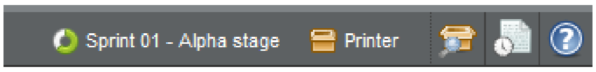
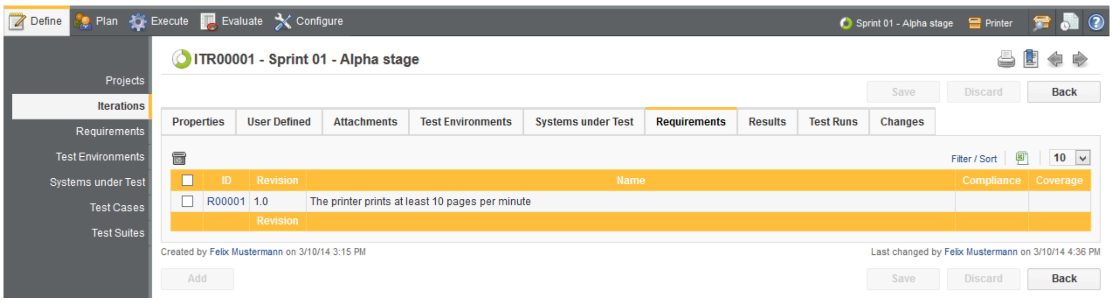


Figure 3.14. The Selected Iteration in the Topbar

Using Iterations with Requirements

Iterations are more useful when combined with requirements (see Section 3.8, “ Requirements ”). For example, the iteration "Alpha release" of a project could contain the requirements `All database unit tests pass` and `The setting menu is working`.



ID	Revision	Name	Compliance	Coverage
R00001	1.0	The printer prints at least 10 pages per minute		

Created by Felix Mustermann on 3/10/14 3:15 PM

Last changed by Felix Mustermann on 3/10/14 4:36 PM

Figure 3.15. The Iteration Details Page

1. Select the *Requirements* tab.
2. Select the requirement *The printer prints at least 10 pages per minute*.
3. Press the Add button.
4. Press the Add button.
5. Click on the Assign button.
6. Select the requirement *The printer prints at least 10 pages per minute*.
7. Click on the Assign button.

You have now successfully linked a requirement to an iteration.

If an iteration is active, only results and artefacts for this iteration are displayed. To deselct the current iteration or switch to a different one, follow the next steps.

1. Click in the icon in the topbar.
2. Select the empty entry in the iteration dropdown box.

Chapter 4. Report Tutorial

This tutorial will show how to create reports for Klaros-Testmanagement. Please note that custom reports are only available in Klaros-Testmanagement Enterprise Edition. Basic knowledge in Java programming and XML might be helpful to understand this tutorial. An example of the resulting report can be found here. The complete code can be downloaded here, feel free to modify it to your needs.

To create custom reports navigate via the top menu *Configure* and then via the side menu *Report Templates* to the custom report section of Klaros-Testmanagement Enterprise Edition. You can create a new report by pressing the New button. You can later use this report via the top menu *Evaluate* and the side menu *Report Templates*.

4.1. Development Environment

The first step in this tutorial will describe how to set up a development environment to help with creating reports. We recommend using Eclipse as the development framework.

Create a project to contain your reports

- Open Eclipse and right click into the *Project Explorer*
- In the menu then select *New -> Other*.
- In the dialog select *Java Project* as shown in the picture then click on *Next*.

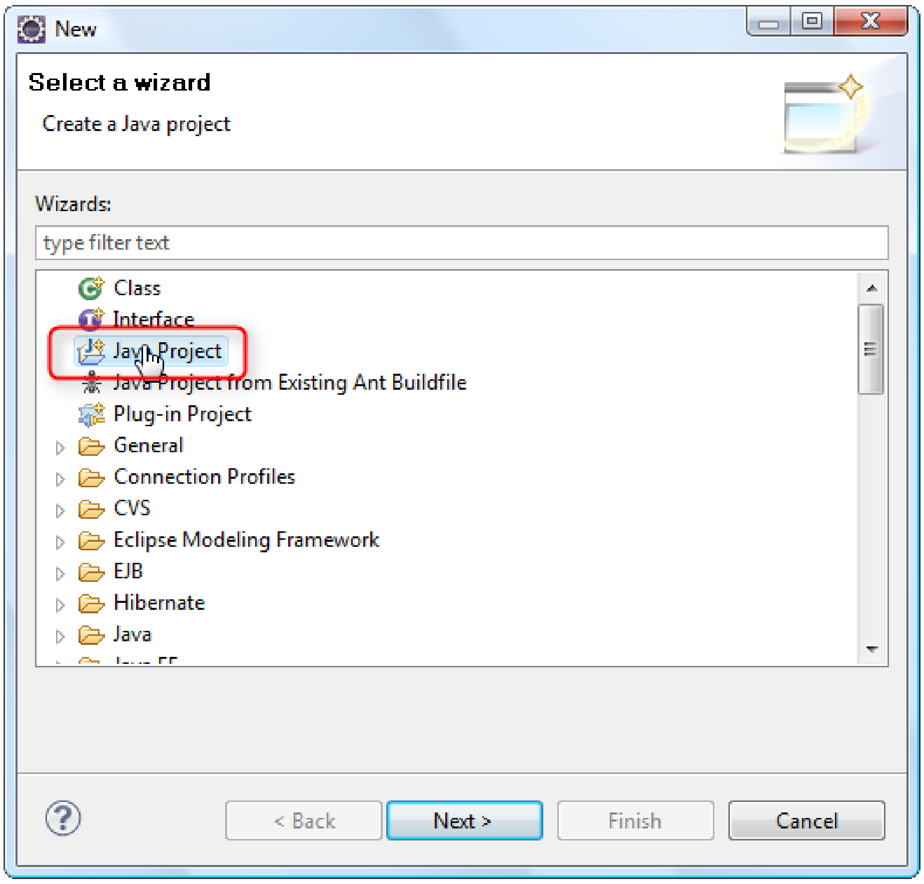
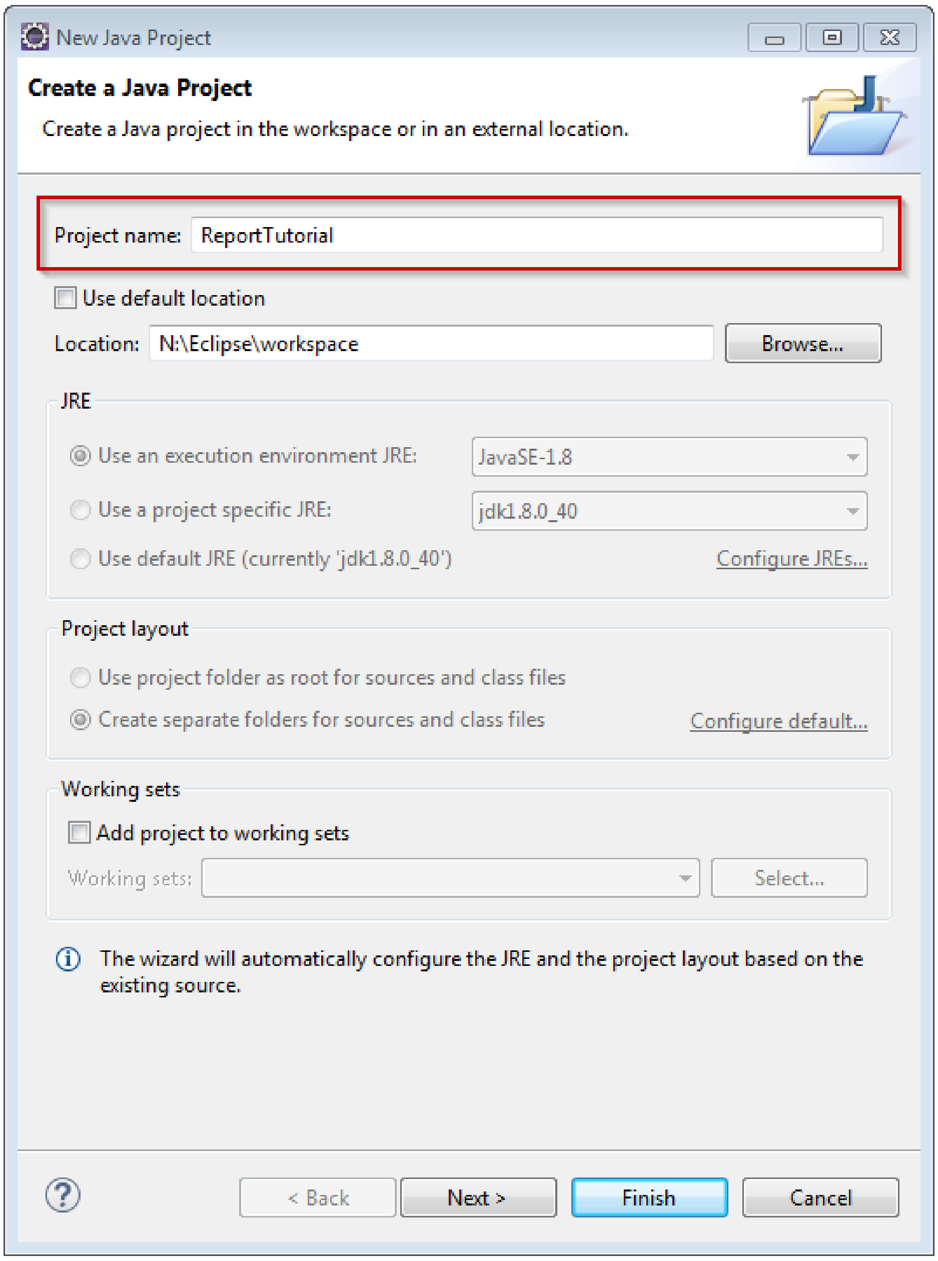


