Klaros-Testmanagement Tutorial

Version 5.7.3

Publication date May 08 2025

Klaros-Testmanagement Tutorial

by Sabrina Gidley, Fabian Klaffke, Claudia Könnecke, Klaus Mandola, Patrick Reilly, and Torsten Stolpmann

Version 5.7.3

Publication date May 08 2025

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

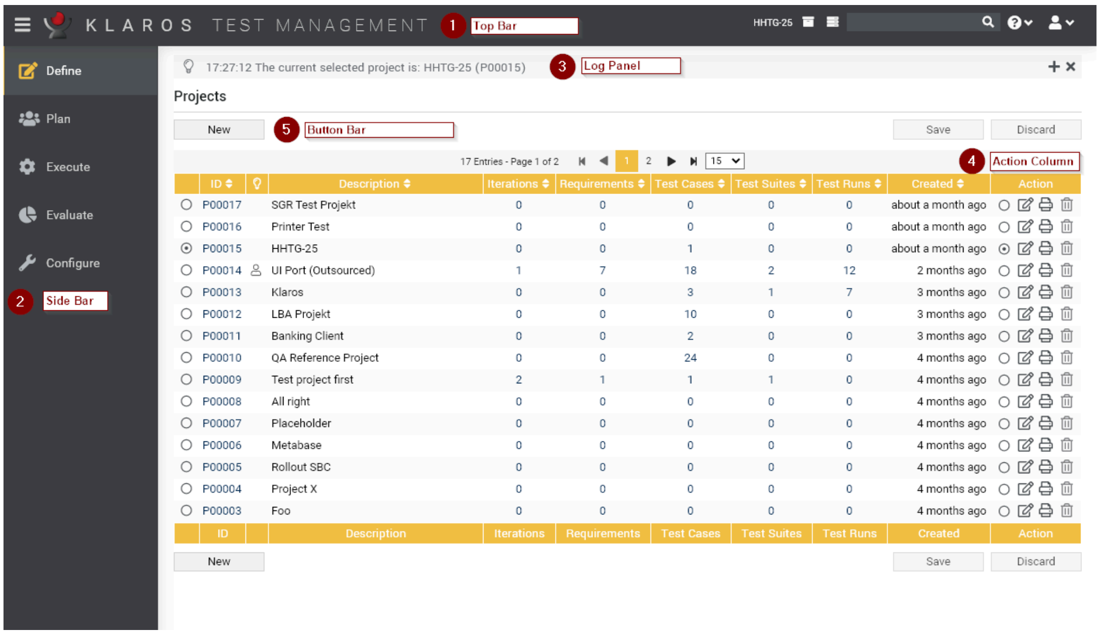
List of Tables

Chapter 1. Introduction

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 . Figure 1.1 shows a typical page with the navigation areas numbered.



17:27:12 The current selected project is: HHTG-25 (P00015)

	ID		Description	Iterations	Requirements	Test Cases	Test Suites	Test Runs	Created	Action
○	P00017		SGR Test Projekt	0	0	0	0	0	about a month ago	
○	P00016		Printer Test	0	0	0	0	0	about a month ago	
●	P00015		HHTG-25	0	0	1	0	0	about a month ago	
○	P00014		UI Port (Outsourced)	1	7	18	2	12	2 months ago	
○	P00013		Klaros	0	0	3	1	7	3 months ago	
○	P00012		LBA Projekt	0	0	10	0	0	3 months ago	
○	P00011		Banking Client	0	0	2	0	0	3 months ago	
○	P00010		QA Reference Project	0	0	24	0	0	4 months ago	
○	P00009		Test project first	2	1	1	1	0	4 months ago	
○	P00008		All right	0	0	0	0	0	4 months ago	
○	P00007		Placeholder	0	0	0	0	0	4 months ago	
○	P00006		Metabase	0	0	0	0	0	4 months ago	
○	P00005		Rollout SBC	0	0	0	0	0	4 months ago	
○	P00004		Project X	0	0	0	0	0	4 months ago	
○	P00003		Foo	0	0	0	0	0	4 months ago	

Figure 1.1. Navigation in

The User Interface of

1. The top menu bar. Here you can access functions for selecting the current project, for keyword search, help, language selection and logging out.

2. The sidebar shows the five main sections. If you hover over a specific section in the sidebar, the menu entries for this section are displayed.

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

4. Most tables in have an action column like the one shown here. A click on the respective icons in this column executes actions for the objects in the corresponding row.

5. Many pages in have buttons like this to perform actions like creating new objects, or saving or discarding changes. These buttons are always located above or below the table.

Chapter 2. Quick Start Guide

Our quick start guide shows how to create a project containing test environments, test systems and test cases and a test suite. The test cases contain test steps, and user-defined fields and categories are created. Then a single test case and a test suite with three test cases will be executed and finally the results will be displayed.

The system under test is a printer for which both hardware and software (drivers) should be tested. The driver software is available in different versions and for different operating systems, here Windows and Linux.

In addition, there are two separate development teams: one for the hardware, the other for the software. The hardware team uses as system for defect management, the software team uses .

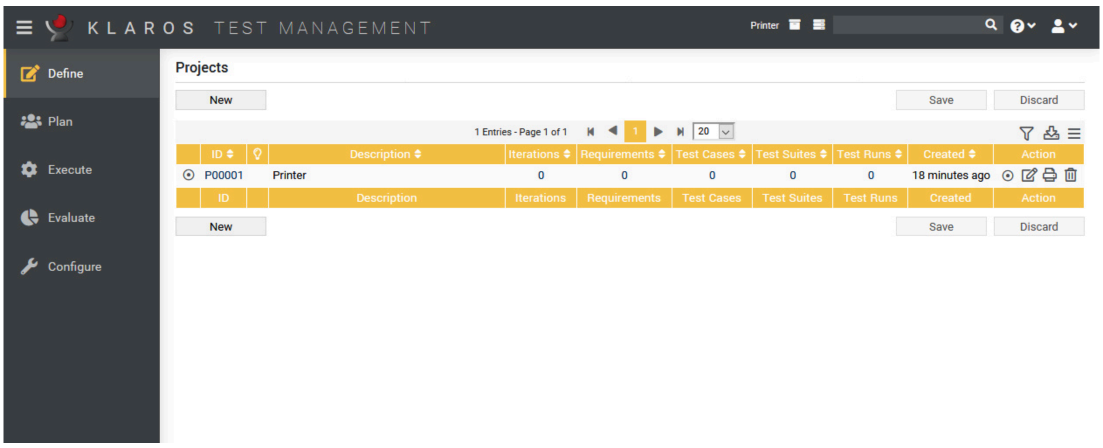
2.1. Setup

First the test setup is created. This includes the project, the test environments, the systems under test, test cases and a test suite.

2.1.1. Creating a Project

Projects are the top-level objects in . All other objects such as test cases, iterations, requirements, or test suites are part of the project. Therefore, creating the project is always the first step when working with :

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



	ID		Description	Iterations	Requirements	Test Cases	Test Suites	Test Runs	Created	Action
●	P00001		Printer	0	0	0	0	0	18 minutes ago	

Figure 2.1. The >Projects page

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

3. Press the Save button. A unique project ID is assigned automatically.

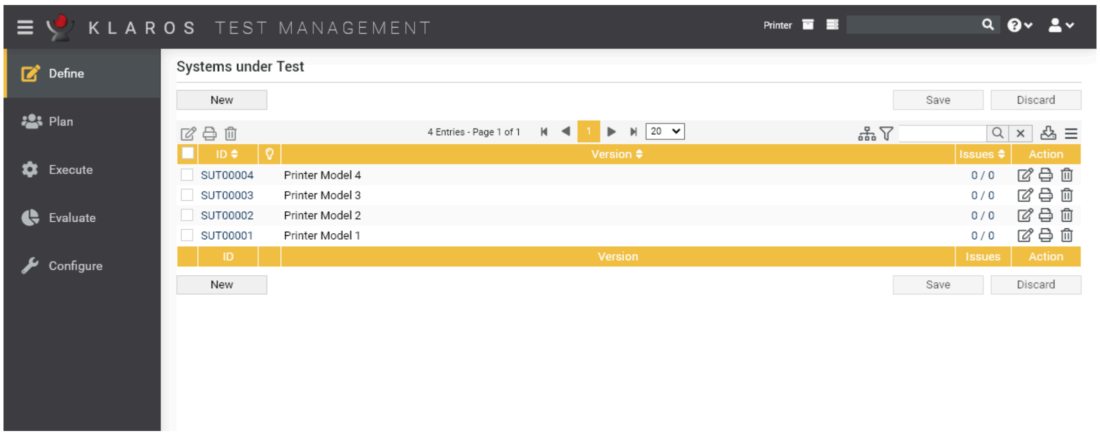
   Activate the new project with the radio button to the left of the project ID or to the right in the action column. Now the other menu items in the side menu bar can also be selected. The new project can now be edited with the icon.

2.2. Creating Systems under Test

Next we create our systems under test. In , a system under test represent the versions of the product or software system to be tested. In our example these are printers.

1. Under *Define*, select *Systems under Test* from the side menu bar.

2. Press the `New` button.



Systems under Test

4 Entries - Page 1 of 1

	ID		Version	Issues	Action
	SUT00004		Printer Model 4	0 / 0	
	SUT00003		Printer Model 3	0 / 0	
	SUT00002		Printer Model 2	0 / 0	
	SUT00001		Printer Model 1	0 / 0	

Figure 2.2. The Systems 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:

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

   Clicking in the *ID* column header changes the order of the displayed test systems.

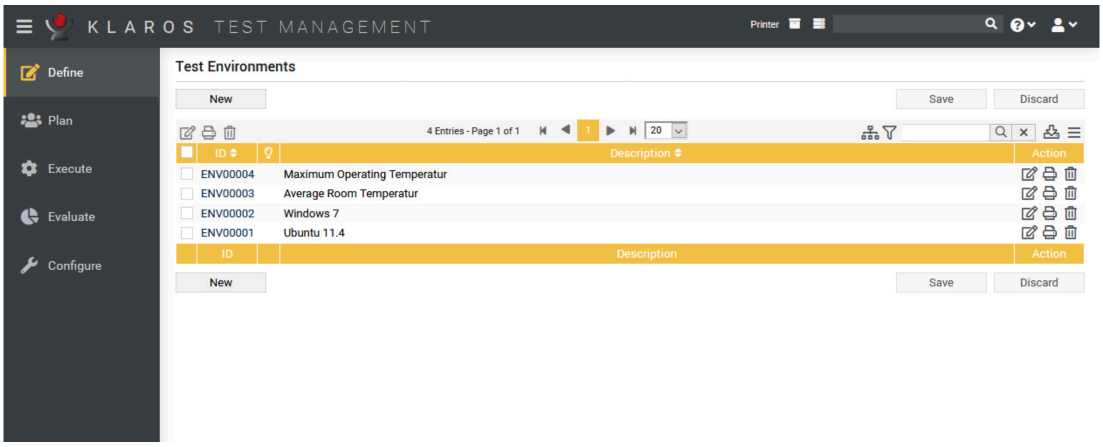
2.3. Creating Test Environments

The next step in setting up the project is to define the test environments in which the tests will take place. In the test environment defines the external influences that can influence the test result. This could be the operating system on which the printer driver is installed or the physical environment in which the printer is operated.

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

1. Under *Define*, select *Test Environments* from the side menu bar.

2. Press the `New` button.



Test Environments

ID	Description	Action
ENV00004	Maximum Operating Temperatur	
ENV00003	Average Room Temperatur	
ENV00002	Windows 7	
ENV00001	Ubuntu 11.4	

Figure 2.3. The Test Environments page

3. Enter `Ubuntu 20.04` in the *Description* text field and click the Save button.

4. Create three more test environments with the following values in the field *Description*:

   - `Windows 10`
   - `Average Room Temperature`
   - `Maximum Operating Temperature`

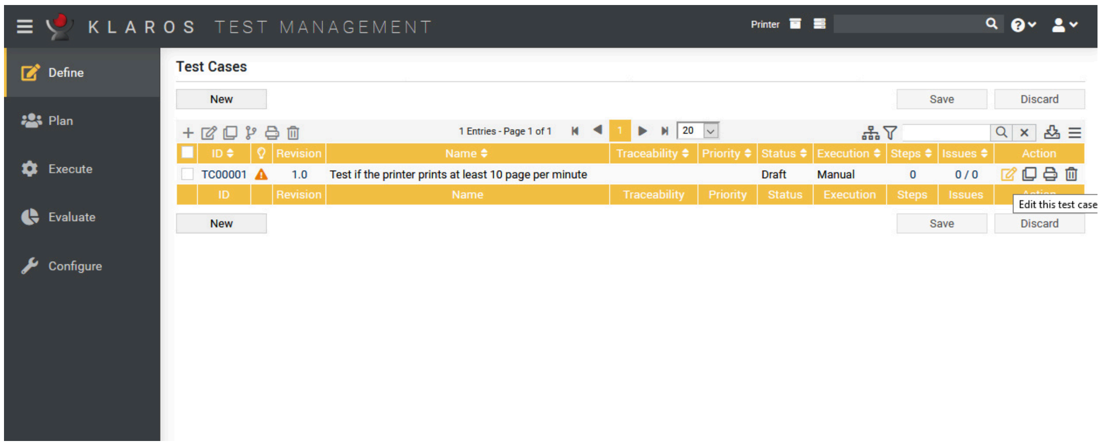
5. Press the Save button.

2.4. Creating Test Cases

Now follows the creation of the test cases. We will create test cases for both the hardware and the software team. The first test for the hardware team is to verify that the printer meets the specifications regarding the minimum number of pages printed per minute.

1. Under *Define*, select *Test Cases* from the side menu bar.

2. Press the New button.



Test Cases

ID		Revision	Name	Traceability	Priority	Status	Execution	Steps	Issues	Action
TC00001	⚠	1.0	Test if the printer prints at least 10 page per minute			Draft	Manual	0	0 / 0	

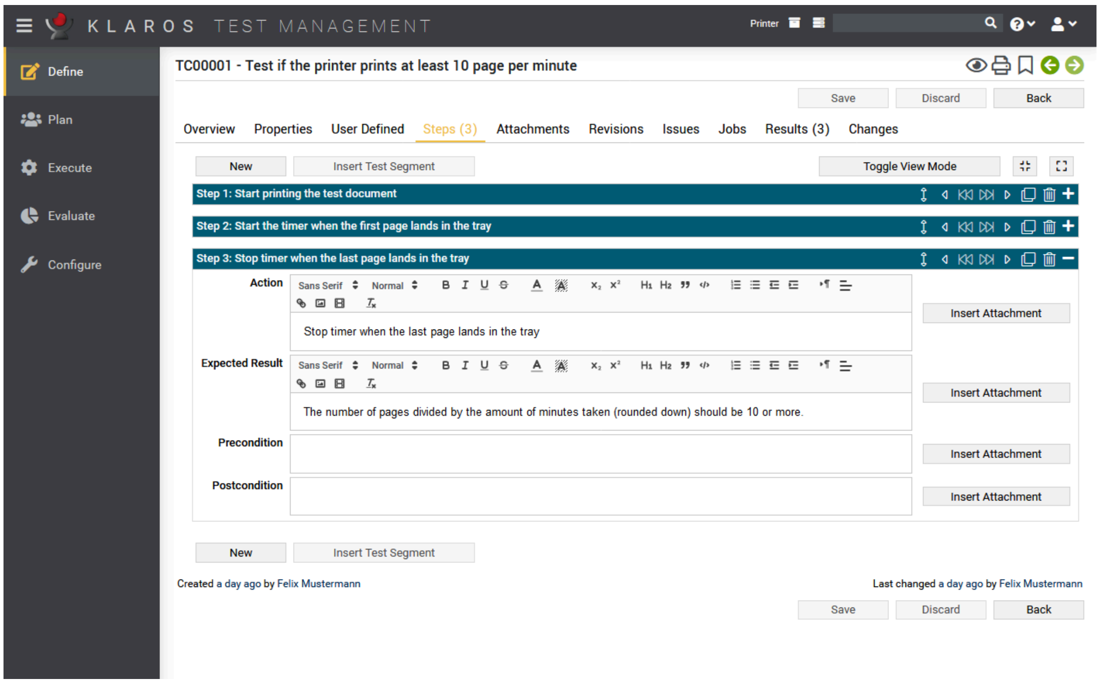
Figure 2.4. The Test Cases page

3. Enter `Test if the printer prints at least 10 page per minute` in the *Name* field. The name provides information about the function/task of the test.

4. Press the `Save` button.

5. Click on the test case ID or on the icon in the action column.

6. Select the *Steps* tab.



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

Step 1: Start printing the test document

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

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

Action: Stop timer when the last page lands in the tray

Expected Result: The number of pages divided by the amount of minutes taken (rounded down) should be 10 or more.

Precondition:

Postcondition:

Created a day ago by Felix Mustermann

Last changed a day ago by Felix Mustermann

Figure 2.5. The Test Case Step tab in the Test Cases page

7. Create the following steps:

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

8. Press the `Save` button.

Next, we create the following test cases, each with a single test case step. First click on the `Back` button to go back to the list of test cases.

