Klaros-Testmanagement Tutorial

Version 5.5.5

Veröffentlicht 8. Dezember 2023

verit
informationssysteme

Klaros-Testmanagement Tutorial

von Sabrina Gidley, Fabian Klaffke, Claudia Könnecke, Klaus Mandola, Patrick Reilly und Torsten Stolpmann

Version 5.5.5

Veröffentlicht 8. Dezember 2023

Zusammenfassung

Dieses Dokument ist eine Anleitung zum schnellen Einstieg in Klaros-Testmanagement. Es enthält eine Beispieltour durch die Webanwendung und ihrer Funktionalität.

Rechtlicher Hinweis .

Markenzeichen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1. Einführung

ist eine benutzerfreundliche Web-basierte Anwendung um Testprojekte zu organisieren. Sie verwaltet Testfälle, Testsuiten, Informationen über die zu testenden Systeme und die Umgebungen, in denen die Tests ausgeführt werden. Wurde ein Testfall oder eine Testsuite ausgeführt, wird das Ergebnis inklusive der zugehörigen Informationen über das zu testende System und die Umgebung in einer Datenbank gespeichert. Dies ermöglicht die volle Nachvollziehbarkeit eines jeden Testlaufs. Zahlreiche Berichte geben zu jeder Zeit Auskunft über den Fortschritt.

1.1. Navigation

Mit diesem Tutorial möchten wir Ihnen den Einsteig beim Arbeiten mit erleichtern. Figure 1.1 zeigt eine typische -Maske. Die Navigationsgebiete sind nummeriert und mit Bezeichnungen versehen, die Ihnen später das schnelle Auffinden erleichtern.

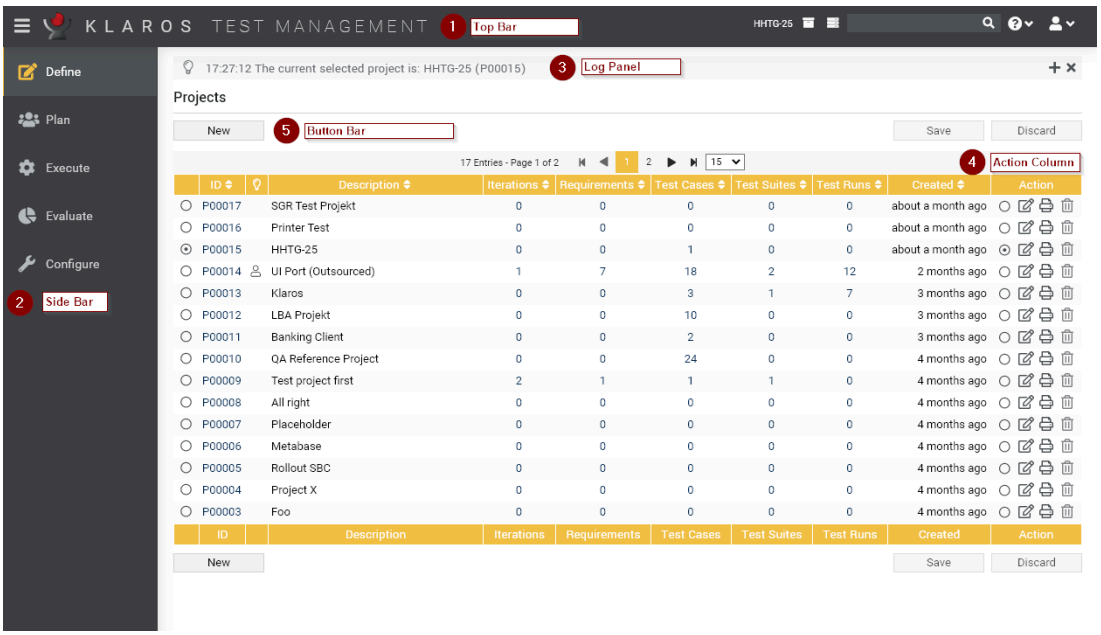


Abbildung 1.1. Navigation in

Die Benutzeroberfläche von

1. Die obere Menüleiste. Hier sind Funktionen zur Auswahl des aktuellen Projektes, zur Stichwort-Suche sowie Hilfe, Sprachwahl sowie zum Ausloggen erreichbar.