Figure 4.1. The Wizard Selection Screen

- In the next dialog enter a title for your project (e.g. `ReportTutorial`) and press *Finish*.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: ReportTutorial

☐ Use default location

Location: N:\Eclipse\workspace

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: jdk1.8.0_40

○ Use default JRE (currently 'jdk1.8.0_40')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

☐ Add project to working sets

Working sets:

The wizard will automatically configure the JRE and the project layout based on the existing source.

Figure 4.2. The New Project Wizard

- The *Package Explorer* in Eclipse should now show your new project.

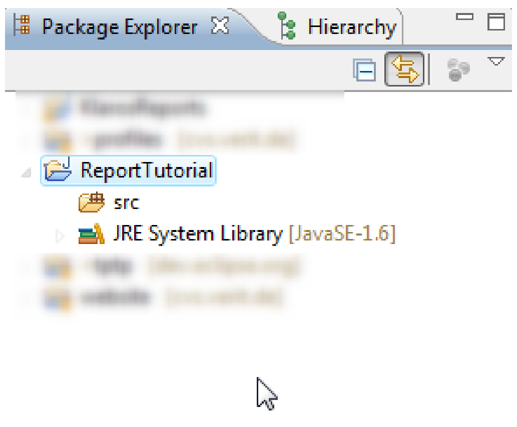


Figure 4.3. The Package Explorer

Set up the Project

- First, create a subfolder to hold your XML files. Right click on your project and select *New -> Folder*. In the dialog enter `xml` in the *Folder name* field and click on Finish .