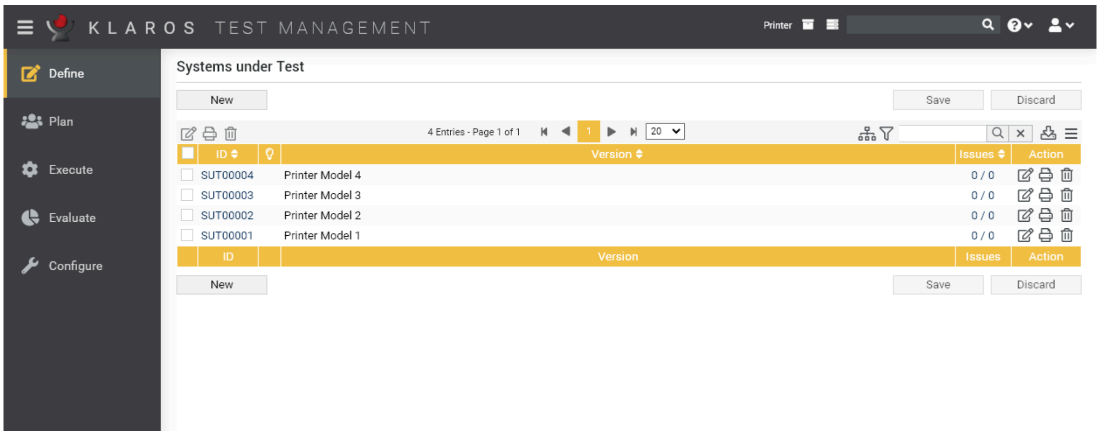
Name	Execution	Step Description	Step Expected Result
`Print test page`	*Manual*	`Print the test page using the Operating - System`	`The page is being printed`
`Detect empty cartridge`	*Manual*	`Insert empty cartridge`	`Printer display shows the Cartridge empty warning`

You should now see three created test cases, one with two test steps and one with three test steps.

2.5. Creating Test Suites

The final step before testing can commence is the definition of test suites. In test cases are assigned to a test suite in order to execute them jointly later. A test suite can contain one or more test cases and test cases can occur in multiple test suites.

1. Under *Define*, select *Test Suites* from the side menu bar.
2. Press the New button.



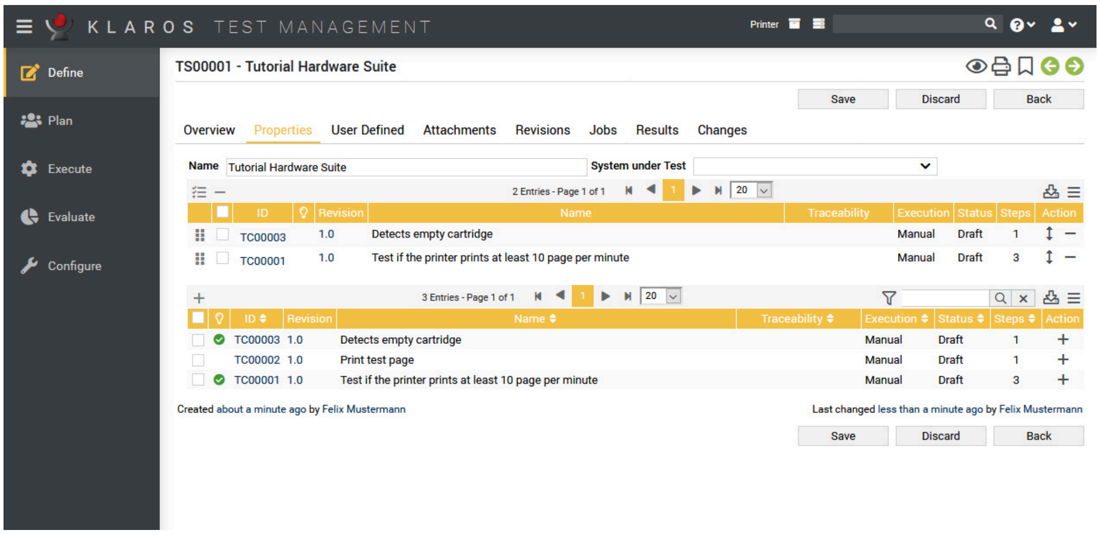
Systems under Test

4 Entries - Page 1 of 1

ID	Version	Issues	Action
SUT00004	Printer Model 4	0 / 0	
SUT00003	Printer Model 3	0 / 0	
SUT00002	Printer Model 2	0 / 0	
SUT00001	Printer Model 1	0 / 0	

Figure 2.6. 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.



TS00001 - Tutorial Hardware Suite

Name: Tutorial Hardware Suite   System under Test:

2 Entries - Page 1 of 1

ID	Revision	Name	Traceability	Execution	Status	Steps	Action
TC00003	1.0	Detects empty cartridge		Manual	Draft	1	
TC00001	1.0	Test if the printer prints at least 10 page per minute		Manual	Draft	3	

3 Entries - Page 1 of 1

	ID	Revision	Name	Traceability	Execution	Status	Steps	Action
✓	TC00003	1.0	Detects empty cartridge		Manual	Draft	1	+
	TC00002	1.0	Print test page		Manual	Draft	1	+
✓	TC00001	1.0	Test if the printer prints at least 10 page per minute		Manual	Draft	3	+

Created about a minute ago by Felix Mustermann

Last changed less than a minute ago by Felix Mustermann

Figure 2.7. The Test Suite page

8. Press the Save button

Our project is now complete, and we can start to execute the test cases.

2.6. Executing Tests

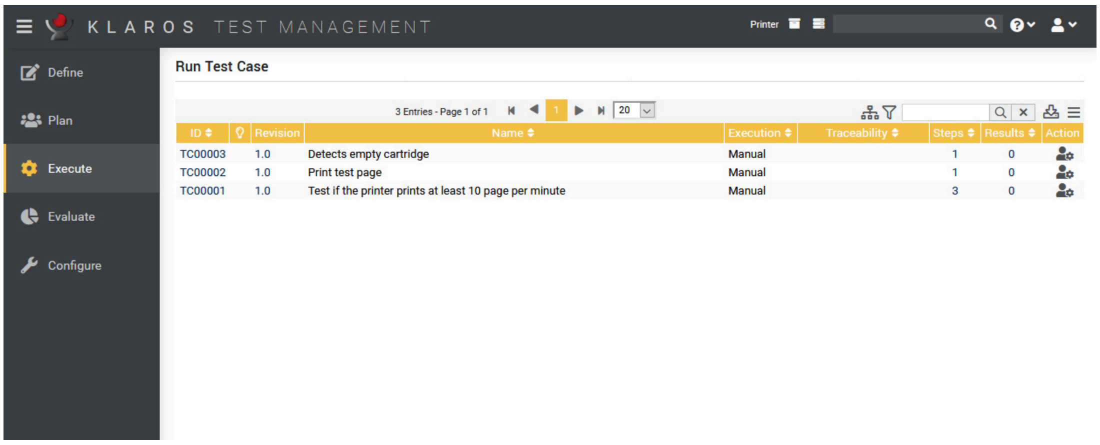
Now that we have created some test cases and a test suite, we are ready for the first test execution.

2.6.1. Executing Single Test Cases

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

1. Under *Execute*, select *Run Test Case* from the side menu bar.

2. Now the page *Run Test Cases* is displayed. For all existing test cases it displays an execution icon in the column *Action*. The appearance of the icon can vary, for manually executable tests the is displayed here.



Run Test Case

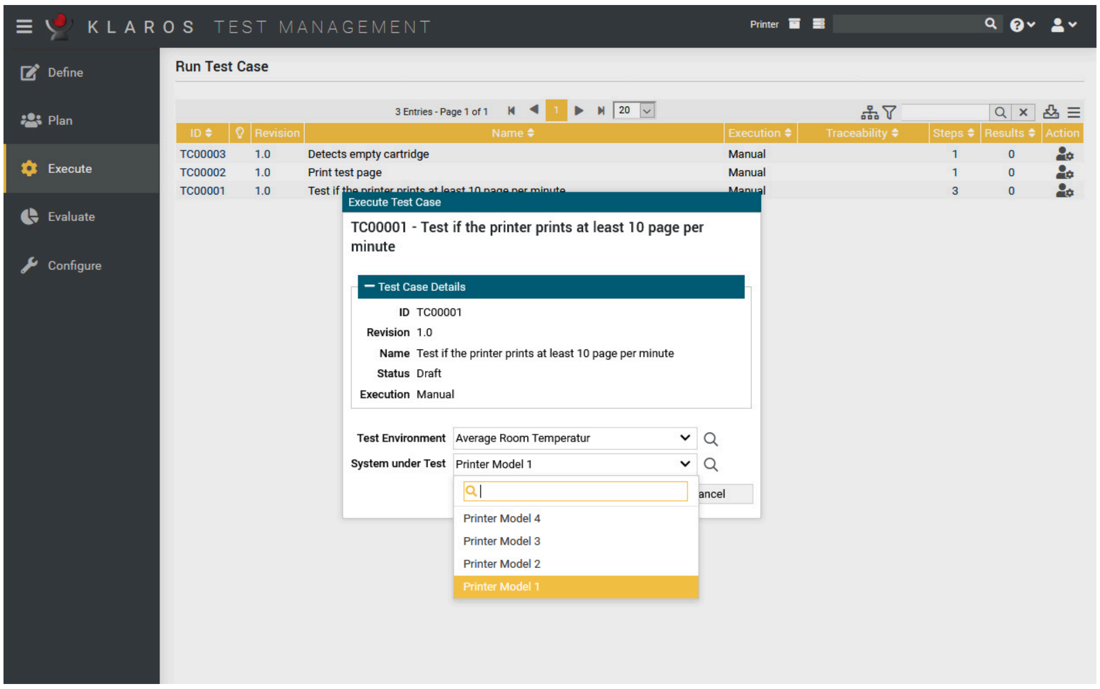
3 Entries - Page 1 of 1 | 20

ID		Revision	Name	Execution	Traceability	Steps	Results	Action
TC00003		1.0	Detects empty cartridge	Manual		1	0	
TC00002		1.0	Print test page	Manual		1	0	
TC00001		1.0	Test if the printer prints at least 10 page per minute	Manual		3	0	

Figure 2.8. The Run Test Case page

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

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



Run Test Case

ID		Revision	Name	Execution	Traceability	Steps	Results	Action
TC00003		1.0	Detects empty cartridge	Manual		1	0	
TC00002		1.0	Print test page	Manual		1	0	
TC00001		1.0	Test if the printer prints at least 10 page per minute	Manual		3	0	

Execute Test Case

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

ID	TC00001
Revision	1.0
Name	Test if the printer prints at least 10 page per minute
Status	Draft
Execution	Manual

Test Environment: Average Room Temperatur

System under Test: Printer Model 1

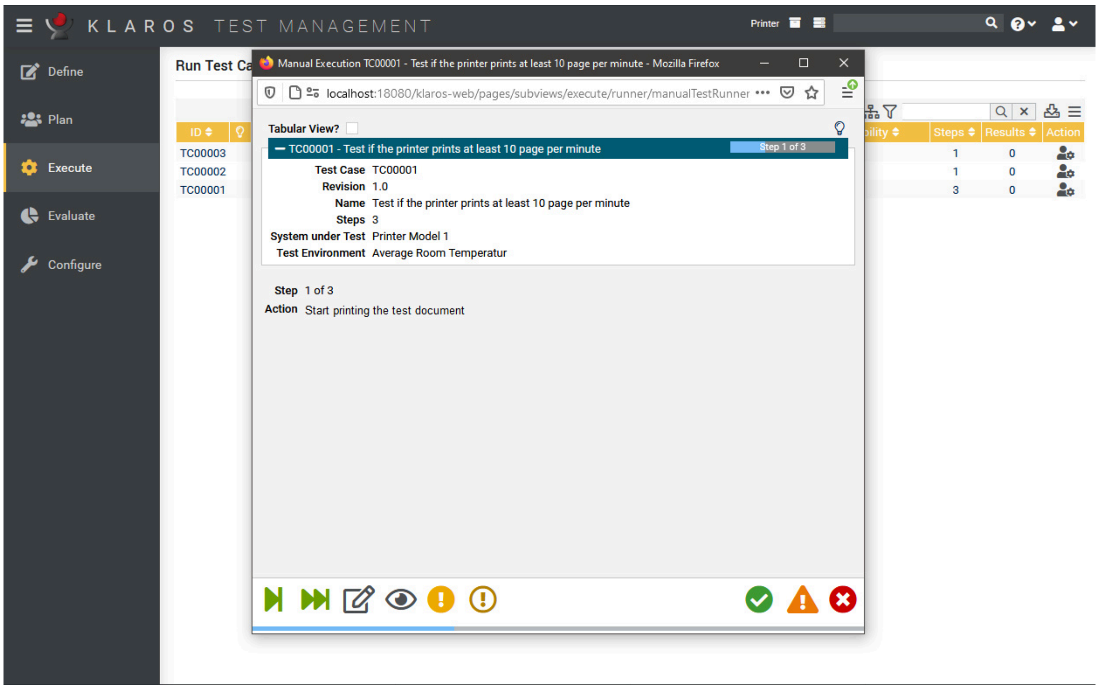
Figure 2.9. The Run Test Case page

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

5. Click the Execute button.

6. A new browser window will open after a message about the current execution. If this does not happen, please check if your browser uses a popup blocker and add the web URL to the exceptions if necessary.

   In the new window the tester is guided step by step through the test procedure. On the first page you can see an overview of the test case. The previous browser window is not required for the test execution and can be minimized.


KLAROS TEST MANAGEMENT
Printer
Define
Plan
Execute
Evaluate
Configure
Run Test Ca
Manual Execution TC00001 - Test if the printer prints at least 10 page per minute - Mozilla Firefox
localhost:18080/klaros-web/pages/subviews/execute/runner/manualTestRunner
Tabular View?
Step 1 of 3
ID
Steps
Results
Action

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

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

Step 1 of 3
Action Start printing the test document

ID	Steps	Results
TC00003	1	0
TC00002	1	0
TC00001	3	0

Figure 2.10. The Test Case Runner

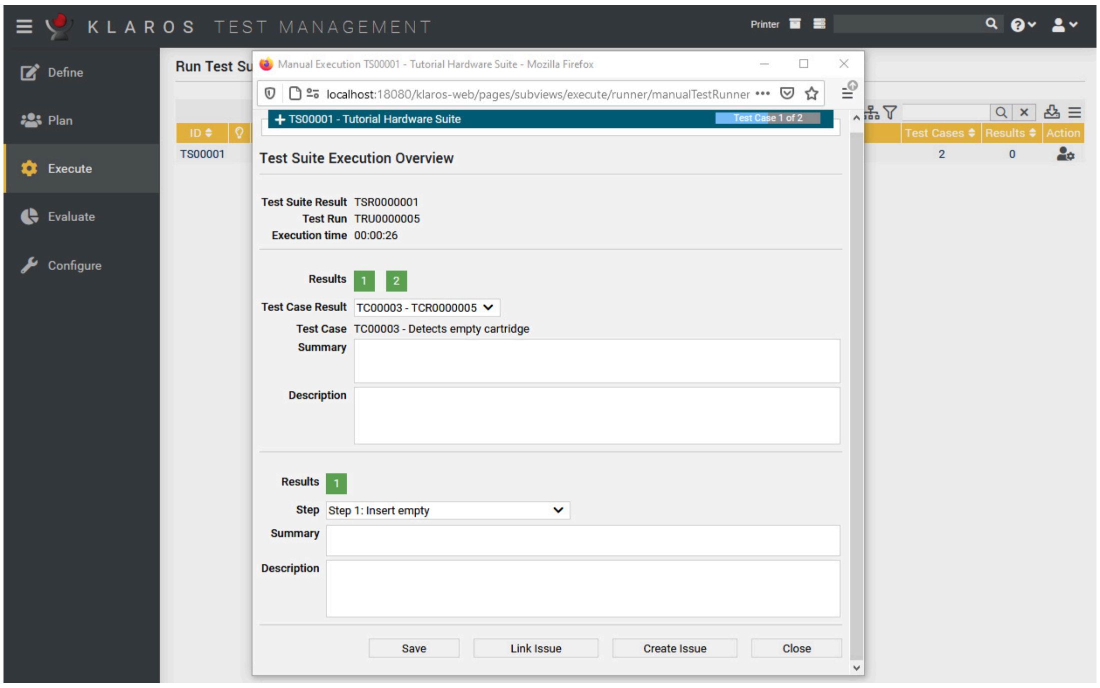
7. The manual test runner will now show the first step. *Action* shows what we as a tester are supposed to do. Let's assume our printer will start printing the test document with no problems, so the first step is passed.

   Press the .

8. Now the second step is displayed. Also mark the further steps as passed. Another dialog will appear in which you confirm that you want to end the test execution. Click the `OK` button.

9. Now the overview of the completed *Test Run* appears. Since all steps passed without complications, it is not necessary to create an issue/defect.

   Click the `Exit` button. Now return to the overview page with the three test cases.