2. Die seitliche Menüleiste zeigt die fünf Hauptabschnitte. Wenn Sie mit der Maus über einen bestimmten Abschnitt in der Seitenleiste verweilen, werden die Menüeinträge für diesen Abschnitt angezeigt.

3. Das Log-Panel zeigt verschiedene nützliche Informationen, Warnungen und Fehlermeldungen an.

4. Die meisten Tabellen in haben eine Aktionsspalte wie sie hier zu sehen ist. Ein Klick auf die jeweiligen Icons in dieser Spalte führt Aktionen für die Objekte in der entsprechenden Zeile aus.

5. Viele Seiten in haben Schaltflächen zum Ausführen von Aktionen wie z.B. für das Erstellen neuer Objekte und zum Speichern oder Verwerfen von Änderungen. Diese Schaltflächen finden Sie stets oberhalb oder unterhalb der Tabelle.

Kapitel 2. Kurzanleitung für den schnellen Einstieg

In unserer Kurzanleitung wird gezeigt, wie ein Projekt mit Testumgebungen, Testsystemen und Testfällen und einer Testsuite angelegt wird. Die Testfälle erhalten Testschritte, außerdem werden benutzerdefinierte Felder und Kategorien angelegt. Danach wird ein einzelner Testfall sowie eine Testsuite mit drei Testfällen ausgeführt und die Ergebnisse dargestellt.

Das zu testende System ist ein Drucker, für den sowohl die Hardware als auch die Software (Treiber) getestet werden sollten. Die Treibersoftware ist in verschiedenen Versionen und für verschiedene Betriebssysteme, hier Windows und Linux, erhältlich.

Zusätzlich existieren zwei getrennte Entwicklungsteams: eines für die Hardware, das andere für die Software. Das Hardwareteam nutzt als System zur Fehlerverwaltung, das Softwareteam nutzt .

2.1. Setup

Zuerst wird das Testsetup erstellt. Dazu gehören das Projekt, die Testumgebungen, die Testsysteme, die Testfälle und eine Testsuite.

2.1.1. Ein Projekt anlegen

Projekte sind die übergeordneten Objekte in . Alle anderen Objekte wie Testfälle, Iterationen, Anforderungen oder Testsuiten sind Bestandteile des Projekts. Daher ist das Anlegen des Projekts immer der erste Schritt bei der Arbeit mit :