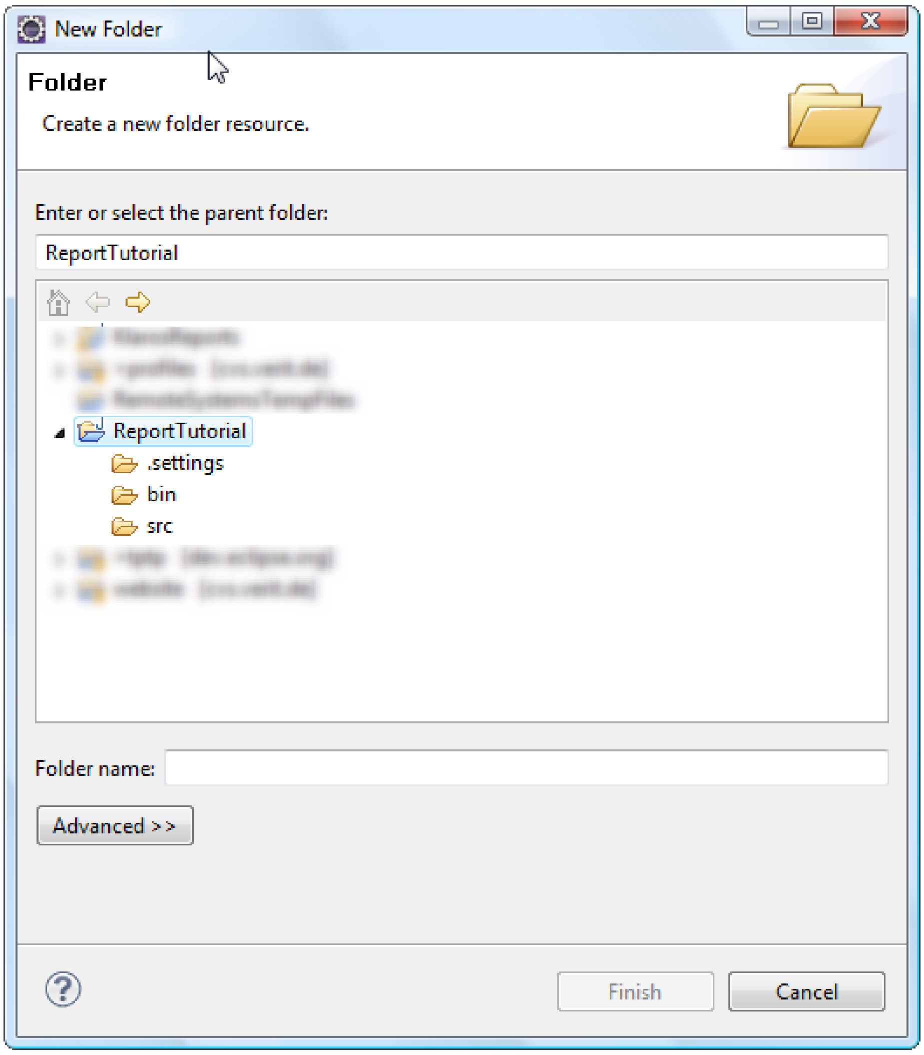


Figure 4.4. Creating a New Folder

- Add the Klaros-Testmanagement scripting library to your build path. Right click on our project and select *Properties*. In the dialog click on *Java Build Path*(1), select the *Libraries*(2) tab and then click on *Add External JARs...*(3).

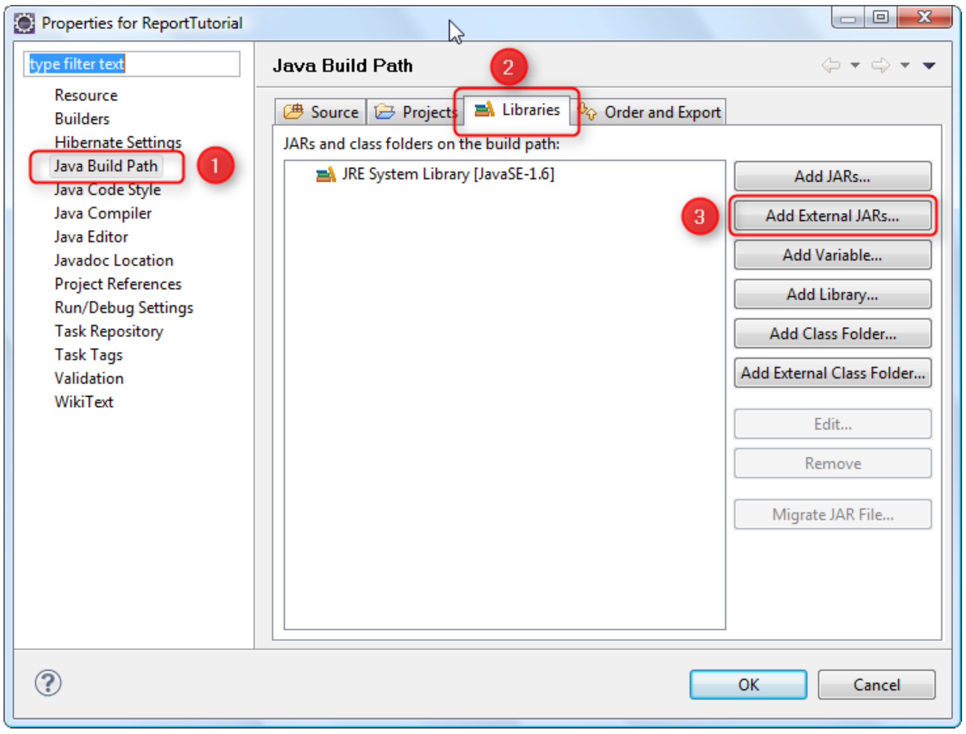


Figure 4.5. The Build Path Setting Screen

- In the following dialog navigate to your Klaros-Testmanagement installation folder. From there navigate to the *webapps/klaros-web/WEB-INF/lib folder* and select the klaros-core-x.y.z, klaros-model-x.y.z.jar and klaros-scripting-x.y.z.jar, then click the `Open` button.

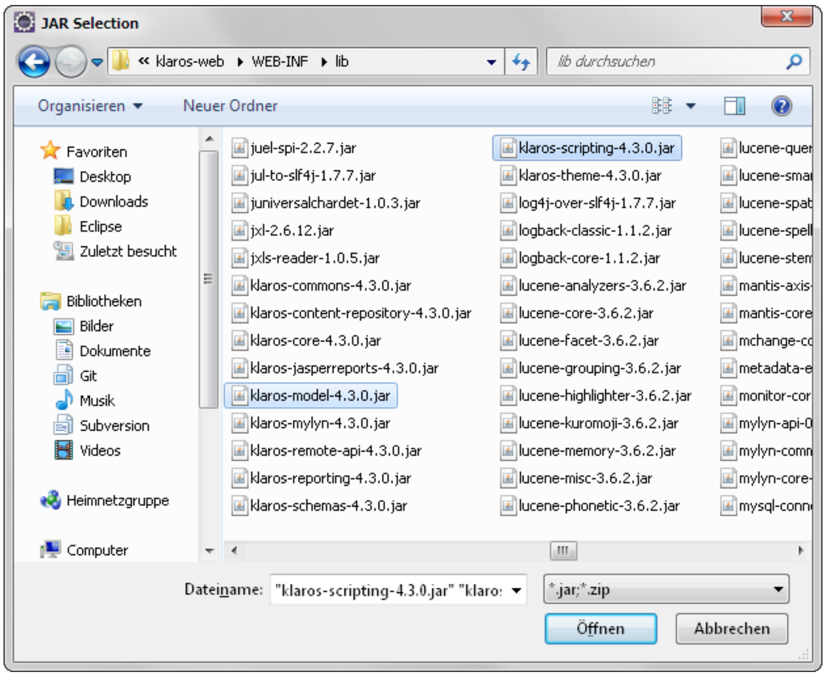


Figure 4.6. Selecting an External Jar File

- Finally click on the `OK` button in the *Build Path* dialog. You now have an additional entry in your project named *Referenced Libraries*.