Manual Execution TS00001 - Tutorial Hardware Suite - Mozilla Firefox

localhost:18080/klaros-web/pages/subviews/execute/runner/manualTestRunner

TS00001 - Tutorial Hardware Suite | Test Case 1 of 2

Test Suite Execution Overview

Test Suite Result TSR0000001
Test Run TRU0000005
Execution time 00:00:26

Results 1 2
Test Case Result TC00003 - TCR0000005
Test Case TC00003 - Detects empty cartridge
Summary
Description

Results 1
Step Step 1: Insert empty
Summary
Description

Figure 2.11. The Results Overview in the Test Runner

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

1. Again click the icon on the `Test if the printer prints at least 10 pages per minute` test case.

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

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

4. Press the Execute button.

5. Press the green icon.

6. Press the green icon.

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

   Press the orange icon.

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

   These comments will help to reproduce the failure.

9. Click on the Finish Test button. A dialog appears in which you confirm the completion of the test case execution with OK.

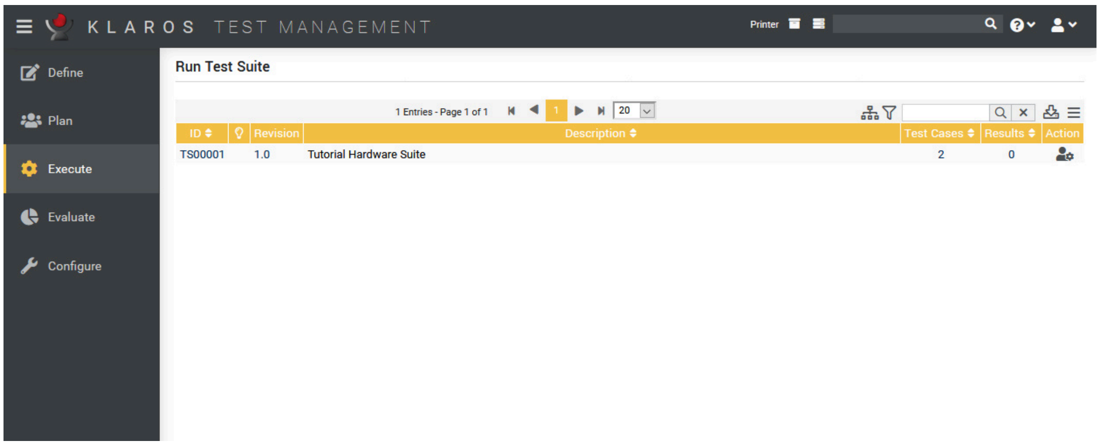
10. If you have set up an issue management system, you can create or link an issue from this page. This process is described in Section 3.6, “Issues”.

11. Press the Finish button.

2.6.2. Executing Test Suites

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

1. Select the entry *Run Test Suites* under *Execute* in the side menu.
2. Click on the icon for the test suite `Tutorial Hardware Suite`.



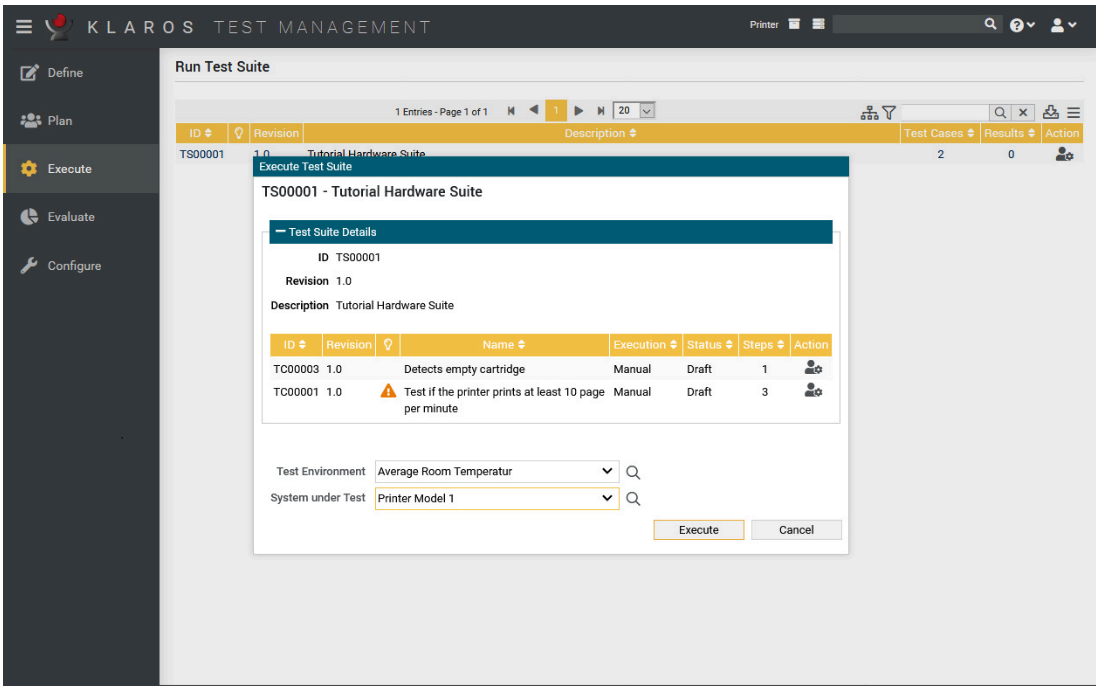
Run Test Suite

1 Entries - Page 1 of 1

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

Figure 2.12. The Run Test Suite page

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



Run Test Suite

Execute Test Suite

TS00001 - Tutorial Hardware Suite

Test Suite Details

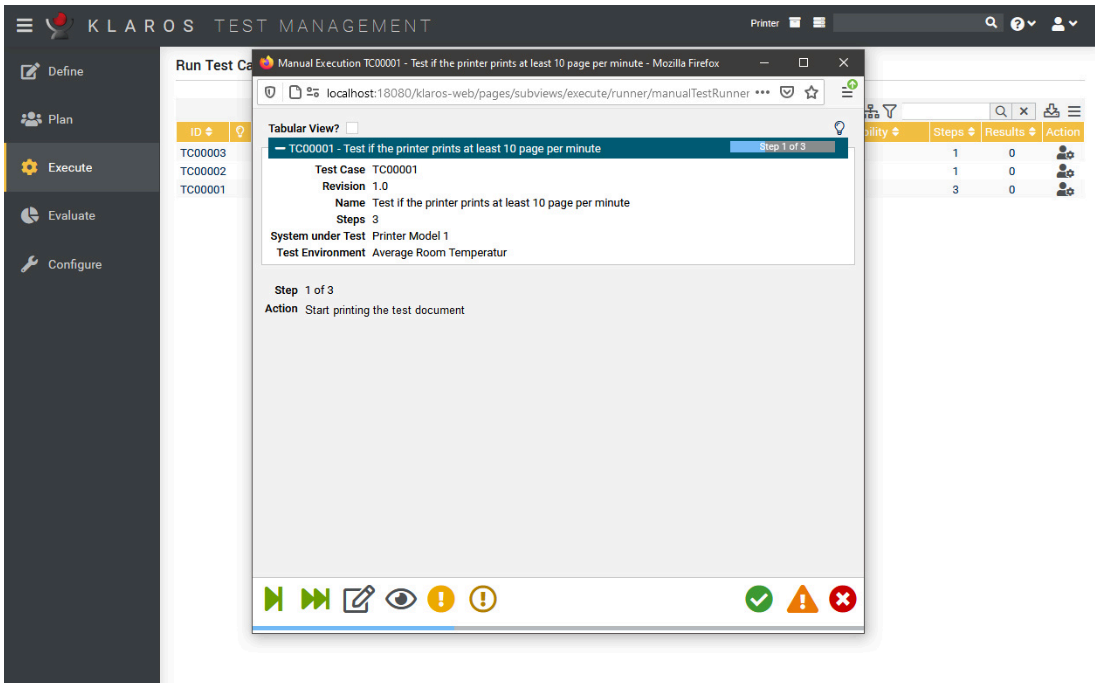
ID TS00001
Revision 1.0
Description Tutorial Hardware Suite

ID	Revision		Name	Execution	Status	Steps	Action
TC00003	1.0		Detects empty cartridge	Manual	Draft	1	
TC00001	1.0	⚠	Test if the printer prints at least 10 page per minute	Manual	Draft	3	

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

Figure 2.13. 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.



Manual Execution TC00001 - Test if the printer prints at least 10 page per minute - Mozilla Firefox

localhost:18080/klaros-web/pages/subviews/execute/runner/manualTestRunner

Tabular View?

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

Step 1 of 3

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

Step 1 of 3

Action Start printing the test document

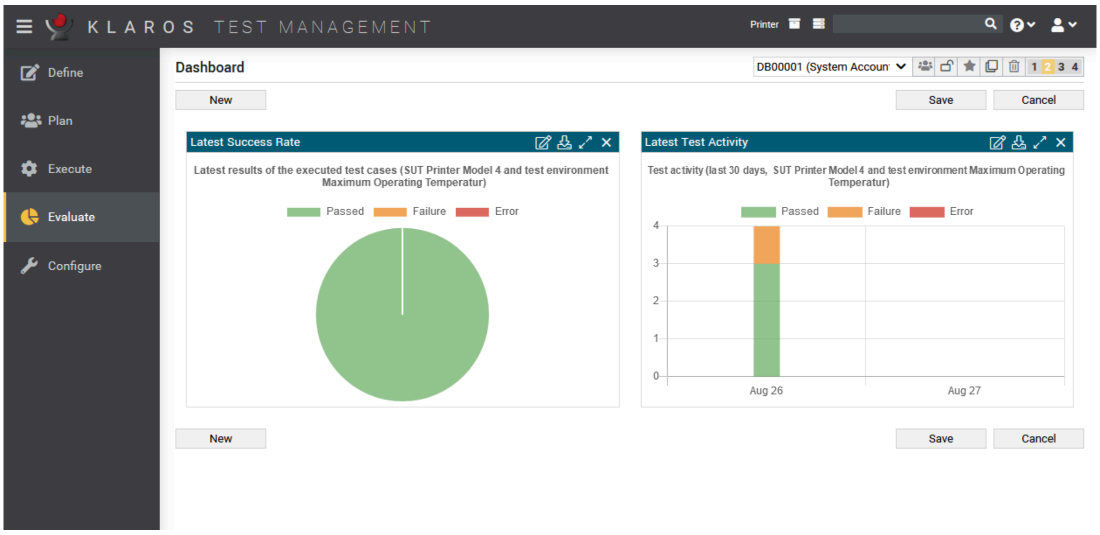
Figure 2.14. The Test Suite Runner

7. Click the green icon for all steps.
8. Confirm the following dialog by clicking the OK button.
9. Repeat for the second test case.
10. Press the Close button.

2.7. Results and Reports

Now we can take a look at the test results. has a comprehensive and detailed reporting system.

1. Log in using the `manager` account.
2. Select the *Printer* project if it is not already selected.
3. Select **Evaluate** in the sidebar.
4. Now the dashboard will be displayed, showing some default reports for the *Printer* project. These show relevant information 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.



Latest results of the executed test cases (SUT Printer Model 4 and test environment Maximum Operating Temperatur)

Test activity (last 30 days, SUT Printer Model 4 and test environment Maximum Operating Temperatur)

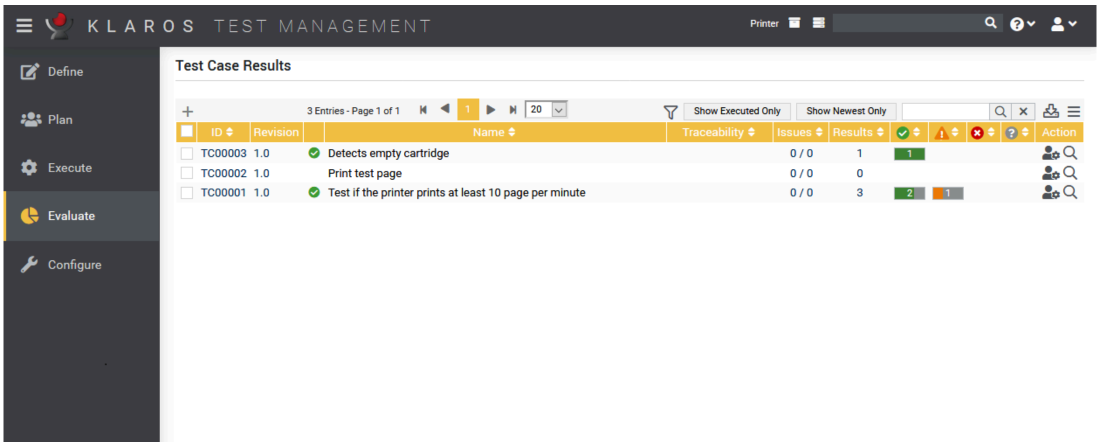
Figure 2.15. The Dashboard

5. The “Latest Success Rate” report can be changed by clicking on the icon. There 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.

   Under *Evaluate*, select *Test Case Results* from the side menu bar.

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.



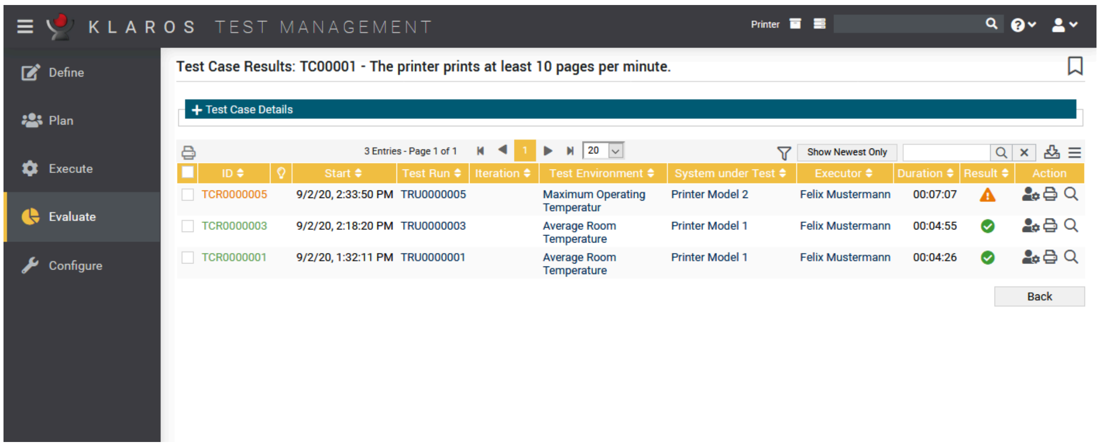
	ID	Revision		Name	Traceability	Issues	Results	✓	⚠	✗	?	Action
	TC00003	1.0	✓	Detects empty cartridge		0 / 0	1	1				
	TC00002	1.0		Print test page		0 / 0	0					
	TC00001	1.0	✓	Test if the printer prints at least 10 page per minute		0 / 0	3	2	1			

Figure 2.16. 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 - The printer prints at least 10 pages per minute.

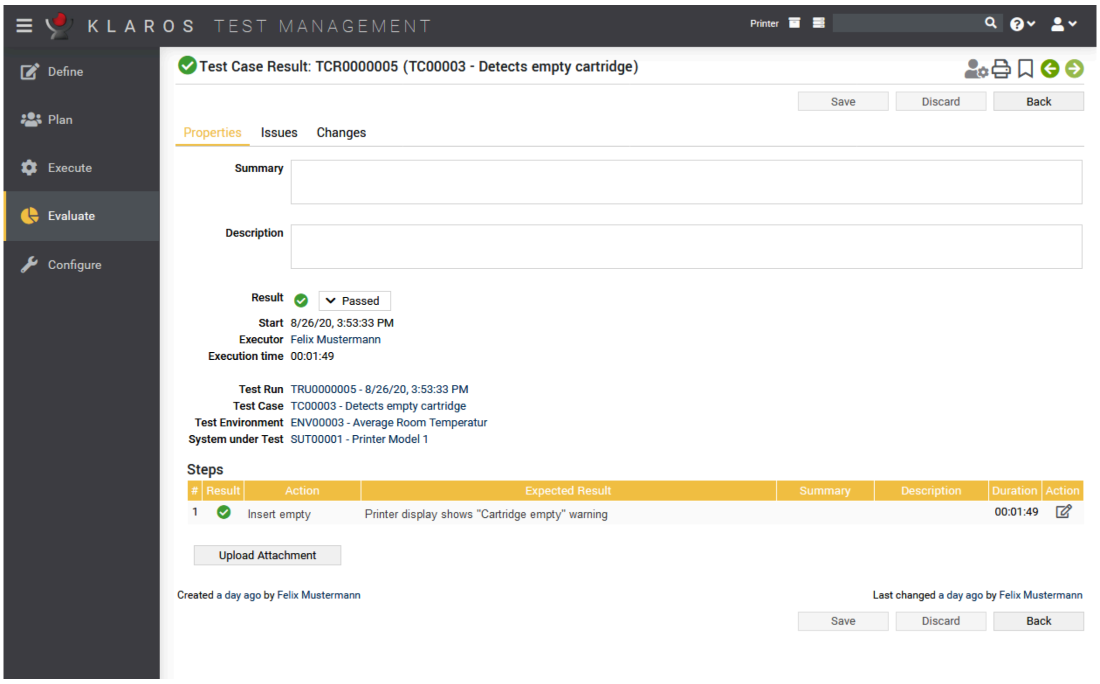
3 Entries - Page 1 of 1

ID		Start	Test Run	Iteration	Test Environment	System under Test	Executor	Duration	Result	Action
TCR0000005		9/2/20, 2:33:50 PM	TRU0000005		Maximum Operating Temperatur	Printer Model 2	Felix Mustermann	00:07:07	⚠	
TCR0000003		9/2/20, 2:18:20 PM	TRU0000003		Average Room Temperature	Printer Model 1	Felix Mustermann	00:04:55	✓	
TCR0000001		9/2/20, 1:32:11 PM	TRU0000001		Average Room Temperature	Printer Model 1	Felix Mustermann	00:04:26	✓	

Figure 2.17. 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.