1. Erzeugen Sie mit Neu ein neues Projekt.

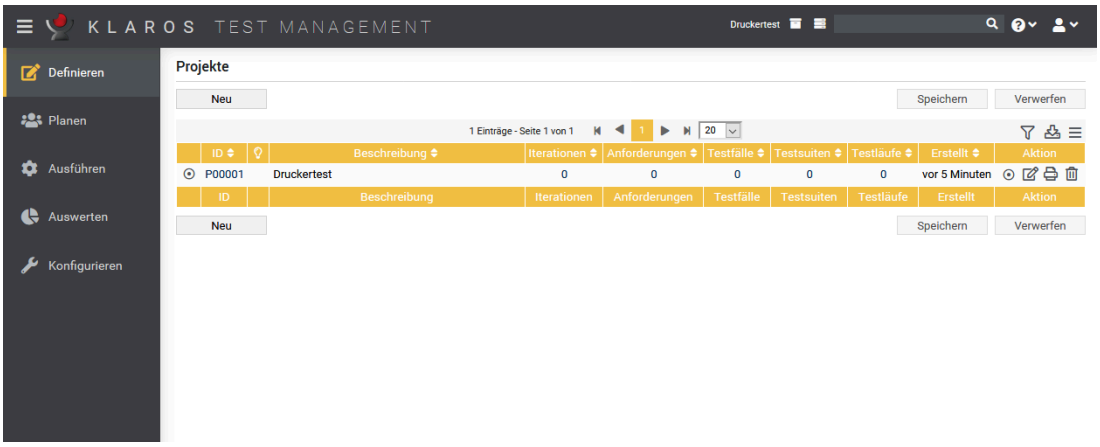


Abbildung 2.1. Die Seite Projekte

2. In das Feld *Beschreibung* geben Sie `Druckertest` ein.

3. Klicken Sie auf den Button Speichern. Es wird automatisch eine eindeutige Projekt-ID zugewiesen.

   Aktivieren Sie mit dem Radio-Button links neben der Projekt-ID oder rechts in der Aktionsspalte das neue Projekt. Nun sind auch die anderen Menüpunkte in der seitlichen Menüleiste anwählbar. Das neue Projekt kann jetzt mit dem Icon bearbeitet werden.

2.2. Testsysteme anlegen

Als Nächstes legen wir unsere Testsysteme - zu testenden Systeme (SUT - System Under Test) - an. In repräsentiert ein Testsystem eine Version eines Produktes oder einer Software, die getestet werden soll. In unserem Beispiel sind dies Drucker.

1. Wählen Sie unter *Definieren* den Eintrag *Testsysteme* in der seitlichen Menüleiste aus.

2. Klicken Sie auf den Button Neu .

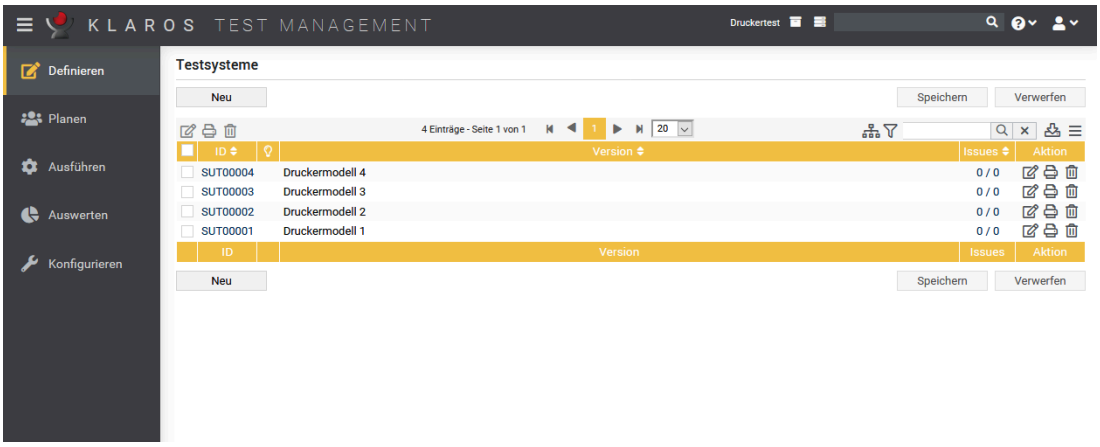


Abbildung 2.2. Die Seite Testsysteme

3. Fügen Sie den Text `Druckermodell 1` in das Feld *Version* ein.

4. Klicken Sie auf Speichern .

5. Wiederholen Sie die Schritte für drei weitere Testsysteme:

   - `Druckermodell 2`
   - `Druckermodell 3`
   - `Druckermodell 4`

   Mit einem Klick in die Spaltenüberschrift *ID* ändert sich die Sortierung der angezeigten Testsysteme.

2.3. Testumgebungen mit benutzerdefinierten Eigenschaften anlegen

Der nächste Schritt beim Einrichten des Projekts besteht im Festlegen der Testumgebungen, in denen die Tests stattfinden sollen. In legt die Testumgebung die äußeren Einflüsse fest, die das Testergebnis beeinflussen können. Dies könnte das Betriebssystem sein, auf dem der Druckertreiber installiert ist oder auch die physische Umgebung, in der der Drucker betrieben wird.

Für dieses Projekt legen wir zwei Software-bezogene und zwei Hardware-bezogene Testumgebungen an.