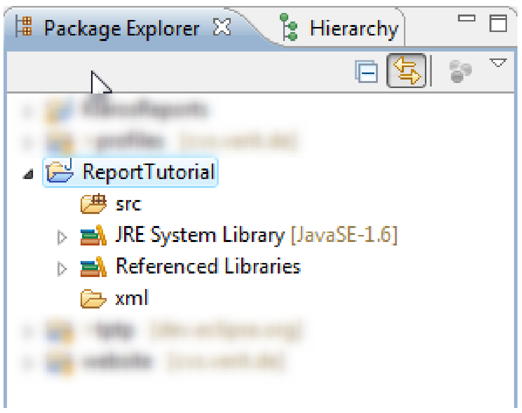
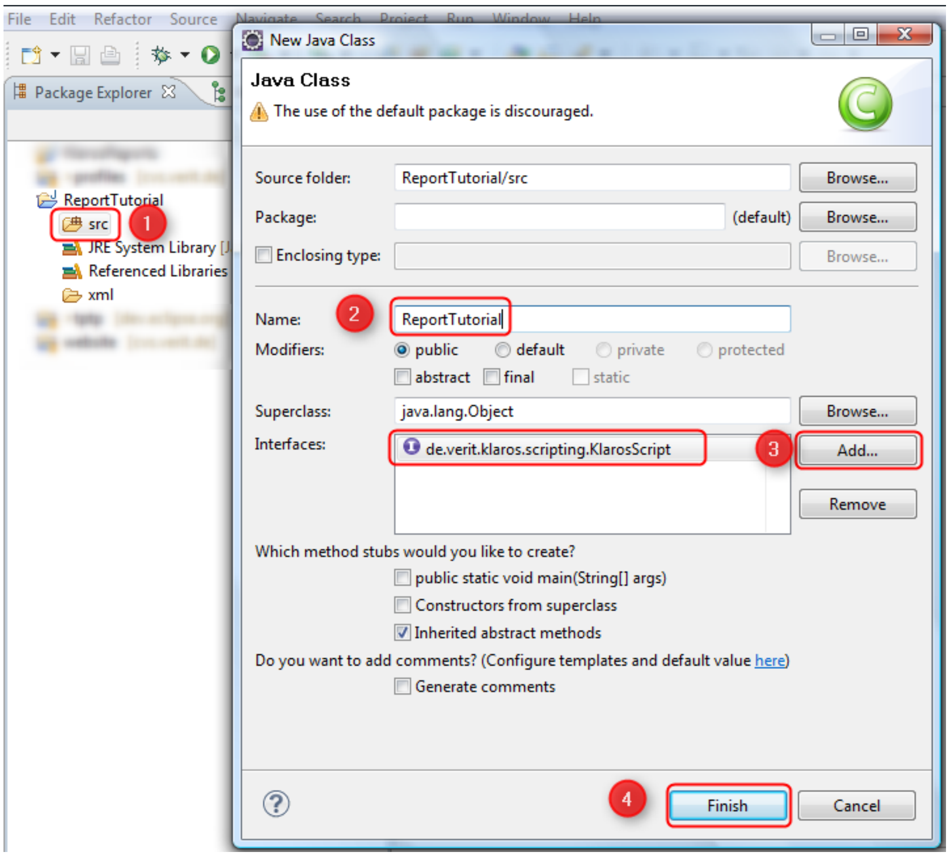


Figure 4.7. The Package Explorer

4.2. Preparing the Data for the Report

The next step of this tutorial will show how to gather and prepare data for the report. The report we are going to create will show a summary of all test runs for all test cases in a test suite together with a detailed description of all executed test steps, listed per test case per test run. To provide the data for the report the API described in the Klaros API documentation can be used to retrieve the required data.

The following picture shows the steps to perform to create a KlarosScript class.



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: ReportTutorial/src

Package: (default)

Enclosing type:

Name: ReportTutorial

Modifiers: [✓ public] default private protected

abstract final static

Superclass: java.lang.Object

Interfaces: de.verit.klaros.scripting.KlarosScript

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

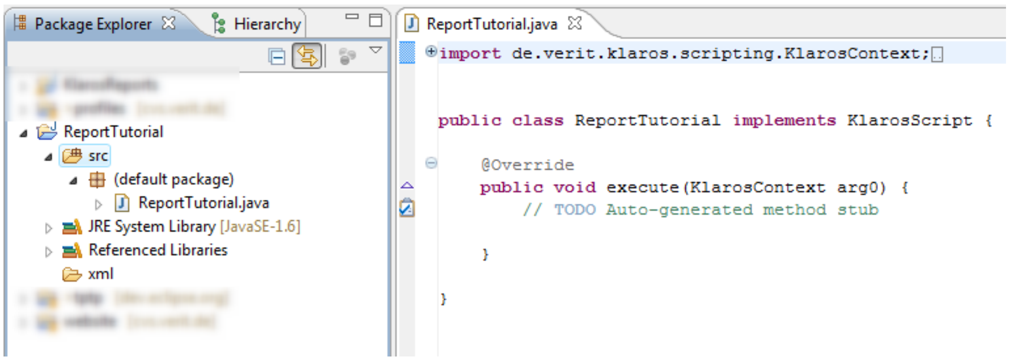
[✓ Inherited abstract methods]

Do you want to add comments? (Configure templates and default value here)

Generate comments

Figure 4.8. Creating a Class

- First a class implementing the KlarosScript interface must be created. Right click on the src folder (1) in your project and select *New -> Class*.
- In the new dialog enter a name for your script class (2)
- Then click on the `Add` button (3) and enter **`KlarosScript`** in the new dialog to provide the KlarosScript interface for the class. Then press the `ok` button to close the interface dialog.
- Finally press the `Finish` button (4). After a short while the class should be created and Eclipse should present you the following view.



```
import de.verit.klaros.scripting.KlarosContext;

public class ReportTutorial implements KlarosScript {

    @Override
    public void execute(KlarosContext arg0) {
        // TODO Auto-generated method stub

    }
}
```

Figure 4.9. The Created Class

- Before entering any code, the import statements must be provided.

```
import de.verit.klaros.scripting.*;
import de.verit.klaros.core.model.*;
import java.util.*;
```

- The code to be executed must be entered into the *execute* method of the KlarosScript class. To provide a certain level of flexibility, the test suite to generate the report for should be passed as a parameter to the KlarosScript class. We will see later in this tutorial how this works in detail. It is achieved by the following line of code:

```
String suiteName = (String)context.getParameterValue("suiteName");
```

- Next the database is accessed to retrieve the data for the report. The following snippet shows how to access the database. This code selects all entries from the KlarosTestSuite table whose ids matches the test suite id passed to the KlarosScript and that are present in the currently active project. The result is returned as a list, though only one single entry is expected.

```
String suiteQuery = "select g from KlarosTestSuite g where g.name='" + suiteName + "'";
if (context.getActiveProject() != null) {
  suiteQuery += " and g.configuration.name='";
  suiteQuery += context.getActiveProject().getName();
  suiteQuery += "'";
}
List testSuites = context.executeQuery(suiteQuery);
```

- To avoid any errors when executing the KlarosScript the following code is contained in an if-clause if no matching test suite was found.

```
if (!testSuites.isEmpty()) {
 testSuite = (IKlarosTestSuite) testSuites.get(0);
 ...
}
```

- Add the test suite to the context so it can later be retrieved from the template.

```
testSuite = (IKlarosTestSuite) testSuites.get(0);
context.add("testSuite", testSuite);
```

- Now the preparations for the pie chart in the report can begin. The report should present a pie chart showing the amount of test case results in the states error, skipped, failure and success. Therefore a list of KlarosTestCaseResult is required for each result type. This code is also placed inside the if-clause.

```
List<KlarosTestCaseResult> error = new ArrayList<>();
List<KlarosTestCaseResult> failure = new ArrayList<>();
List<KlarosTestCaseResult> success = new ArrayList<>();
List<KlarosTestCaseResult> skipped = new ArrayList<>();
```