    Under *Evaluate*, select *Test Suite Results* from the side menu bar.



Test Case Result: TCR0000005 (TC00003 - Detects empty cartridge)

Summary

Description

Result ✓ Passed
Start 8/26/20, 3:53:33 PM
Executor Felix Mustermann
Execution time 00:01:49

Test Run TRU0000005 - 8/26/20, 3:53:33 PM
Test Case TC00003 - Detects empty cartridge
Test Environment ENV00003 - Average Room Temperatur
System under Test SUT00001 - Printer Model 1

Steps

#	Result	Action	Expected Result	Summary	Description	Duration	Action
1	✓	Insert empty	Printer display shows "Cartridge empty" warning			00:01:49	

Created a day ago by Felix Mustermann

Last changed a day ago by Felix Mustermann

Figure 2.18. The Test Case Result page

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

This concludes our quick introduction. The following chapters contain more detailed sections that explain the possibilities of in greater detail.

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 will 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

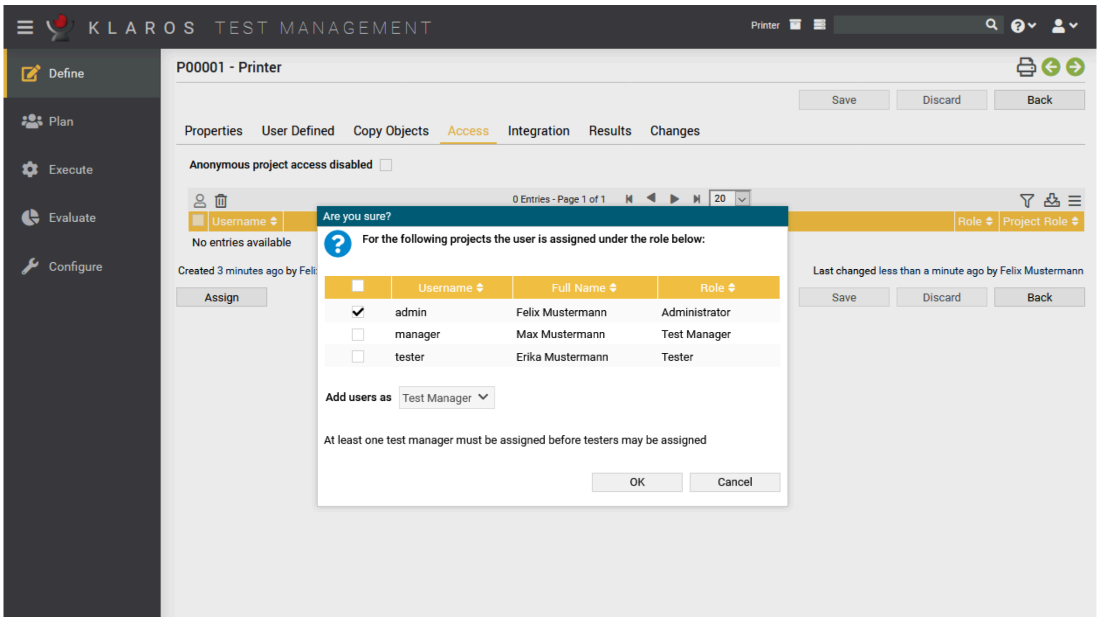
enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

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 sidebar.
3. Select the project *P00001*, or the “Printer Test” project if you already created other projects.
4. Select the *Access* tab.

5. 

Anonymous project access disabled

0 Entries - Page 1 of 1

No entries available

Created 3 minutes ago by Feli

Last changed less than a minute ago by Felix Mustermann

Are you sure?

For the following projects the user is assigned under the role below:

	Username	Full Name	Role
✓	admin	Felix Mustermann	Administrator
	manager	Max Mustermann	Test Manager
	tester	Erika Mustermann	Tester

Add users as Test Manager

At least one test manager must be assigned before testers may be assigned

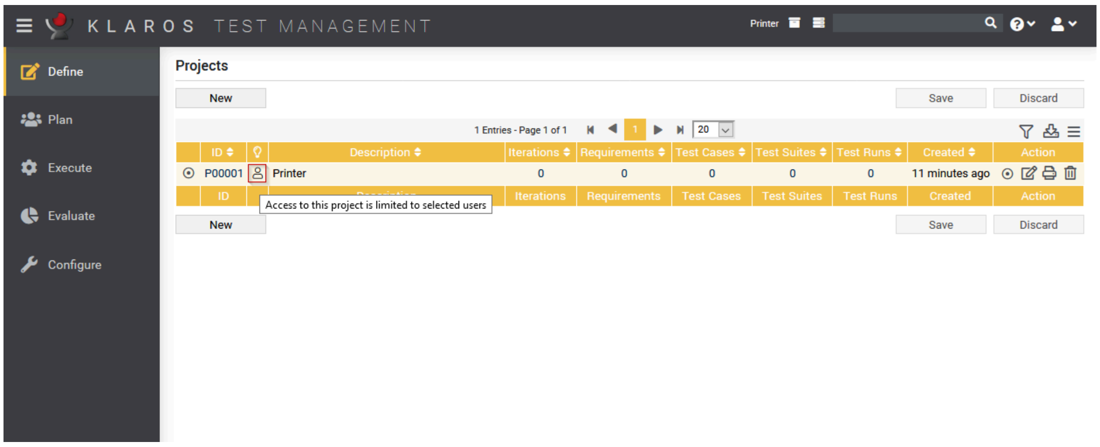
Figure 3.1. The “Access” Tab

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 OK 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.



1 Entries - Page 1 of 1

ID		Description	Iterations	Requirements	Test Cases	Test Suites	Test Runs	Created	Action
P00001		Printer	0	0	0	0	0	11 minutes ago	

Access to this project is limited to selected users

Figure 3.2. A Project with Limited Access

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

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

3. Select the project “Printer Test”.

4. Select the *Access* tab.

5. First we need to make sure, that at least one manager is assigned to the project. Select checkbox in the row containing *Max Mustermann*.

   Click the `OK` button. *Max Mustermann* is now the only user with access to this project.

Important

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

3.2. Categorization

enterprise edition Feature only available in Klaros-Testmanagement Enterprise Edition

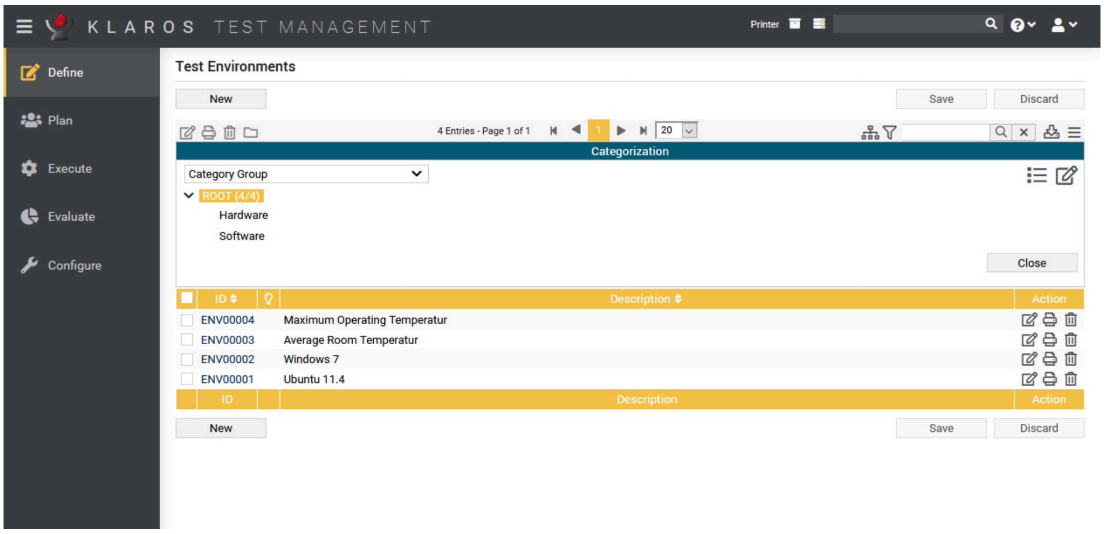
Important

In order to use the examples in this guide it is recommended that you complete Section 2.3, "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.

In Klaros-Testmanagement objects such as test cases, test environments, etc. can be assigned to user-defined categories. This works similar to storing files in folders of a file system. We will now create two categories and assign the created test environments to the respective teams.

1. Click on the icon directly above the table in the workspace to open the Categories panel. Select *Click here to edit categories*.



Test Environments

ID	Description	Action
ENV00004	Maximum Operating Temperatur	
ENV00003	Average Room Temperatur	
ENV00002	Windows 7	
ENV00001	Ubuntu 11.4	

Figure 3.3. Creating Categories

2. Press the icon to create a new category group.

3. Insert `Printer Test` into the field *Name*.

4. Click the `Save` button and select the category *Printer Test* from the drop-down menu *Category Group*.

5. Next, rename the field *ROOT* to *Team*.

6. Now click on the icon in the field *Team* and enter `Hardware` in the text field of the new subcategory. Repeat the step with the entry `Software` .

7. Click the `Save` button.

After the categories have been successfully created, the test environments can now be assigned to the individual teams.

1. To do this, use the icon right above the table to change the view.

2. Select the checkboxes to the left of the software-related test environments (`Windows 10` and `Ubuntu 20.04)` .

3. Click on the icon left above the table.

4. Select the entry `Software` in the displayed dialog.

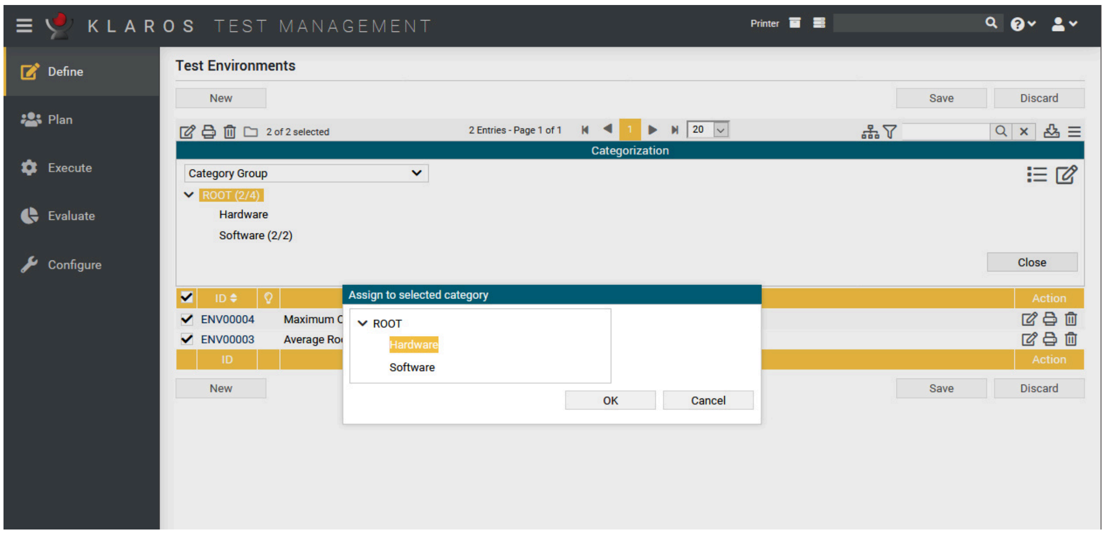


Figure 3.4. Assigning Test Environments to Categories

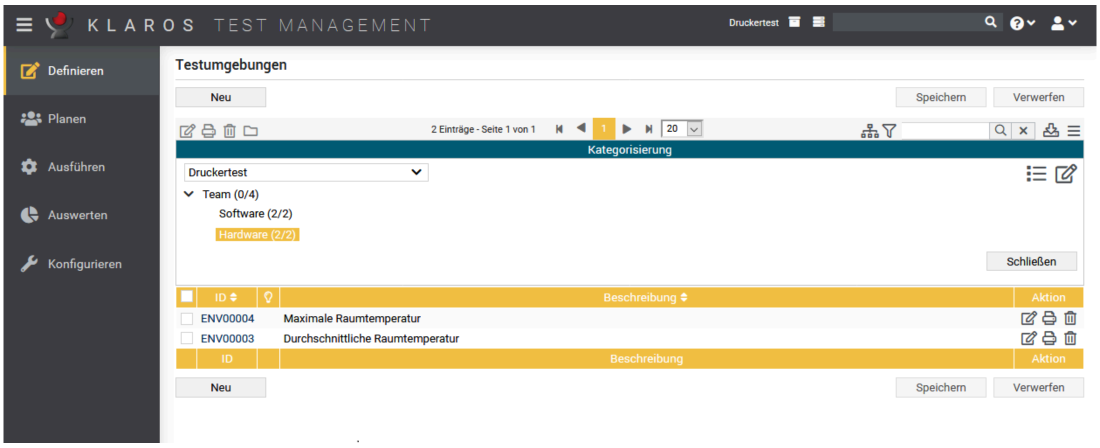
5. Click the `OK` button. The two entries now appear in the category *Software*.

6. Now select the checkboxes next to the hardware-related test environments.

7. Click the icon.

8. Now click on the entry `Hardware` in the dialog.

9. Click the `OK` button.



	ID		Beschreibung	Aktion
	ENV00004		Maximale Raumtemperatur	
	ENV00003		Durchschnittliche Raumtemperatur	

Figure 3.5. The Test Environment Categories

If the categories panel is closed, all existing test environments are displayed. If the panel is open, only the test environments of the selected category are displayed.

3.3. User Defined Properties

enterprise edition 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).

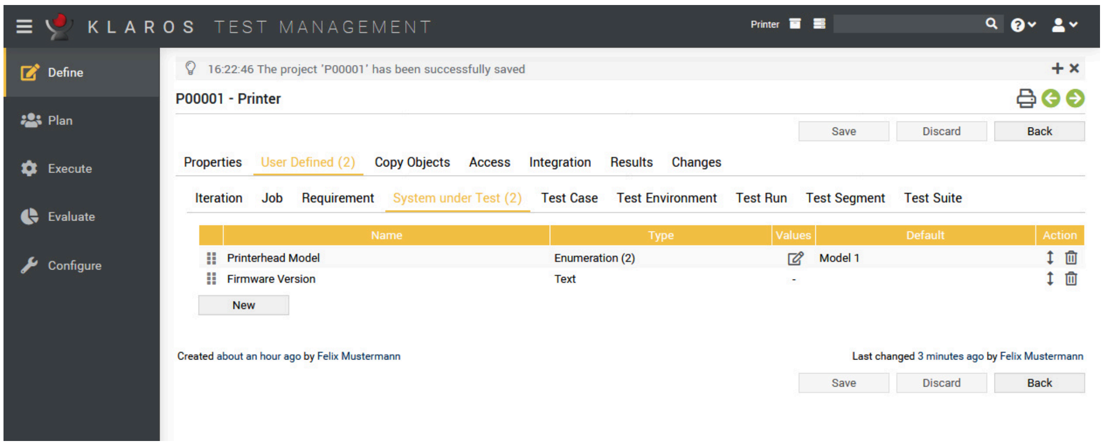
Since the SUTs in our example are printers, there are a number of properties that will make searching through them easier. One of these is the firmware version, an alphanumeric value which is changed relatively often.

The printers used in our example have different properties. These properties can be customized in the tab *User Defined*. You will find a detailed description in Section 3.3, “User Defined Properties”. Here we define the firmware version and the printer head models.

1. Click in the project ID or in the icon in the action column.

2. Select the tab *User defined* and *System under Test* below.

3. Click the New button.


16:22:46 The project 'P00001' has been successfully saved

P00001 - Printer

Save | Discard | Back

Properties | User Defined (2) | Copy Objects | Access | Integration | Results | Changes

Iteration | Job | Requirement | System under Test (2) | Test Case | Test Environment | Test Run | Test Segment | Test Suite

Name	Type	Values	Default	Action
Printerhead Model	Enumeration (2)	✎	Model 1	
Firmware Version	Text	-		

New

Created about an hour ago by Felix Mustermann

Last changed 3 minutes ago by Felix Mustermann

Figure 3.6. Adding User Defined Properties

4. Add the entry `Firmware Version` in the field *Name*. The *Type* is already preset with the desired value *Text*.

5. Click the Save button.

Next, we create 2 printer head models and define an enumeration type for them.