1. Wählen Sie unter *Definieren* den Eintrag *Testumgebungen* in der seitlichen Menüleiste aus.

2. Klicken sie auf Neu .

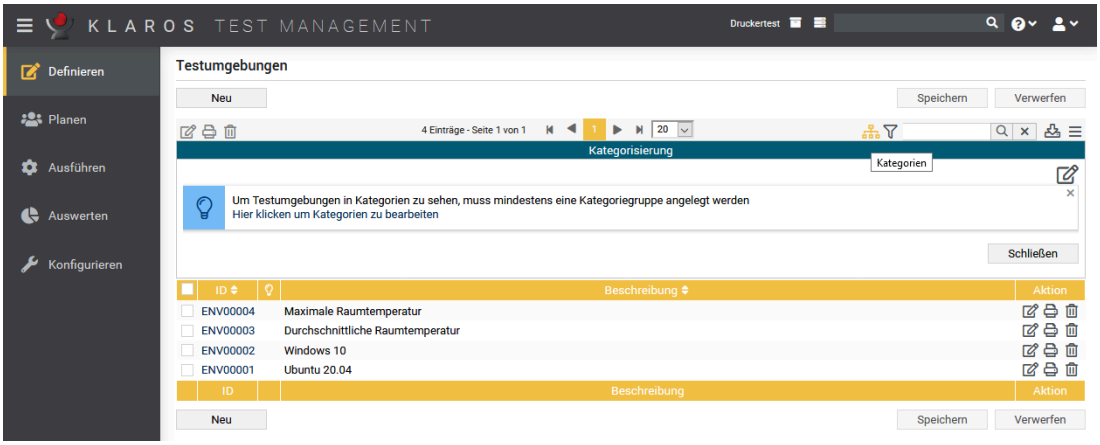


Abbildung 2.3. Die Seite Testumgebungen

3. Fügen Sie `Ubuntu 20.04` in das Feld *Beschreibung* ein und Speichern Sie.

4. Legen Sie drei weitere Testumgebungen mit den folgenden Werten im Feld *Beschreibung* an:

   - `Windows 10`
   - `Durchschnittliche Raumtemperatur`
   - `Maximale Arbeitstemperatur`

5. Speichern Sie alle Einträge.

2.4. Testfälle anlegen

Nun folgt die Erstellung die Testfälle. Wir werden sowohl für das Hardware- als auch für das Softwareteam Testfälle erstellen. Der erste Test für das Hardwareteam soll sicherstellen, dass der Drucker die Vorgaben bezüglich der Mindestanzahl der gedruckten Seiten pro Minute einhält.

1. Wählen Sie unter *Definieren* den Eintrag *Testfälle* in der seitlichen Menüleiste aus.

2. Klicken Sie auf *Neu*.

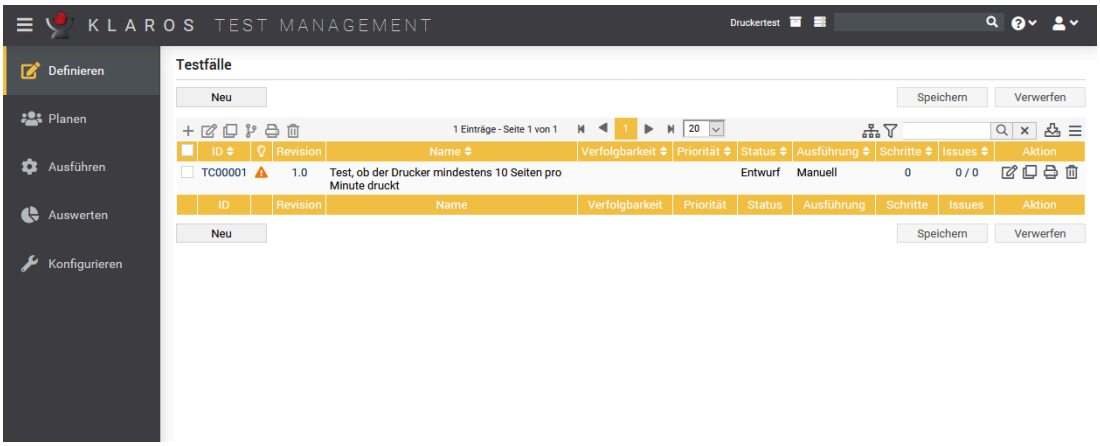


Abbildung 2.4. Die Seite Testfälle

3. Fügen Sie `Test ob der Drucker mindestens 10 Seiten pro Minute druckt` in das Feld *Name* ein. Der Name gibt Aufschluss über die Funktion/Aufgabe des Tests.