- Now the lists get filled. First we step through the list of test cases belonging to the test suite.

```
 for (KlarosTestCase klarosTestCase : testSuite.getTestCases())
{
...
}
```

- Inside this loop we iterate over the test results for each test case. This can be more than one since a test case can have several test runs. The following code must be placed in the for loop from the step before.

```
for (KlarosTestCaseResult testResult: klarosTestCase.getResults())
{
...
}
```

- We now have the test results at hands and can distribute them to the four lists defined earlier, depending on the state of the test result. This code must be placed inside the second for loop defined one step before.

```
if (testResult.isError()) error.add(testResult);
else if (testResult.isFailure()) failure.add(testResult);
else if (testResult.isPassed()) success.add(testResult);
else if (testResult.isSkipped()) skipped.add(testResult);
```

- We are almost done. The last step is to store the four lists containing the KlarosTestCaseResults in the context to access it later from the template. This code is placed inside the outermost if-clause and outside of the two for loops.

```
context.add("error", error);
context.add("failure", failure);
context.add("success", success);
context.add("skipped", skipped);
```

- The complete KlarosScript class can be found in the archive here.

4.3. Create the Layout of the Report

Klaros-Testmanagement uses the *JBoss Seam PDF* framework to render the report and fill the retrieved data into the layout template.

1. We will start with an empty document that contains a header and footer definition.

```
<p:document xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:p="http://jboss.org/schema/seam/pdf"
  title="Klaros-Testmanagement Test Plan Report" marginMirroring="true"
  author="#{user.name}" creator="#{user.name}" pageSize="A4">
  <f:facet name="header">
    <p:font size="8">
      <p:header borderWidthBottom="0.1" borderColorBottom="black" borderWidthTop="0" alignment="center">
        <p:text value="Test Run Report - Generated on #{date} by #{user.name}" />
      </p:header>
      <p:footer borderWidthTop="0.1" borderColorTop="black" borderWidthBottom="0" alignment="center">
        <p:text value="Page " />
        <p:pageNumber />
        <p:text value=" - Created with Klaros-Testmanagement (www.klaros-testmanagement.com)" />
      </p:footer>
    </p:font>
  </f:facet>
  <p:paragraph>
    <p:text value=" " />
  </p:paragraph>
</p:document>
```

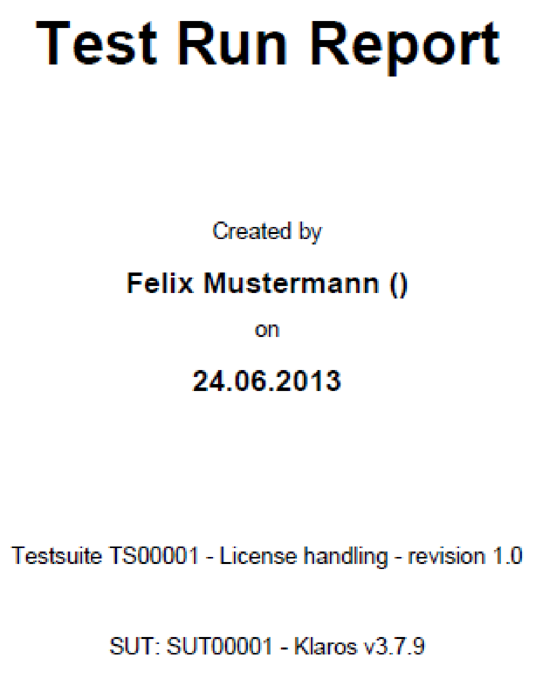
This snippet creates a header that is displayed on every page. It contains the date and the name of the Klaros-Testmanagement user who created the report. The footer contains the page number and an informative text that the report was created with Klaros-Testmanagement. Note that for the header the `borderWidthBottom` attribute was set to provide a separation from the following text, while for the footer the `borderWidthTop` attribute was used to create a single line for separation.

2. Next we create a front page accumulating the main information about the report.

   The layout of the front page is done using `<p:paragraph>` elements for formatting. These elements use alignment and spacing to define the position where the text will be placed. To change the font of the paragraphs, you can use the `<p:font>` element and use its style and size attributes to highlight certain parts.

```
  <p:paragraph alignment="center" spacingAfter="100">
        <p:text value="" />
    </p:paragraph>
    <p:font style="bold" size="32">
        <p:paragraph alignment="center" spacingAfter="75">
            <p:text value="Test Run Report" />
        </p:paragraph>
    </p:font>
    <p:font style="normal" size="12">
        <p:paragraph alignment="center" spacingAfter="5">
            <p:text value="Created by" />
        </p:paragraph>
    </p:font>
    <p:font style="bold" size="16">
        <p:paragraph alignment="center" spacingAfter="5">
            <p:text value="#{user.name} (#{user.email})" />
        </p:paragraph>
    </p:font>
    <p:font style="normal" size="12">
        <p:paragraph alignment="center" spacingAfter="5">
            <p:text value="on" />
        </p:paragraph>
    </p:font>
    <p:font style="bold" size="16">
        <p:paragraph alignment="center" spacingAfter="75">
            <p:text value="#{date}" />
        </p:paragraph>
    </p:font>
    <p:font style="normal" size="12">
        <p:paragraph alignment="center" spacingAfter="30">
            <p:text value="Testsuite " />
            <p:text value="#{testSuite.name}" />
            <p:text value=" - " />
            <p:text value="#{testSuite.shortname}" />
            <p:text value=" - revision " />
            <p:text value="#{testSuite.revisionId}" />
        </p:paragraph>
        <p:paragraph alignment="center" spacingAfter="5">
            <p:text value="SUT: " />
            <p:text value="#{testSuite.sut.name}" />
            <p:text value=" - " />
            <p:text value="#{testSuite.sut.productversion}" />
        </p:paragraph>
    </p:font>
    <p:newPage />
```

An image of a sample front page can be found below.