1. Click the New button.

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

3. Select *Enumeration* in the *Type* column. Click on the icon that appears in the *Values* column. A dialog opens.

4. Click the New button and enter the entry `Model 1`.

5. Again click the New button and enter the entry `Model 2`.

6. Click the Ok button and then click Save on the main screen.

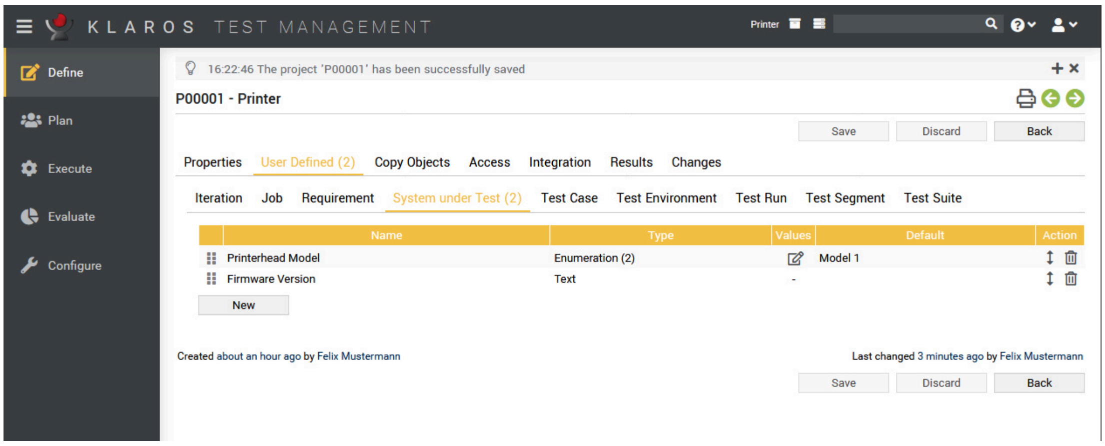
1. Click on the ID for *Printer Model 1* or select the icon in its action column

2. Select the *User Defined* tab.

3. Enter `V23.41.06` in the *Firmware Version* field.

4. Select *Model 1* in the *Printerhead Model* dropdown list.

5. Press the Save button.



16:22:46 The project 'P00001' has been successfully saved

P00001 - Printer

Name	Type	Values	Default	Action
Printerhead Model	Enumeration (2)		Model 1	
Firmware Version	Text	-		

Created about an hour ago by Felix Mustermann

Last changed 3 minutes ago by Felix Mustermann

Figure 3.7. The User-Defined / System under Test page

6. Now click on the green arrow in the upper right corner to switch to the next element. Now printer model 2 is displayed. Repeat the process for the remaining 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`

3.4. Pausing and Resuming Test Cases and Test Runs

Important

In order to use the examples in this guide it is recommended that you complete Section 2.5, “Creating Test Suites” of the quick-start guide.

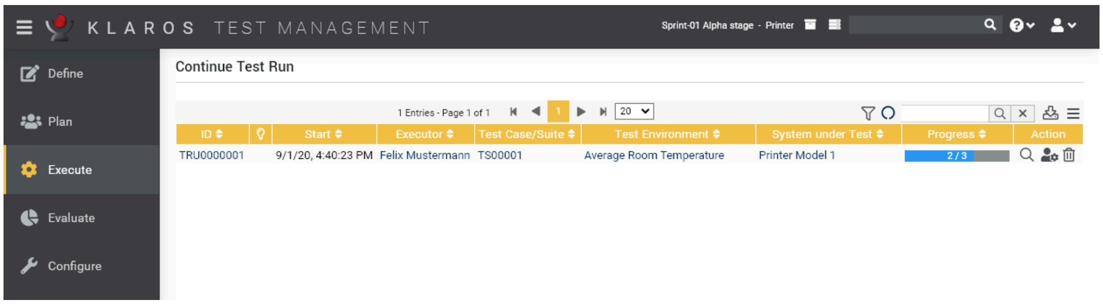
Klaros-Testmanagement supports pausing and resuming of test cases and test runs. If the test runner is cancelled after at least one test step has been completed, the test run is automatically saved in the *Continue Test Runs* section.

1. Login to Klaros-Testmanagement using the *Tester* account.
2. Select the *Printer* project.
3. Under *Execute*, select *Run Test Suite* from the side menu bar.
4. Select the *Tutorial Hardware* test suite.
5. Complete the first test case as explained in Section 2.6.2, “Executing Test Suites”.
6. Press the Cancel button.

   The manual test runner will now close, returning you to the main Klaros-Testmanagement window.
7. 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.

8. 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

1 Entries - Page 1 of 1 · 20

ID		Start	Executor	Test Case/Suite	Test Environment	System under Test	Progress	Action
TRU0000001		9/1/20, 4:40:23 PM	Felix Mustermann	TS00001	Average Room Temperature	Printer Model 1	2 / 3	

Figure 3.8. Continuing a Test Run

3.5. Configuring Issue Management Systems

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

Note

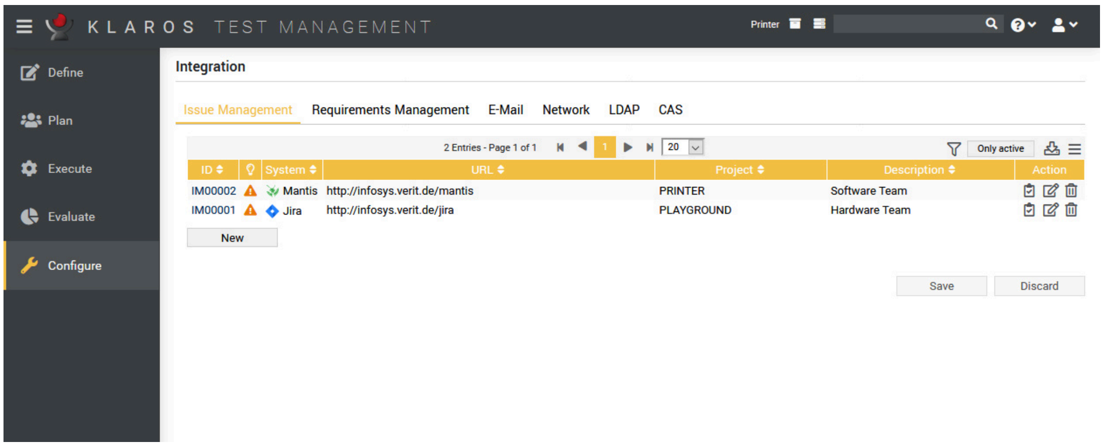
Three user accounts are set up by default during the installation of Klaros-Testmanagement an administrator account, a manager account and a tester account.

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	URL	Project	Description	Action
IM00002	⚠	Mantis	http://infosys.verit.de/mantis	PRINTER	Software Team	
IM00001	⚠	Jira	http://infosys.verit.de/jira	PLAYGROUND	Hardware Team	

Figure 3.9. 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 icon.

7. Press the Save button to save your new issue management system.

Repeat the process, select Mantis instead of JIRA and use `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

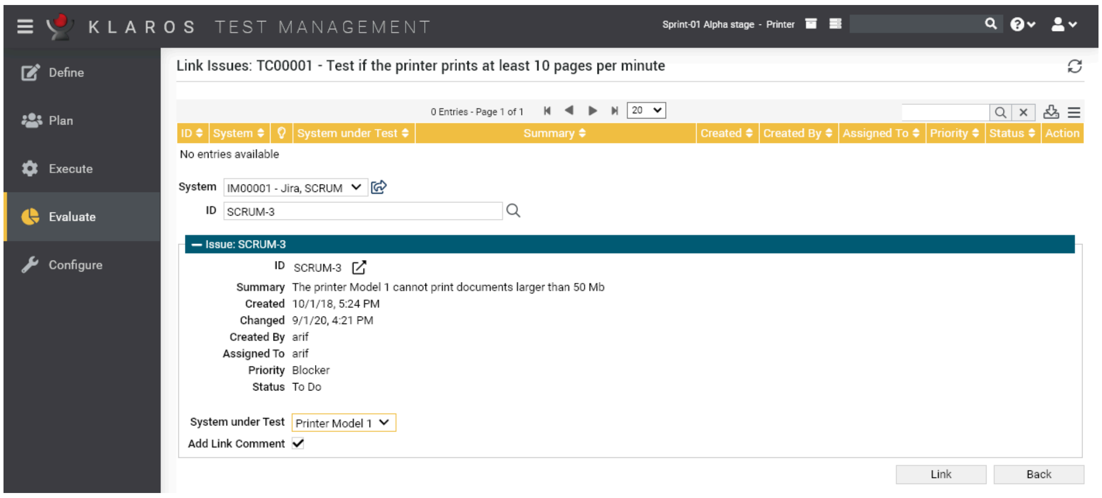
Important

In order to use the examples in this guide it is recommended that you complete Section 2.6.1, "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 Systems") 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 - 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



Link Issues: TC00001 - Test if the printer prints at least 10 pages per minute

0 Entries - Page 1 of 1 | 20

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

No entries available

System IM00001 - Jira, SCRUM
ID SCRUM-3

Issue: SCRUM-3

ID	SCRUM-3
Summary	The printer Model 1 cannot print documents larger than 50 Mb
Created	10/1/18, 5:24 PM
Changed	9/1/20, 4:21 PM
Created By	arif
Assigned To	arif
Priority	Blocker
Status	To Do
System under Test	Printer Model 1
Add Link Comment	✓

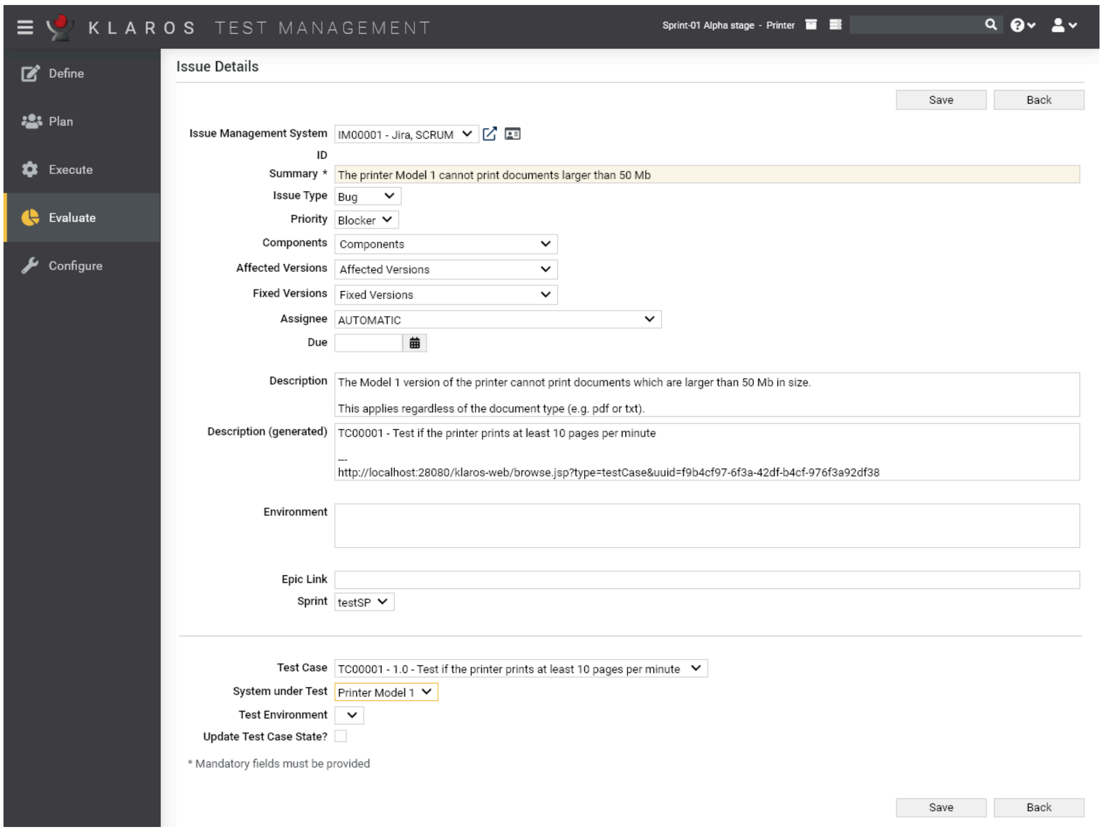
Link | Back

Figure 3.10. Linking Issues

1. Login to Klaros-Testmanagement using the *Tester* account.
2. Under *Evaluate*, select *Issues* from the side menu bar.
3. Press the icon in the action column of the test case *Test if the printer prints at least 10 pages per minute*.
4. Press the **New** button.
5. Select the *Hardware Team* issue management system from the *Issue Management System* dropdown list.
6. Enter a valid issue id in the *ID* text field (this id has to match an existing issue in the IMS).
7. Press the icon next to the *ID* field (Please wait a few moments until the issue has been re-trieved from the IMS).
8. Select the system under test *Printer Model 1* in the *System under Test* dropdown list.
9. Press the Link button.

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

3.6.2. Creating Issues



Issue Details

Issue Management System: IM00001 - Jira, SCRUM
ID
Summary *: The printer Model 1 cannot print documents larger than 50 Mb
Issue Type: Bug
Priority: Blocker
Components: Components
Affected Versions: Affected Versions
Fixed Versions: Fixed Versions
Assignee: AUTOMATIC
Due
Description: The Model 1 version of the printer cannot print documents which are larger than 50 Mb in size.
This applies regardless of the document type (e.g. pdf or txt).
Description (generated): TC00001 - Test if the printer prints at least 10 pages per minute
---
http://localhost:28080/klaros-web/browse.jsp?type=testCase&uuid=f9b4cf97-6f3a-42df-b4cf-976f3a92df38
Environment
Epic Link
Sprint: testSP

Test Case: TC00001 - 1.0 - Test if the printer prints at least 10 pages per minute
System under Test: Printer Model 1
Test Environment
Update Test Case State?

* Mandatory fields must be provided

Figure 3.11. Creating Issues

1. Login to Klaros-Testmanagement using the *Tester* account.
2. Press the **Evaluate** link in the sidebar.
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 2.4, “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 segments, 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 sidebar.
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 OK 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

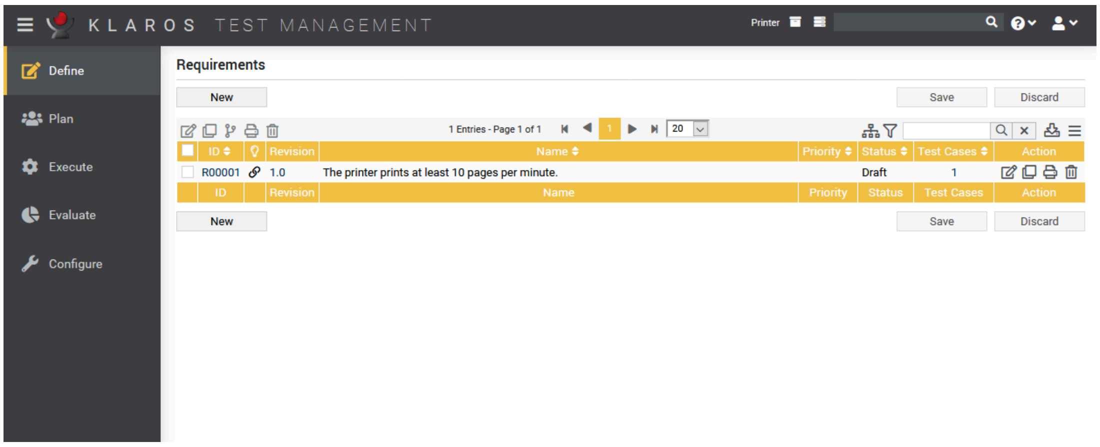
When creating a new revision, this revision gets selected as the active revision for that test case. You can, at any time, switch back to an older revision via the revision tab.

3.8. Requirements

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 section, we will create a requirement, link it to a test case and execute this test case.



ID	Revision	Name	Priority	Status	Test Cases	Action
R00001	1.0	The printer prints at least 10 pages per minute.		Draft	1	

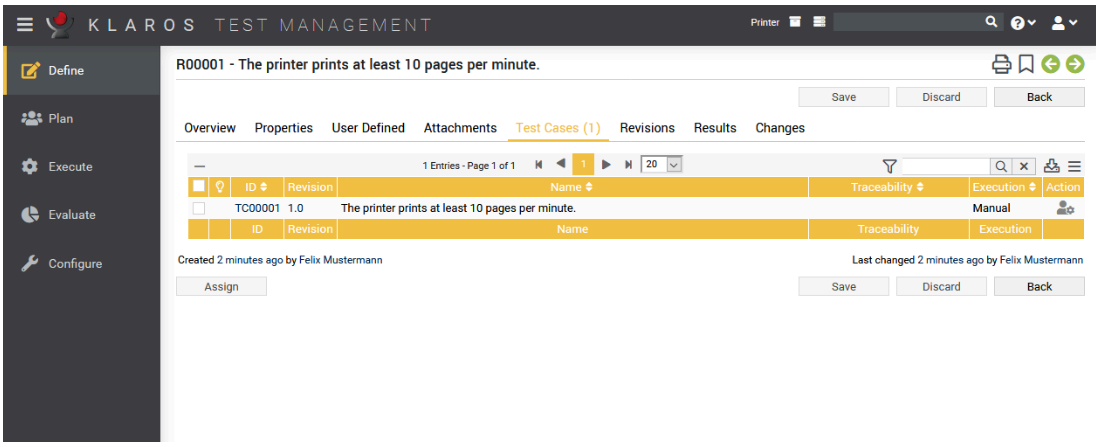
Figure 3.12. The Requirements Page

1. Select the **Define** section in the sidebar.
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 Neu button.
10. Select the test case *Test if the printer prints at least 10 page per minute*.
11. Press the Assign button.