Test Run Report

Created by

Felix Mustermann ()

on

24.06.2013

Testsuite TS00001 - License handling - revision 1.0

SUT: SUT00001 - Klaros v3.7.9

Figure 4.10. A Sample Front Page of a Report

As you might have noticed from the header and footer definition, values stored in the context can be accessed by preceding the context variable enclosed in curly brackets with a #, e.g. *#{user.name}*. The context variable for the user is automatically inserted into the context by Klaros-Testmanagement. You might remember the part where we prepared the data for this report and added the test suite to the context.

```
context.add("testSuite", testSuite);
```

For the front page we access this context variable to retrieve the data that came with this test suite, e.g. *#{testSuite.name}*, *#{testSuite.shortname}*, and *#{testSuite.revisionId}*. You can find the accessible attributes in the Klaros-Testmanagement API Documentation reference in the online documentation. All attributes that have a getter can be accessed, e.g. for `#{testSuite.name}` see the interface IKlarosTestSuite where you will find the method getName in its IKlarosLabeledObject parent interface. To add a page break to the document, you can use

```
<p:newPage />
```

3. Now that we have a front page we can start to fill in the data we prepared and stored in the context. To give a quick overview over the test results we will start with a pie chart displaying the amount of successful, failed, skipped, and errorneous test case results in this test suite.

```
<p:paragraph horizontalAlignment="center" spacingAfter="25">
      <p:piechart title="Testresults per Testrun" direction="anticlockwise"
          circular="true" startAngle="30" labelGap="0.1" labelLinkPaint="black"
          plotBackgroundPaint="white" labelBackgroundPaint="white" is3D="true"
          borderVisible="false">
          <p:series key="results">
              <p:data key="Error [#{error.size}]" value="#{error.size}"
                  sectionPaint="#FF0A0A" />
              <p:data key="Success [#{success.size}]" value="#{success.size}"
                  sectionPaint="#33CC00" />
              <p:data key="Failure [#{failure.size}]" value="#{failure.size}"
                  sectionPaint="#FFCC00" explodedPercent=".2" />
              <p:data key="Skipped [#{skipped.size}]" value="#{skipped.size}"
                  sectionPaint="#FFFFFF" />
          </p:series>
      </p:piechart>
  </p:paragraph>
```

An example of a generated pie chart can be found below.

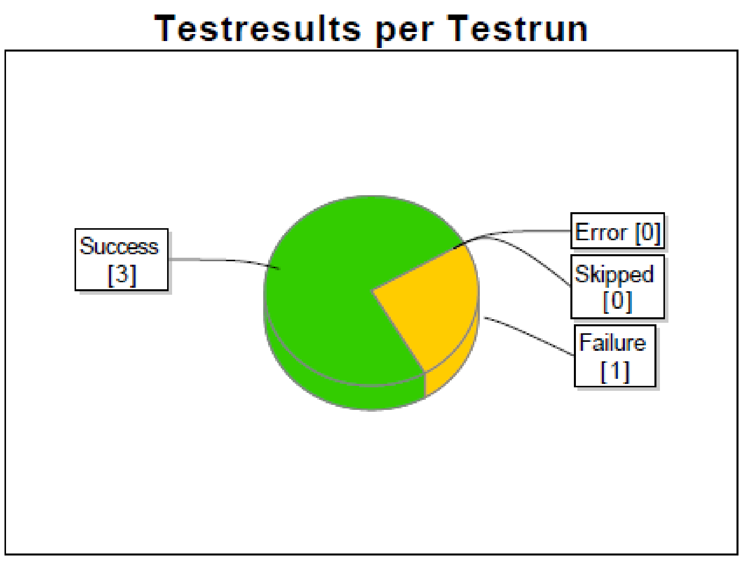


Figure 4.11. A Generated Pie Chart

For a detailed description about how to layout the pie chart and many other charts, please check the seam-pdf documentation. Again you might remember that we added four `List<KlarosTestCaseResult>` objects to the context:

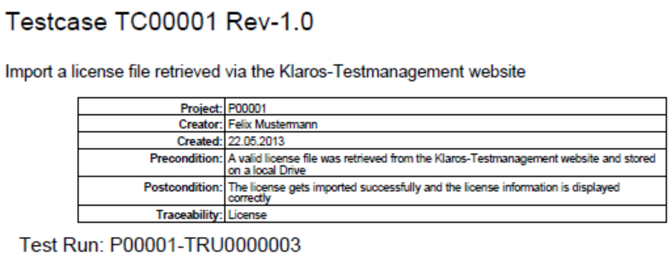
```
context.add("error", error);
context.add("failure", failure);
context.add("success", success);
context.add("skipped", skipped);
```

These four objects represent the results of our test suite. We can simply use the size of the collections representing the data for the pie chart:

```
<p:data key="Error [#{error.size}]" value="#{error.size}" sectionPaint="#FF0A0A" />
```

4. Next we want to print a page for each Test Case contained in the test suite and present its data in a table.

   As this might get a bit lengthy, only fragments will be presented here. An example can be found below.



Testcase TC00001 Rev-1.0

Import a license file retrieved via the Klaros-Testmanagement website

Project:	P00001
Creator:	Felix Mustermann
Created:	22.05.2013
Precondition:	A valid license file was retrieved from the Klaros-Testmanagement website and stored on a local Drive
Postcondition:	The license gets imported successfully and the license information is displayed correctly
Traceability:	License

Test Run: P00001-TRU0000003

Figure 4.12. The Header of a Test Case Report

First we need to loop over all test cases of the test suite:

```
<ui:repeat value="#{testSuite.testCases}" var="testCase">
  ...
</ui:repeat>
```

A loop can be done using the *<ui:repeat>* element. The attribute `value` gets a collection and the attribute `var` gets the name of a variable that can be used for further processing, e.g. `testCase`. With every step through the loop, the variable `testCase` will get the next element from the collection.

Now for the table. We need a two column table and the width of the second column should be three times the size of the first. The code is as follows:

```
<p:table columns="2" widths="1 3">
  ...
</p:table>
```

Let's get some data into the table. Therefore we have to describe what each table cell will look like. This code goes inside the `<p:table>` element.

```
<p:cell horizontalAlignment="right">
      <p:font style="bold" size="8">
          <p:paragraph>
              <p:text value="Project: " />
          </p:paragraph>
```

```
        </p:font>
    </p:cell>
    <p:cell>
        <p:paragraph alignment="left">
            <p:text value="#{testCase.configuration.name}" />
        </p:paragraph>
    </p:cell>
    <p:cell horizontalAlignment="right">
        <p:font style="bold" size="8">
            <p:paragraph>
                <p:text value="Creator: " />
            </p:paragraph>
        </p:font>
    </p:cell>
    <p:cell>
        <p:paragraph alignment="left">
            <p:text value="#{testCase.creator.name}" />
        </p:paragraph>
    </p:cell>
    ...
```

Since we defined that the table should be two columns wide, two `<p:cell>` elements build one row in the table. In the code above the first row holds the name of the project while the second line holds the name of the user who created the test case. A cell can also contain text formatting information as you can see from the `<p:font>` elements.