Multiple Requirements

You can assign multiple test cases to the same requirement.

You can also assign test cases to multiple requirements!



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

Overview | Properties | User Defined | Attachments | Test Cases (1) | Revisions | Results | Changes

1 Entries - Page 1 of 1

		ID	Revision	Name	Traceability	Execution	Action
		TC00001	1.0	The printer prints at least 10 pages per minute.		Manual	

Created 2 minutes ago by Felix Mustermann

Last changed 2 minutes ago by Felix Mustermann

Figure 3.13. 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 sidebar.
2. Select the **Run Test Case** menu entry in the sidebar.
3. Press the icon.
4. Press the Execute button.
5. Press the green icon.
6. Press the OK 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 sidebar.
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.

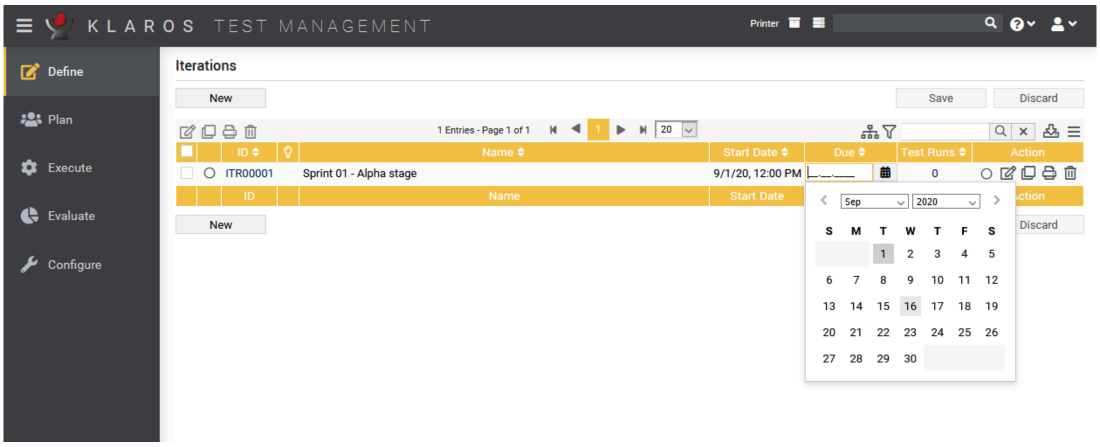
In the results tab, you can see the results of all test cases that are assigned to this requirement. Since we've executed a single test case, there is only one element in this table. Pressing the icon shows you the details of this test case result.

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	9/1/20, 12:00 PM		0	

Figure 3.14. The Iterations Page

1. Select the **Define** section in the sidebar.
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. Select the *Properties* tab.
7. 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.
8. Press the `Save` button.

You have now successfully created a new iteration. Activate the new iteration with the radio button to the left of the iteration ID or to the right in the action column. You can see the active iteration in the topbar (see Section 3.9, "Iterations").

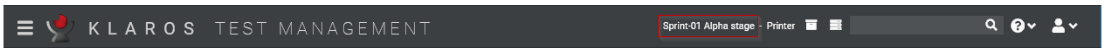
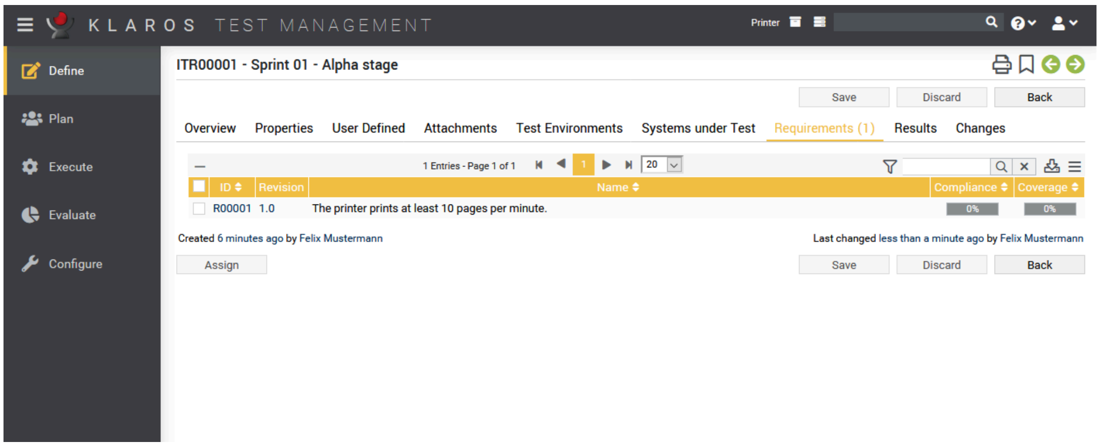


Figure 3.15. The Selected Iteration in the Topbar

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`.



ITR00001 - Sprint 01 - Alpha stage

ID	Revision	Name	Compliance	Coverage
R00001	1.0	The printer prints at least 10 pages per minute.	0%	0%

Created 6 minutes ago by Felix Mustermann

Last changed less than a minute ago by Felix Mustermann

Figure 3.16. The Iteration Details Page

1. Select the *Requirements* tab.
2. Click on the Assign button.
3. Select the requirement *The printer prints at least 10 pages per minute*.
4. 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 deselect 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. Creating custom reports

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 is required to follow 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 *Configure* entry. and *Report Templates* in the side menu bar to get to the overview of custom Reports of the Klaros-Testmanagement Enterprise Edition.

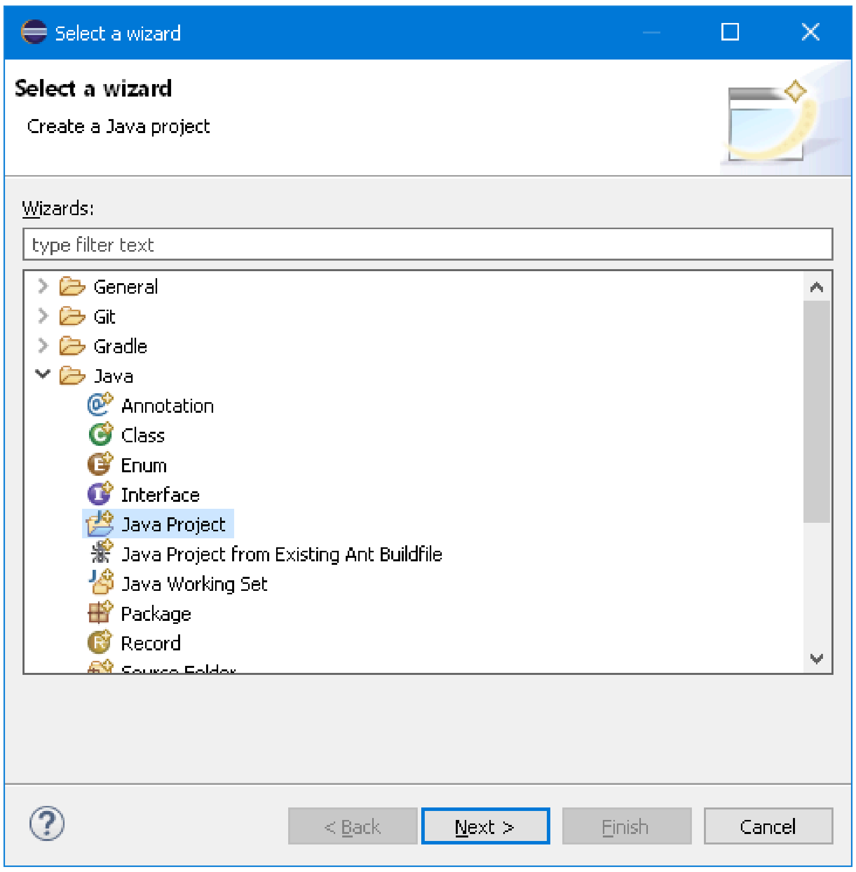
There you can create a new report by clicking on the New button. You can later evaluate the report via the *Evaluate* entry in the top menu bar and the *Report Templates* entry in the side menu bar.

4.1. Development Environment

As a first step, this tutorial shows how to set up the development environment. In this tutorial we set Eclipse as the development environment.

4.1.1. Creating a project to contain your reports

- Start Eclipse and select *File -> New -> Other...*.
- In the dialog select *Java -> Java Project* as shown in the picture, then click on *Next*.



Select a wizard

Select a wizard

Create a Java project

Wizards:

type filter text

- General
- Git
- Gradle
- Java
  - Annotation
  - Class
  - Enum
  - Interface
  - Java Project
  - Java Project from Existing Ant Buildfile
  - Java Working Set
  - Package
  - Record

Figure 4.1. The Wizard Selection Screen

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

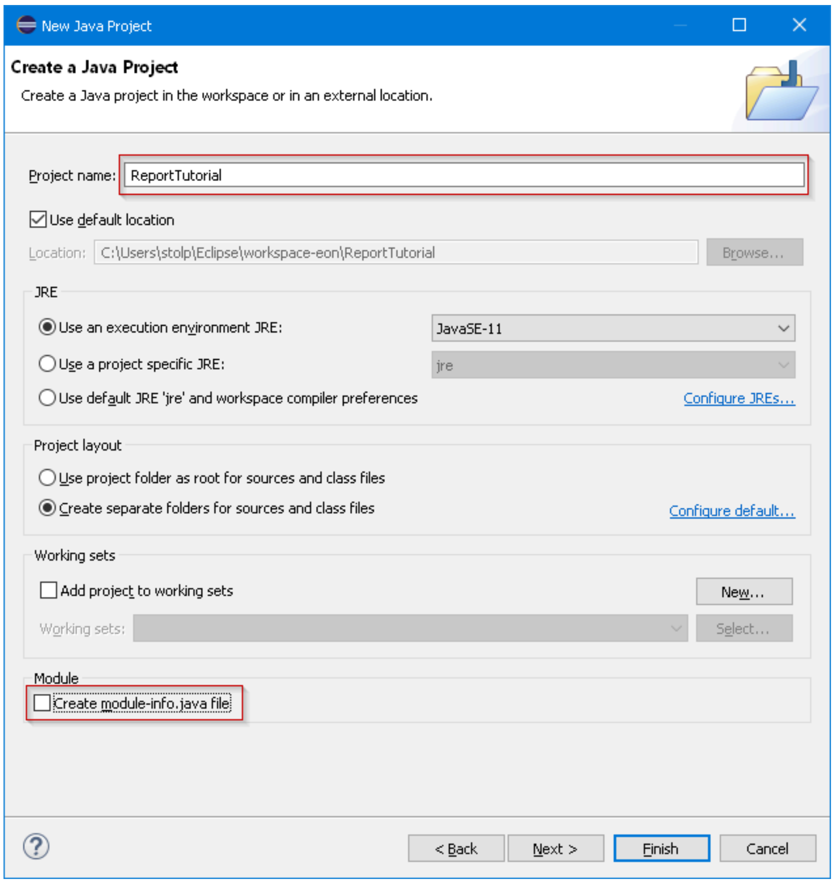

New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: ReportTutorial
☑ Use default location
Location: C:\Users\stolp\Eclipse\workspace-eon\ReportTutorial
Browse...
JRE
● Use an execution environment JRE: JavaSE-11
○ Use a project specific JRE: jre
○ Use default JRE 'jre' and workspace compiler preferences
Configure JREs...
Project layout
○ Use project folder as root for sources and class files
● Create separate folders for sources and class files
Configure default...
Working sets
☐ Add project to working sets
New...
Working sets:
Select...
Module
☐ Create module-info.java file
< Back
Next >
Finish
Cancel

Figure 4.2. The New Project Wizard

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

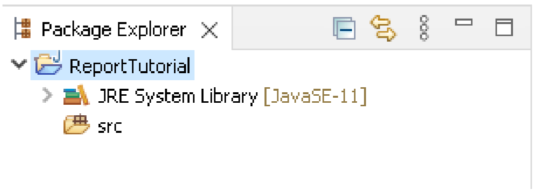


Figure 4.3. The Package Explorer

4.1.2. Setting up the Project

- First, create a folder 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 .

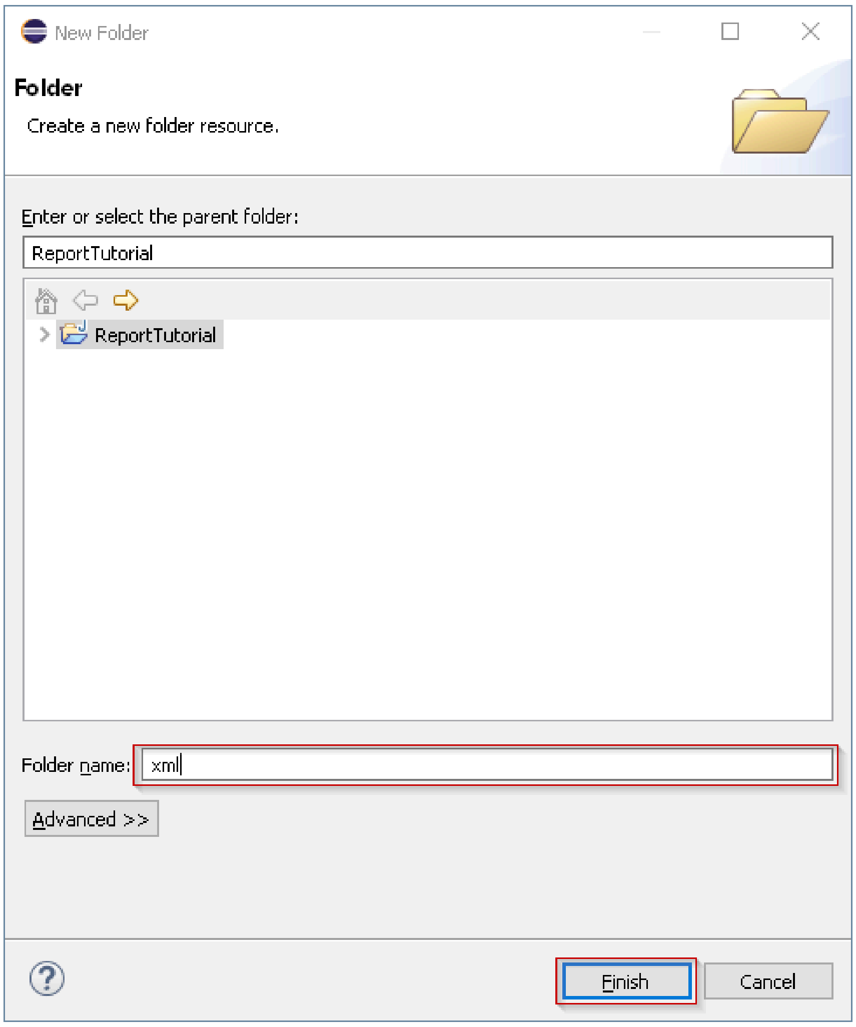


Figure 4.4. Creating a New Folder

- Add the required libraries 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).

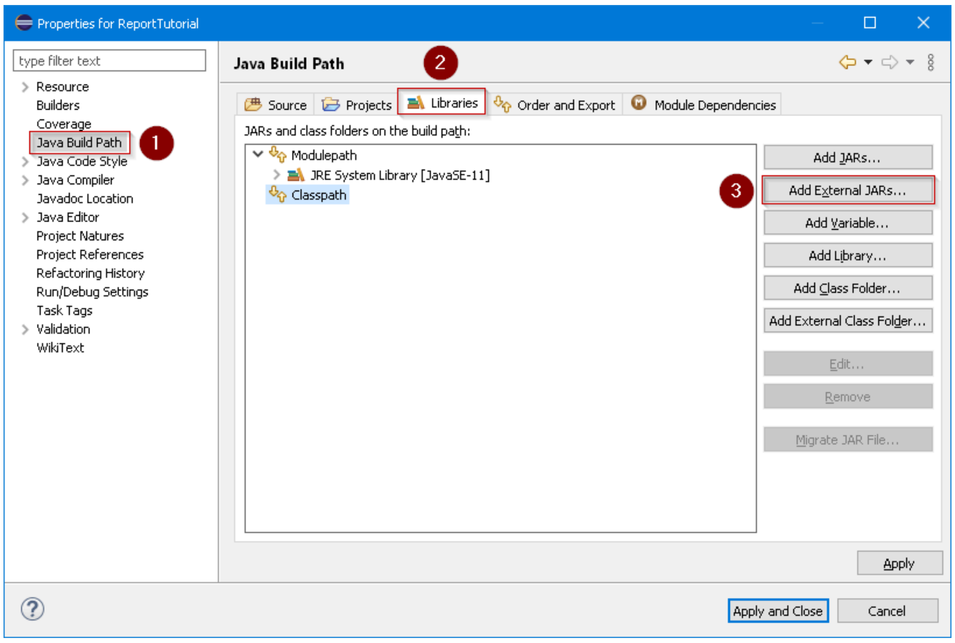
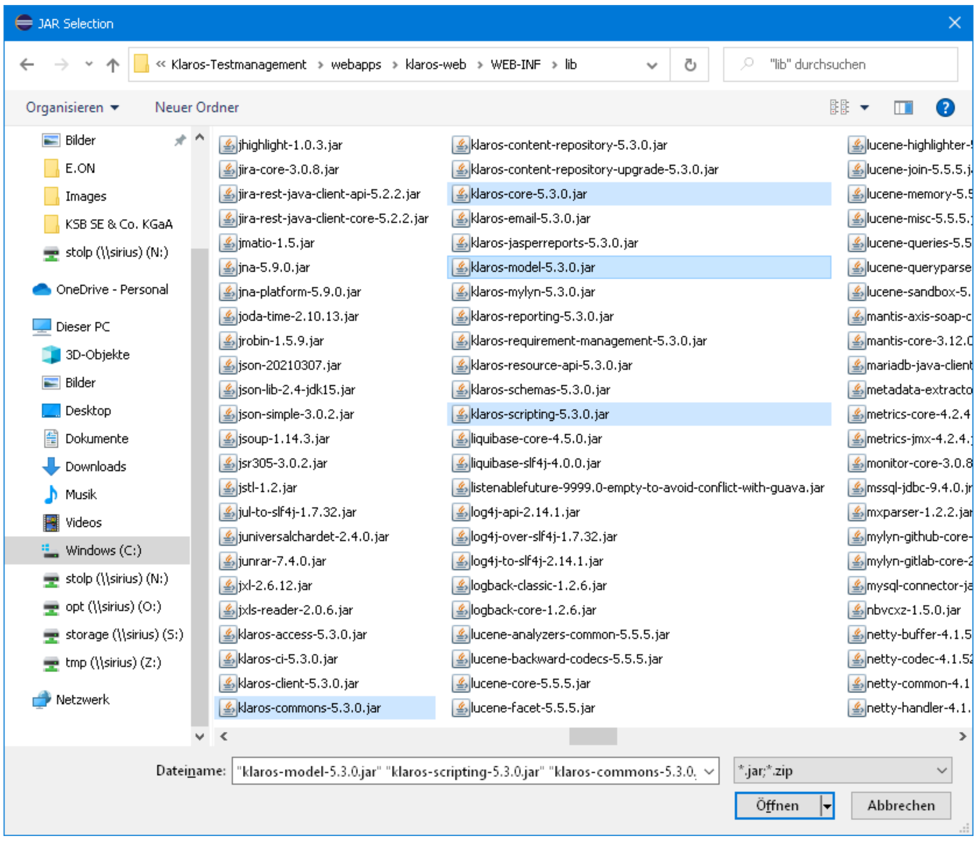


Figure 4.5. The Build Path Setting Screen

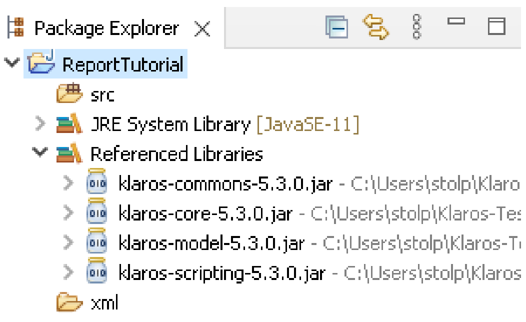
- In the following dialog navigate to your Klaros-Testmanagement installation folder. From there navigate to the *webapps/klaros-web/WEB-INF/lib folder* and select the *klaros-commons-x.y.z.jar*, *klaros-core-x.y.z*, *klaros-model-x.y.z.jar* and *klaros-scripting-x.y.z.jar* files, then click the Open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



Dateiname: "klaros-model-5.3.0.jar" "klaros-scripting-5.3.0.jar" "klaros-commons-5.3.0.
.jar;.zip

Figure 4.6. Selecting an External Jar File

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



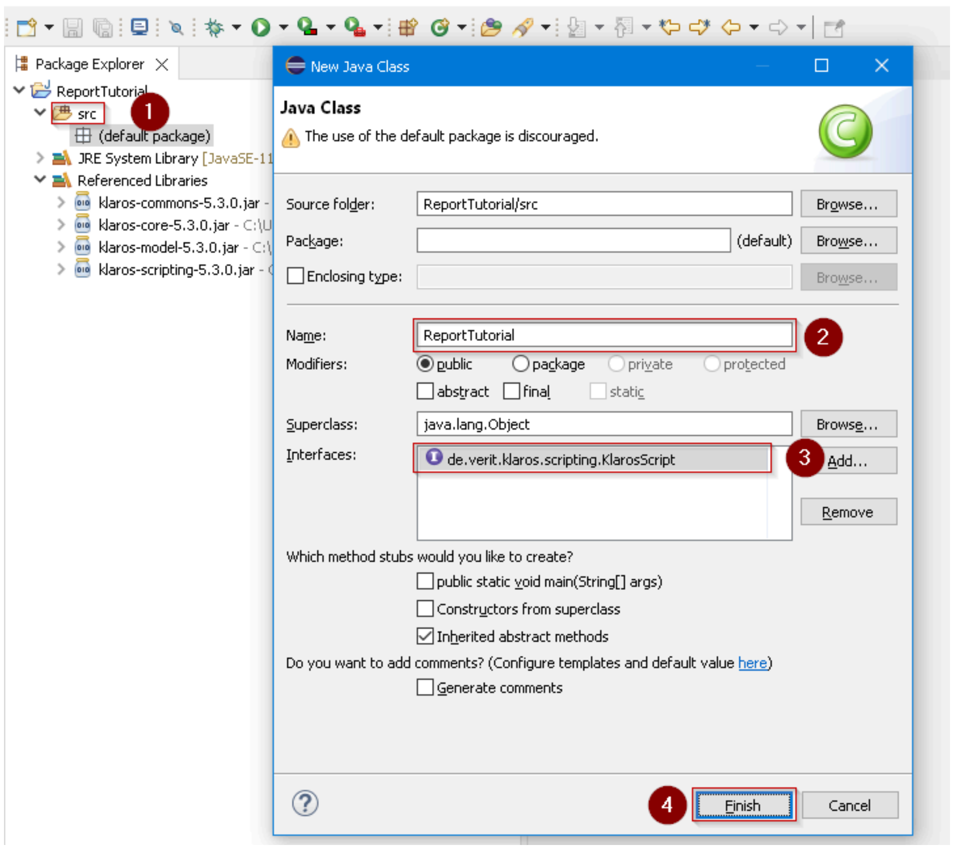
```
ReportTutorial
  src
  JRE System Library [JavaSE-11]
  Referenced Libraries
    klaros-commons-5.3.0.jar - C:\Users\stolp\Klaro
    klaros-core-5.3.0.jar - C:\Users\stolp\Klaros-Tes
    klaros-model-5.3.0.jar - C:\Users\stolp\Klaros-T
    klaros-scripting-5.3.0.jar - C:\Users\stolp\Klaros
  xml
```

Figure 4.7. The Package Explorer

4.2. Preparing the Data for the Report

In the next step, you will see how to prepare your data for your report. The report in this tutorial shows a summary of all test runs for all test cases of a test suite. Furthermore, it presents a description of all executed test steps per test run, sorted by test case. In the Klaros API documentation. you will find an overview of the available data model.

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



Java Class

The use of the default package is discouraged.

Source folder: ReportTutorial/src

Package: (default)

☐ Enclosing type:

Name: ReportTutorial

Modifiers: [✓ public] package 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

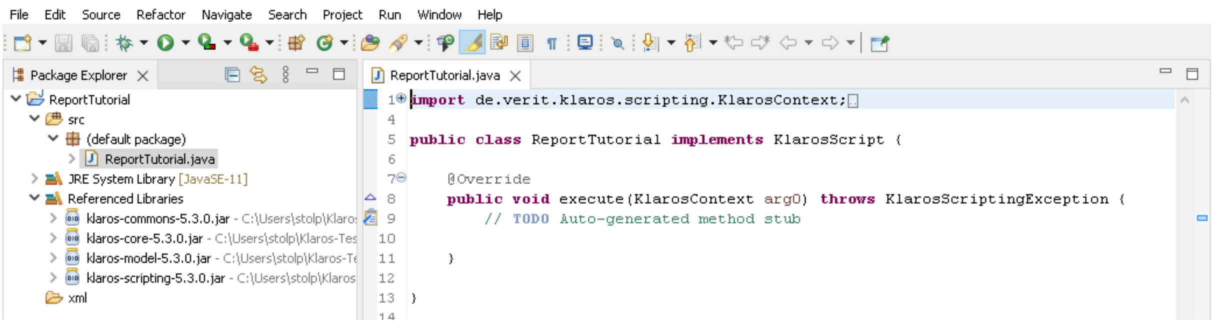
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Figure 4.8. Creating the KlarosScript 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 define the KlarosScript interface for the class. Then click OK to close the dialog.
- Finally press the Finish button (4). After a short while the class should be created and Eclipse should present you the following view.


File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Package Explorer
ReportTutorial
src
(default package)
ReportTutorial.java
JRE System Library [JavaSE-11]
Referenced Libraries
klaros-commons-5.3.0.jar - C:\Users\stolp\Klaro
klaros-core-5.3.0.jar - C:\Users\stolp\Klaros-Tes
klaros-model-5.3.0.jar - C:\Users\stolp\Klaros-Te
klaros-scripting-5.3.0.jar - C:\Users\stolp\Klaros
xml
ReportTutorial.java

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

public class ReportTutorial implements KlarosScript {

    @Override
    public void execute(KlarosContext arg0) throws KlarosScriptingException {
        // 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. The test suite to generate the report for will 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 match the test suite id passed to the KlarosScript and that are present in the currently active project. The result is returned as a list, yet we will only expect a single entry to be returned.

```
        // Build the query using the parameter
        String suiteQuery = "select suite from KlarosTestSuite suite where suite.name='" + suiteName + "'";
        if (context.getActiveProject() != null) {
            suiteQuery += " and suite.configuration.name='";
            suiteQuery += context.getActiveProject().getName();
            suiteQuery += "'";
        }
          List testSuites = context.executeQuery(suiteQuery);
```

- To avoid any errors when executing the KlarosScript, the following code is added for the case that no matching test suite has been found.

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

- Now we add the test-suite to the context so that the template engine can access it later.

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

- The preparations for the pie chart in the report can now 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 suite results belonging to the test suite.

```
 for (KlarosTestSuiteResult testSuiteResult : testSuite.getResults()) {
...
}
```

- Within the test-suite results loop, the test results of each test case are being iterated over. The following code is inserted into the for loop created in the previous step.

```
for (KlarosTestCaseResult testCaseResult : testSuiteResult.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 (testCaseResult.isError()) error.add(testResult);
else if (testCaseResult.isFailure()) failure.add(testResult);
else if (testCaseResult.isPassed()) success.add(testResult);
else if (testCaseResult.isSkipped()) skipped.add(testResult);
```

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

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

- The complete Class can be found in the archive here.

4.3. Creating the Report Layout

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 only 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>
```

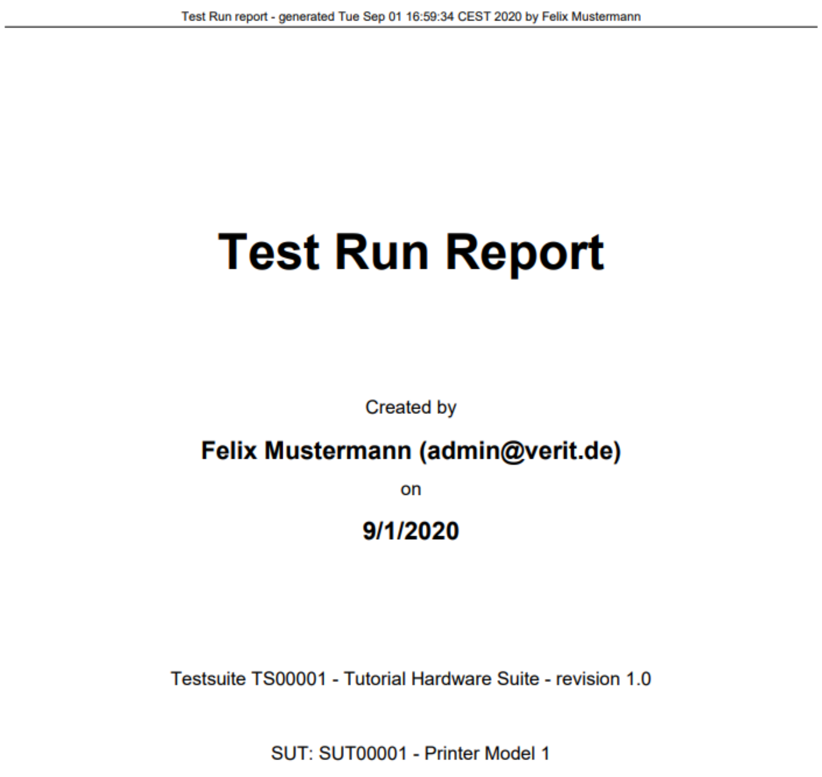
This snippet creates a header that is displayed on every page. It contains the date and the name of the Klaros-Testmanagement user who created the report. The footer contains the page number and an informative text that the report was created with Klaros-Testmanagement. For the header the `borderWidthBottom` attribute was set to provide a separation from the following text. Likewise 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 position the text. To change the font of the paragraphs, you can use the `<p:font>` element and use its `style` and `size` attributes to highlight text.

```
  <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 />
```

The resulting cover sheet can be found below.



Test Run report - generated Tue Sep 01 16:59:34 CEST 2020 by Felix Mustermann

Test Run Report

Created by

Felix Mustermann (admin@verit.de)

on

9/1/2020

Testsuite TS00001 - Tutorial Hardware Suite - revision 1.0

SUT: SUT00001 - Printer Model 1

Figure 4.10. Sample Front Page of a Report

As seen with the header and footer definition, values stored in the context can be accessed by preceding the context variable enclosed in curly brackets with a leading # character, e.g. *#{user.name}*. The context variable for the user is automatically inserted into the context by Klaros-Testmanagement. We used this already when 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 get-method can be accessed. e.g. for `#{testSuite.name}` you will find the method `getName()` in its derived IKlarosLabeledObject interface.

To add a page break to the document, you can use the following:

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

3. Now that we have created 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 test case results in this test suite.

```
<p:paragraph horizontalAlignment="center" spacingAfter="25">
      <p:piechart title="Test Results per Test Run" direction="anticlockwise"
          circular="true" startAngle="30" labelGap="0.1" labelLinkPaint="#000000"
          plotBackgroundPaint="#ffffff" labelBackgroundPaint="#ffffff" 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.

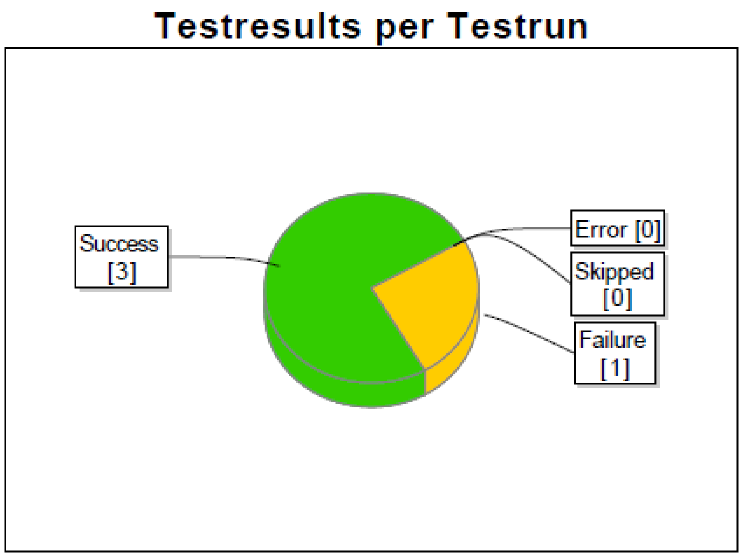


Figure 4.11. A Generated Pie Chart

For a detailed description about how to lay out 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:

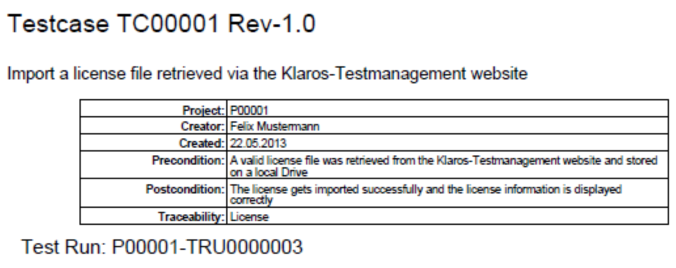
```
context.add("error", error);
context.add("failure", failure);
context.add("success", success);
context.add("skipped", skipped);
```

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 create a separate page for each test case in the test-suite and display the results in a table.