5. Next we need the test case results for the test case being displayed. This is where another element comes in. If there is no test result for a test case contained in the test suite an informative text should be displayed instead of just leaving a blank space.

```
  <ui:fragment rendered="#{testCase.results.isEmpty()}">
        <p:font style="normal" size="14">
            <p:paragraph alignment="left" spacingAfter="15"
                indentationLeft="10">
                <p:text value="No test runs found for this Test Case." />
            </p:paragraph>
        </p:font>
    </ui:fragment>
```

Notice the `<ui:fragment rendered="...">` element. This part of the document is only integrated into the pdf if the condition defined in the rendered attribute evaluates to `true`. This could be compared to an if-clause in modern programming languages. In the expression

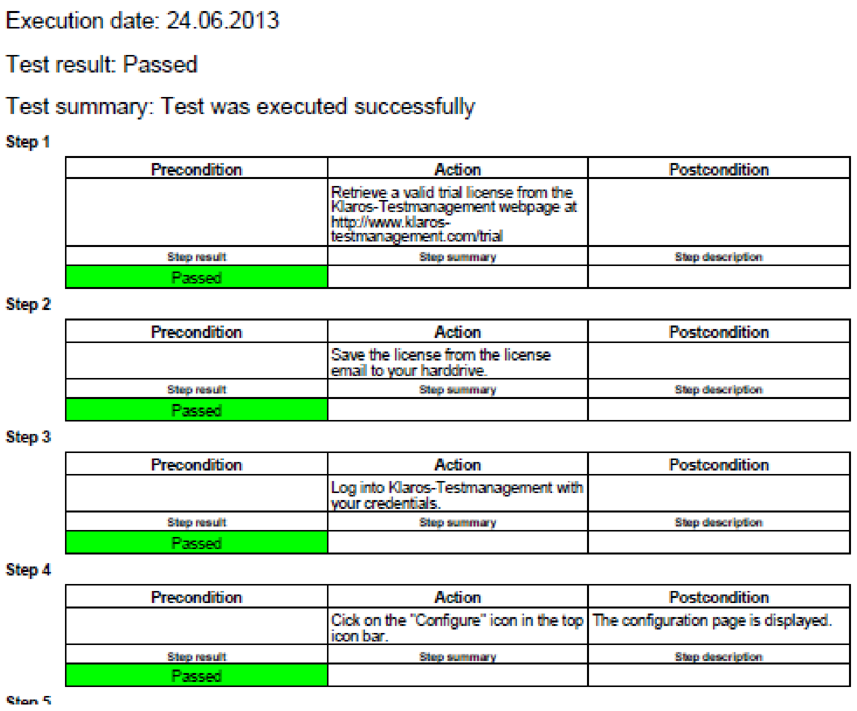
```
#{testCase.results.isEmpty()}
```

the `isEmpty()` method of a the test case result list is called which evaluates to a boolean value. Next we have to define the block that displays the data in case there are test results present for the test case:

```
  <ui:fragment rendered="#{!testCase.results.isEmpty()}">
    ...
  </ui:fragment>
```

Note the `!` which is used to negate the boolean expression we used in the block before. The code inside this expression is called if there is at least one test case result.

6. And now we loop over the results of the test cases, which will include each test run for a test case. An example of a Test Run can be found below.



Execution date: 24.06.2013

Test result: Passed

Test summary: Test was executed successfully

Step 1

Precondition	Action	Postcondition
	Retrieve a valid trial license from the Klaros-Testmanagement webpage at http://www.klaros-testmanagement.com/trial	
Step result	Step summary	Step description
Passed		

Step 2

Precondition	Action	Postcondition
	Save the license from the license email to your harddrive.	
Step result	Step summary	Step description
Passed		

Step 3

Precondition	Action	Postcondition
	Log into Klaros-Testmanagement with your credentials.	
Step result	Step summary	Step description
Passed		

Step 4

Precondition	Action	Postcondition
	Click on the "Configure" icon in the top icon bar.	The configuration page is displayed.
Step result	Step summary	Step description
Passed		

Step 5

```
<ui:repeat value="#{testCase.results.toArray()}" var="testResult">
  <!-- Start Test Result summary (equivalent to Test Run) -->
  ...
</ui:repeat>
```

This will provide us with the test results of a test case in the variable `testResult`. The next pieces of code include the displaying of a summary of the test result which is quite similar to the code used displaying the test case summary. You can have a look at it in the attached sourcefile. The next thing to display are the test case steps and their results for the current test run.

7. Therefore another loop is added inside the Test Result loop.

```
<!-- Start Test Step Result summary -->
  <ui:fragment rendered="#{!testResult.stepResults.isEmpty()}">
      <ui:repeat value="#{testResult.stepResults}" var="testStepResult">
      ...
      </ui:repeat>
  </ui:fragment>
  <ui:fragment rendered="#{testResult.stepResults.isEmpty()}">
      <p:font style="normal" size="10">
          <p:paragraph alignment="left" spacingAfter="35"
              indentationLeft="25">
              <p:text value="No test step results found." />
          </p:paragraph>
      </p:font>
  </ui:fragment>
```

Inside the `<ui:repeat> </ui:repeat>` block a table is created. This table should be displayed if there are any results for the test case step. The table displays the precondition, action, and postcondition of the test case step and also the test case step result, the test case step summary, and the test case step description. The test case step result cell of the table will get coloured according to the result, e.g. green if the step passed. Each table will have the number of the step as a headline.

8. Displaying the step number.

```
<p:font style="bold" size="8">
    <p:paragraph indentationLeft="20">
        <p:text value="Step #{testResult.stepResults.indexOf(testStepResult)+1}" />
    </p:paragraph>
</p:font>
```

   Here you can see how to retrieve the index of the test case step result from the list of test results. `testStepResult` is the variable from the inner loop, while `testResult` is the variable from the outer loop. As counting starts at zero we have to increment the retrieved value, otherwise the first step would be step 0.

9. Displaying the table if the test case has at least one test case step defined.

```
<p:table columns="3" widths="3 3 3" spacingBefore="5"
      rendered="#{testCase.testCaseSteps!=null and testCase.testCaseSteps.size() > 0}">
  ...
</p:table>
```

10. Retrieving the test case step properties (precondition, action, postcondition)

```
<p:cell horizontalAlignment="left">
  <p:font style="normal" size="8">
      <p:paragraph>
          <p:text
          value="#{testCase.testCaseSteps.get(testStepResult.precondition}" />
      </p:paragraph>
  </p:font>
</p:cell>
```

   To get to the test case step properties we have to access the test case to retrieve the data. From the list of test case steps we retrieve the test case step via the index in the list of the test case step results and then the `precondition` can be accessed. The same applies to the `action` and the `postcondition` of the test case step.