   As this might get a bit lengthy, only fragments of the code 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>
```

We create a loop in the template engine using the *<ui:repeat>* element. The `value` attribute gets assigned a collection, in our case the list of test cases. The `var` attribute defines a variable which can be used within the loop to address the current value. of the current value, e.g. `testCase`. With each pass of the loop, the variable `testCase` gets the next value 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>
```

Now the table is filled with data. For this purpose, it is necessary to define how each table cell should look like. The following code is embedded between the `<p:table>` elements.

```
<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>
    ...
```

The table is supposed to have two columns, thus two cells per row must be specified as well. In the above code, the first cell of a row contains the name of the test project, while the second cell contains the name of the user who created the test case. It is possible to format the text inside a cell as shown by the `<p:font>` element.

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 result available for a test case, a standard text should be displayed instead of an empty cell.

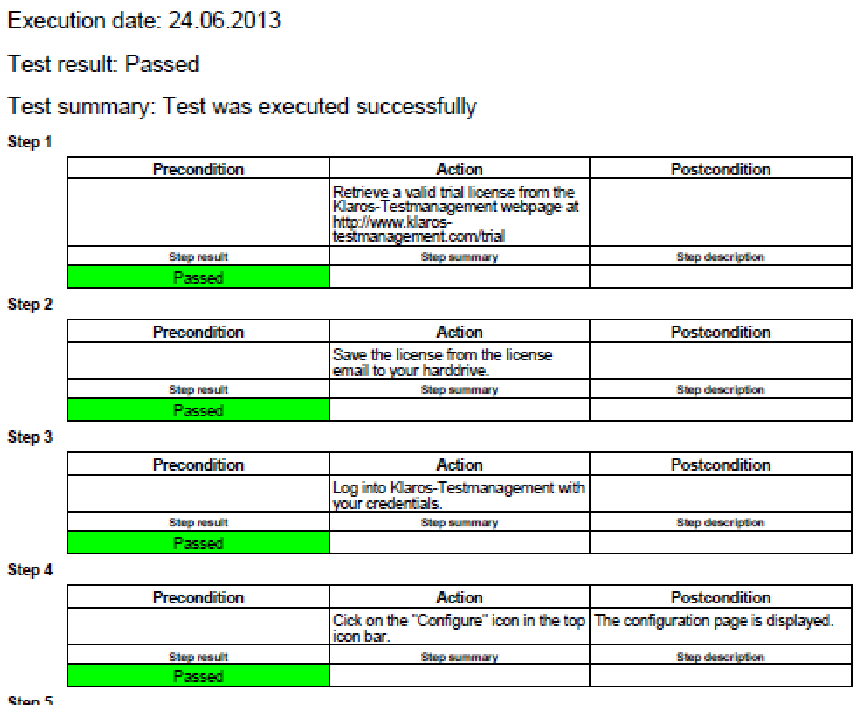
```
  <ui:fragment rendered="#{testCase.results.isEmpty()}">
        <p:font style="normal" size="14">
            <p:paragraph alignment="left" spacingAfter="15"
                indentationLeft="10">
                <p:text value="No test runs found for this test case." />
            </p:paragraph>
        </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 programming languages. In the expression `#{testCase.results.isEmpty()}` the `isEmpty()` method of 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 block will be executed if the list of results has at least one entry.

6. 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
	Cick 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 examine it in the provided source file.

7. For this purpose, another loop must be created within 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 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 shown as step 0.

9. The table should only be displayed if the test case contains at least one test step.

```
<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. Coloring 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 cells are colored by setting the `backgroundColor` attribute to the desired RGB value.

Glossary

A

Administrator	A user role that has full access to all functions. This includes installation and maintenance of the software, updates, system settings, user administration, backups and all rights to create, execute and evaluate tests.
Authentication	Users are usually authenticated via user name and password. In Klaros-Testmanagement the user authentication can be done directly in the software or via an external LDAP or Active Directory directory.
Automated Test Cases	The test results from automation tools are imported via result files into Klaros-Testmanagement. If both manual and automated automated test results are available, they can be evaluated together.

B

Bugzilla	Bugzilla is an open source issue management system. Detailed information can be found on the Bugzilla Web Site. Klaros-Testmanagement provides an interface to Bugzilla.

C

Change Tracking	Complete tracking of all changes to managed objects through automatic logging of modifications. In Klaros-Testmanagement, all changes are highlighted in color on the detail page of the corresponding Object over time.
Category	All objects can be arranged in categories for better overview. Grouping of objects (requirements, iterations, test systems, test environments, test cases or test suites) into a category tree can be done according to self-selected criteria. Multiple categorizations are also possible.
Community Edition	The free edition of Klaros test management. It has a limited feature set, but can also be used freely for commercial purposes. See also Enterprise">Enterprise Edition.
Compliance	The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions [ISO9126].
Compliance Rate	In Klaros Test Management, the conformance rate shows how many test cases assigned to a requirement have been executed with the result "Passed".

Continuous Integration	Continuous Integration (CI) is the automated integration of code changes from multiple contributors into a single software project. It is an important DevOps best practice that enables developers to regularly merge code changes into a central repository where builds and tests are then executed. A plug-in for the Jenkins Continuous Integration Server automatically transfers automatically transfers the test results of a build to Klaros Test Management, where they can be are evaluated.
Coverage, Requirements Coverage	The requirement coverage represents the ratio of requirements covered by tests to the total set of defined requirements. In Klaros Test Management, requirements can be directly entered and assigned to test cases. assigned to test cases. Individual requirements can be covered by several tests and one test can cover more than one requirement.

D

Dashboard	A dashboard displays a collection of information, typically in the form of charts. In Klaros-Testmanagement dashboards display the evaluations for a specific test project. The displayed diagrams can be individually combined to form a dashboard. Each user can compile as many dashboards as desired. Dashboards can be private (visible only to the creator) or public. Administrators can create default dashboards.
Data Base	A database is a collection of information organized in interrelated tables of data and specifications of data objects. Klaros-Testmanagement requires a database system to store the objects and is delivered with a preconfigured Apache Derby database. Other supported database systems are MariaDB, Microsoft SQLServer, MySQL and PostgresSQL.
Details Page	Each object (such as test cases, requirements, iterations, etc.) has its own detail page with various other subviews. These vary depending on the object and show detailed information for example " Properties", "Attachments", "Changes" and "Results" an.
Docker	Docker is free software for isolating applications through container virtualization. The containers contain all necessary packages and can thus be easily transported and installed as files. Klaros Test Management can also be run as a container within a Docker-based environment. Ready-to-use Docker images for various databases are already available. already available. Detailed documentation is available at GitHub.

E

Enterprise Edition	The commercial edition with full support is licensed per user. The minimum installation supports 3 users. A free 30-day demo can be requested Demo.
Environment Variables	Environment variables are dynamically named values that can affect the behavior of running processes on a computer. They are part of the environment in which a process runs. By setting the environment variable KLAROS-HOME before the start of the application, the destination of the Klaros home directory can be moved to another location.
Error (Verdict)	A test execution error occurs when the system cannot execute the test correctly. It should not be confused with a failure.
Evaluation	The type of test result evaluation: manual or automated.

F

Failure (Verdict)	A test is deemed to fail if its actual result does not match its expected result. (ISTQB). It should not be be confused with an error.
Functional Test Design Technique	Procedure to derive and select test cases based on an analysis of the specification of the functionality of a component or system without reference to its internal structure. (ISTQB), also known as black box and white box tests.

G

Guest	A user role with full, but read-only access to all data. A guest can view and save reports in various formats. This role is intended for e.g. project managers, customers or reviewers/reviewers.

I

ID	The ID of an object is created automatically when it is created and cannot be changed later. All IDs consist of an abbreviation for the respective object (e.g. TC for test case, ITR for iteration) and an ascending number. number. “SEG00025” would therefore be the test segment 25.
Inconclusive (Verdict)	A test is inconclusive if it is unclear, if its actual result matches its expected result.
Integration	Integration refers to the connection of external systems. Klaros-Testmanagement has interfaces for integration with issue and requirements management systems, test automation and load testing tools, and and continuous integration servers. There are also interfaces to authentication and e-mail servers.

Issue	An issue is a partial aspect that is required to complete a piece of work. This can be a bug, a requested feature, a task, missing documentation or something similar. An issue is tracked in an issue management system. Klaros-Testmanagement has interfaces to several issue management systems. This means that errors found can be transferred directly to the connected IMS, without having to leave the application.
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) In Klaros test management supports agile development principles and manages iterations as dedicated objects.

J

Java	In order to run Klaros-Testmanagement, a Java runtime environment is required. This is delivered with the installation file and used automatically.
JavaScript	JavaScript is a programming language used in web browsers and is required for running Klaros test management. When using JavaScript blockers such as NoScript or similar. When using JavaScript blockers such as NoScript or similar, the execution of JavaScript for Klaros-Testmanagement must be allowed as an exception.
JIRA	JIRA is a popular application by the company Atlassian Software for defect management, troubleshooting, and project management. Klaros-Testmanagement has an integration with JIRA.
Job	Test activities are planned and logged by means of jobs. The execution and results of the executed jobs are automatically logged and used for evaluation.

L

Latest Success Rate, Report Diagram	The “Last Success Rate” report displays the latest results of executed test cases for the selected combination of test system (one or more) and test environment (one or more) as a chart on the dashboard.
Layout Template	Besides the script, the layout template is the other elementary part of a report template. It describes the structure of the generated report document in an XML-based description language. Layout templates can be adapted to individual requirements.
Log Panel	The log panel displays status messages such as warnings or information are displayed. By default, only the last status message is displayed. Clicking on the + icon opens the history.

M

Mantis	Mantis (MantisBT, Mantis Bug Tracker). is an open source application for defect management and issue tracking in software development. Klaros test management provides integration with Mantis.

N

Notification Event	Notification events may trigger an automatic e-mail notification to specific users. These are sent out as soon as the designated event has has occurred. Possible notification events are, “Job assigned”, “User account created”, “Test execution failed”, “User account password changed” and “Job ready for execution”.

O

Object	The objects defined in Klaros-Testmanagement are project, test case, test step, test segment, test step result, test case result, test suite, test suite result, test environment, system under test, test run, requirement, iteration and job.
Overview Page	Each object type has an overview page. There the individual objects, such as test cases or iterations, are displayed in tabular form. New objects are also created here created.

P

Passed (Verdict)	A test is passed if its actual result matches its expected result. (ISTQB)
Precondition	Environmental and state conditions that must be fulfilled before the component or system can be executed with a particular test or test procedure. (ISTQB).
Postcondition	Environmental and state conditions that must be fulfilled after the execution of a test or test procedure. (ISTQB)
Print View	All objects can be rendered in a printout-optimized representation. The level of detail can be specified with various parameters.
Project	A project is the main object in Klaros Test Management and contains all other objects assigned to a test project.
Project Overview, Report Diagram	The “Project Overview” report displays the number of executed/not executed test cases or execution definitions and the number of scheduled/executed jobs as a chart on the dashboard.

Q

QF-Test	QF-Test from Quality First Software is a cross-platform GUI test automation tool.

	Klaros-Testmanagement integrates with QF-Test and can import the automatically generated test results for further processing.

R

Redmine	Redmine is an open source application for defect management and issue tracking in software development. Klaros-Testmanagement integrates with Redmine.
Reports	Reports provide information on the status of testing activities, the condition of the software, indications of critical components and enable forward planning. For this purpose, the data from the database are sorted according to certain criteria and prepared in textual and graphical overviews. Examples of reports are “Project overview” and “Test activity” as well as overviews of test progress, test results and test runs. Klaros-Testmanagement contains numerous predefined, basic reports. Individual reports can be created by the user. All reports are exportable to various formats such as PDF or Excel (also for further editing).
Report Template	A report template predefines a report that can take report parameters and be customized by the user as needed. It consists of a *Script* and a *Layout Template*. With report templates, ready-made reports are available which can be customized by the user as needed.
REST (Representational State Transfer)	Representational State Transfer (REST) is a software architecture style that uses a subset of HTTP. It is commonly used to provide web services. Klaros-Testmanagement offers several integration interfaces based on REST. These include read access to all stored information as well as import interfaces for test results, test cases and requirements.
Requirement	A user-required property or capability that software must meet or possess in order to Comply with a contract, standard, specification, or other formal document. [According to IEEE 610]. Requirements can be managed either directly in Klaros Test Management or or synchronized with external sources such as JIRA. A reference to the corresponding requirement, e.g. to a document, can be added to test cases and test suites (traceability).
Result, expected	The behavior predicted by the specification, or another source, of a component or system under specified conditions (ISTQB).
Revision	If a major change is made to an object and the previous version is still needed, a new revision should be created. Only selected objects such as test cases or requirements support this mechanism.

Role	A user role defines the rights of the user. Roles can be defined globally and project project-specific. The following user roles are provided: “Administrator”, “Test Manager”, “Tester” and “Guest”.

S

Selenium	Selenium is a web browser automation tool and is mainly used for testing web apps. Selenium produces JUnit XML compliant test results that can be imported into Klaros test management for further processing.
Script	The script is an elementary component of a report template along with the layout template. It extracts the data intended for the report and is available as a Java class. Scripts can be adapted to individual requirements.
Skipped, Verdict	A test case or a test step was omitted during execution.
System Account	A system access reserved for automated operations such as importing test results. A login via the login page is not possible with this.

T

Test Activity, Report Diagram	The "Test Activity" report shows the test case results for a selected combination of one or more test systems and test environments during a selected in a selected time period as a graph on the dashboard.
Test Case	A test case comprises a set of input values, execution conditions, expected results, and postconditions defined for a specific system or test object under test. It is developed with respect to a specific goal or test condition, such as conformance to specific requirements.
Test Case History, Report Diagram	The “Test History - Overview” report displays the rate of defined test cases versus executed test cases of a project for one or more test environments and one or more test systems in a given time period as a graph on the dashboard.
Test Case Priority	The priority of the test case: “Low”, “Medium” or “High”.
Test Case Progress, Report Diagram	The Test Progress - Overview report shows the rate of the executed versus the defined test cases of a project project for a given set of test environments and test systems as a graph on the dashboard.
Test Case Status	The status of the test case. Possible values are “Locked”, “Approved”, “Draft” and “Omitted”.
Test Step	A test step in Klaros-Testmanagement is the smallest action that is processed within a test case. It contains execution preconditions, ex-

	pected results, and execution postconditions for each individual action during test execution.
Test Case Result	The result of the execution of a test case including a verdict such as “Skipped”, “ Passed”, “Failed” or “Failed”.
Test Environment	A test environment represents the external circumstances that can influence the test result. Examples for test environments are e.g. the operating system or an application server (e.g. Tomcat 10 with Ubuntu 20.04.2) or also parameters such as temperature or speed.
Test Environment Overview, Report Diagram	The “Test Environment - Overview” report displays the success and progress rate of test systems under a single test environment in a radar chart on the dashboard. If less than three test systems are configured, a bar chart is displayed instead.
Tester	A user role that is allowed to view objects and execute test cases and test suites.
Test Execution	The process of running a test on the component or system under test, producing actual result(s). (ISTQB) Klaros-Testmanagement automatically guides and logs test execution, see Testrunner.
Test Manager	A test manager has full read and write access to all data of a project. He can create, modify and delete all types of objects and manage project settings.
Test Run	A test run contains the results of the execution of a single test case or an entire test suite. It always refers to exactly one test system and exactly one test environment.
Testrunner	Klaros-Testmanagement contains a web-based client for the execution of manual tests. This guides the tester through the test steps, gives him the possibility to create annotations and attachments and automatically logs the test progress and results.
Test Step	A test step in Klaros Test Management is the smallest action that is processed within a test case. It contains execution preconditions, expected results and execution postconditions for each individual action during test execution.
testSegment	A test segment in Klaros-Testmanagement is defined sequence of individual test steps that can be inserted into a test case. A test segment can be used in several test cases at once. This makes it possible to realize a module concept based on test steps.
Test Suite	A test suite is a set of test cases that can be executed together in a group.
Test system, System under test, Unit under test, Test object	In Klaros-Testmanagement, a test system represents the component or system that is to be tested. Depending on the business sector, the

	terms system under test, SUT, DUT or test object are also commonly used.
System under Test Overview, Report Diagram	The Test System Overview report shows the success and progress rate of test progress rate of test environments under a single test system in a radar chart on the dashboard. If less than three test environments are configured, a bar chart is displayed instead.
Test Type	A group of test activities based on specific test objectives with the purpose of testing a component or system for specific properties (Functional, Non-Functional, Structural, Regression, Retest).
Trac	Trac is an open source application for defect management and issue tracking in software development. Klaros Test Management integrates with Trac.

V

Variety	The expected outcome of the test: positive or negative.
Verdict	A rating is the result of the executed test case, test step or test suite. Possible ratings are “Passed”, “Failed”, “Error”, “Inconclusive”, “Skipped”.

W

Web Browser	Klaros-Testmanagement is a web application and runs within a web browser. Supported browsers are Apple Safari, Google Chrome, Microsoft Edge and Mozilla Firefox.
Work Log	The work log shows the chronological progression and duration of the activities for processing a job. The time spent for the entire test execution results from the sum of the execution times. the sum of the execution times.

Y

YouTrack	YouTrack is a popular application by the company JetBrains for defect management, troubleshooting, and project management.