11. Colouring a cell depending on the test step result.

```
  <ui:fragment rendered="#{testStepResult.isPassed()}">
      <p:cell backgroundColor="rgb(0,255,0)"
          horizontalAlignment="center">
          <p:font style="normal" size="8">
              <p:paragraph>
                  <p:text value="Passed" />
              </p:paragraph>
          </p:font>
      </p:cell>
  </ui:fragment>
  <ui:fragment rendered="#{testStepResult.isError()}">
      <p:cell backgroundColor="rgb(255,0,0)"
          horizontalAlignment="center">
          <p:font style="normal" size="8">
              <p:paragraph>
                  <p:text value="Error" />
              </p:paragraph>
          </p:font>
      </p:cell>
```

```
    </ui:fragment>
    <ui:fragment rendered="#{testStepResult.isFailure()}">
        <p:cell backgroundColor="rgb(255,215,0)"
            horizontalAlignment="center">
            <p:font style="normal" size="8">
                <p:paragraph>
                    <p:text value="Failed" />
                </p:paragraph>
            </p:font>
        </p:cell>
    </ui:fragment>
    <ui:fragment rendered="#{testStepResult.isSkipped()}">
        <p:cell horizontalAlignment="center">
            <p:font style="normal" size="8">
                <p:paragraph>
                    <p:text value="Skipped" />
                </p:paragraph>
            </p:font>
        </p:cell>
  </ui:fragment>
```

The cells are rendered depending on the status of the test case step, therefore the methods `isError()`, `isFailure()` and so on are called for the test result object under processing. The cell gets its colour by setting the `backgroundColor` attribute to the desired rgb value.

Glossary

A

Admin	See Administrator.
Administrator	User role that has access to all functionalities in Klaros-Testmanagement.
Artifact	An Artifact is a definable object like a Project, Iteration, Requirement, Test Environment, System under Test, Job, Test Case, Test Suite or Test Case.

B

Bugzilla	Bugzilla is an open source bug tracking and testing tool.

C

Category	Artifacts can be assigned to any number of user-defined categories. Using Categories Users can group related Artifacts together.

C

Coverage	A database is a collection of information organized into interrelated tables of data and specifications of data objects.
Compliance	

D

Database	A database is a collection of information organized into interrelated tables of data and specifications of data objects.
Defect	See Issue.
Defect Management System	See Issue Management System.

E

E-Mail	Electronic mail, often abbreviated as e-mail, is any method of creating, transmitting, or storing primarily text-based communications with digital communications systems.
Error	An error is the inability of the system to perform a test correctly. Not to be confused with Failure.

F

Failure	A failure is a discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. Not to be confused with Error.

G

Guest	User role that can display artifacts and reports but may not change any data.
GUIdancer	GUIdancer is an Eclipse-based tool for automated functional testing through the Graphical User Interface (GUI).

I

Incident	See Issue.
Incident Management System	See Issue Management System.
Issue	The term issue is a unit of work to accomplish an improvement in a system. An Issue could be a bug, a requested feature, task, missing documentation, and likewise.
Issue Management System	An Issue Management System (Issue Tracking System) is a software to manage Issues.
Issue Tracking System	See Issue Management System.
Iteration	An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows from iteration to iteration to become the final product (ISTQB glossary).

J

Java	Java is a programming language. Most often, Java is used as a abbreviation for Java Runtime Environment, which needs to be installed in order to run Klaros-Testmanagement.
Java Runtime Environment	The Java Runtime environment needs to be installed in order to execute applications programmed in the Java programming language.
JavaScript	JavaScript is a scripting language most often used to add functionality to web pages. Most newer Web browsers can process JavaScript generated code.
Java Runtime Environment	See Java Runtime Environment.
JIRA	JIRA is a bug tracking, Issue tracking, and project management system by Atlassian Software .

Job	Jobs may consist of the execution of Test Cases, Test Suites or any other possible task. Jobs can be nested and assigned to individual users. The executions and results of jobs can be tracked by Klaros-Testmanagement.
Jubula	Jubula provides automated functional GUI testing for various types of applications.
JUnit	JUnit is a unit testing framework for the Java programming language.

M

Manager	User role that has access to create, edit, delete and search for objects, run test cases and test suites, show results and generate reports.
Mantis	Mantis (MantisBT, Mantis Bug Tracker) is an open source bug tracking system.

O

Operating System	An operating system (commonly abbreviated to either OS or O/S) is an interface between hardware and applications. It is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. Common contemporary operating systems include Microsoft Windows, Mac OS, Linux, BSD and Solaris.
OS	See OS.

P

Postcondition	Environmental and state conditions that must be fulfilled after the execution of a test or test procedure.
Precondition	Environmental and state conditions that must be fulfilled before the component or system can be executed with a particular test or test procedure.
Project	A project is the main unit that contains all other Artifacts that are needed to execute Test Cases.

Q

QFTest	QF-Test is a professional tool for automated testing of Java and Web applications with a graphical user interface from Quality First Software.

R

Redmine	Redmine is an open source bug tracking, issue tracking, and project management system.

Requirement	A condition or capability needed by a user to solve a problem or achieve an objective that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document (After IEEE 610).
Role	A role defines the rights the rights a user has regarding the application or selected projects. In Klaros-Testmanagement a user can take the role of an `Administrator`, `Manager`, `Tester` or `Guest`.

S

Selenium	*Selenium* is a web browser automation tool primarily used for automated testing of web apps. Selenium is able to produce *JUnit*-compatible test results, which can be imported into Klaros-Testmanagement.
SUT	See System under Test.
System Account	A System Account is a user that is not able to login at the login page and interactively control the application. System Accounts may be used for automated tasks like importing test results. If this flag is set the user is not able to login at the login page and interactively control the application. System accounts may be used for automated tasks like importing test results.
System under Test	A System under Test is used to represent a version of a software product that can be tested.

T

Test Case	A Test Case is a set of input values, execution preconditions, expected results and execution post-conditions, developed for a particular objective or test condition, such as determine whether an application or software system meets its specifications.
Test Case Result	The final verdict on the execution of a test and its outcomes, like pass, fail, or error. The result of error is used for situations where it is not clear whether the problem is in the test object.
Test Case Step	A test case step consists of execution preconditions, expected results and execution post-conditions for a single action during test execution.
Test Environment	Test Environments represent the extrinsic settings that may influence the test result. Examples for components of a Test Environments are the Operating System or an application server (e.g. Tomcat 7 on Ubuntu 12.10).
Tester	User role that has access to display artifacts and reports and run jobs, test cases and test suites.

Test Execution	The process of running a test by the component or system under test, producing actual result(s).
Test Run	The result of running a test case or test suite under the same system under test and test environment giving or more test case result(s).
Test Suite	A test suite is a set of test cases and the test cases can be executed in groups. The test suite will be used to verify and ensure that a product or system meets its design specifications and other requirements.
Trac	Trac is an open source bug tracking and issue tracking system.

U

URL	A Uniform Resource Locator (URL) specifies where an identified resource is available and the mechanism for retrieving it. Examples for URLs are http://www.klaros-testmanagement.com/" or file:///C:/klaros.txt

W

Web browser	A Web browser is a software application which enables a user to display and interact with text, images and other information typically located on a Web page at a Web site on the World Wide Web or a local area network.
Windows	Windows is the name of several Operating Systems by Microsoft.