4. Klicken Sie auf Speichern.

5. Klicken Sie auf die Testfall-ID oder auf das Icon in der Aktionsspalte.

6. Wählen Sie die Ansicht *Schritte* aus.

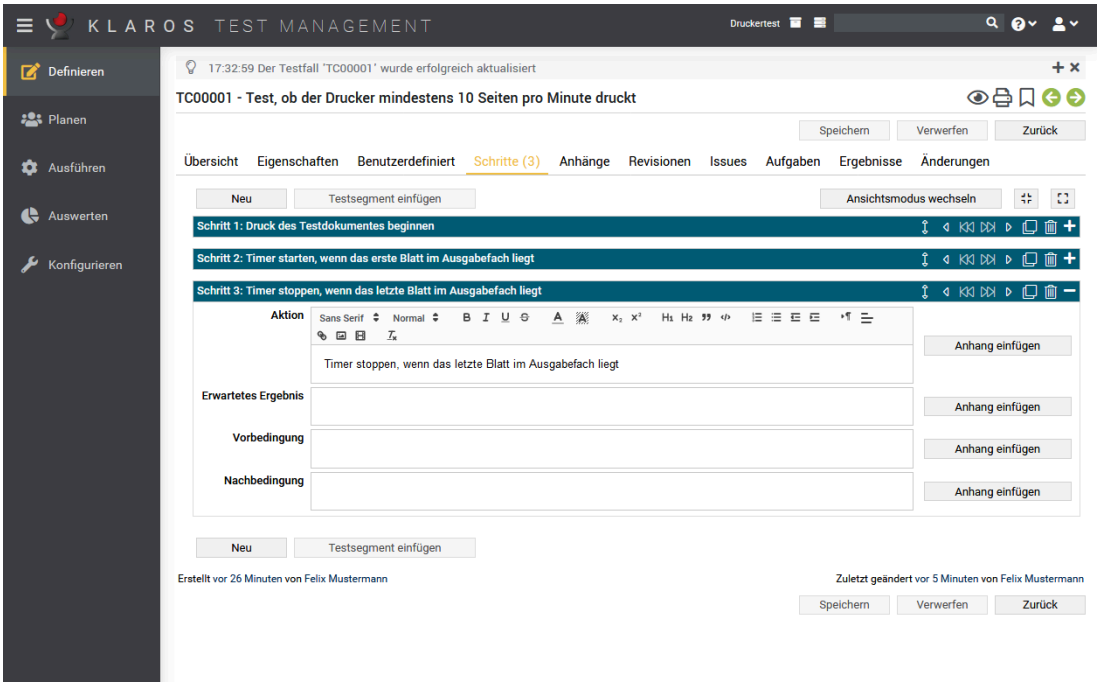


Abbildung 2.5. Der Tab Schritte in der Seite Testfälle

7. Erzeugen Sie die folgenden Schritte:

Beschreibung	Erwartetes Ergebnis
`Den Ausdruck des Testdokumentes starten`	
`Timer starten, wenn das erste Blatt im Ausgabefach liegt`	

`Timer stoppen, wenn das letzte Blatt im Ausgabefach liegt`	`The number of pages printed divided by the minutes required (rounded down) must equal 10 or more.`
	
	`Die Anzahl der gedruckten Seiten dividiert durch die benötigten Minuten (abgerundet) muss 10 oder mehr ergeben.`

8. Klicken Sie auf Speichern.

Als Nächstes legen wir die folgenden Testfälle mit jeweils einem Testfallschritt an. Klicken Sie zuerst auf Zurück um zur Liste der Testfälle zurückzukehren.

Name	Ausführung	Beschreibung der Testschritte	Erwartetes Ergebnis der Testschritte
`Drucke Testseite`	*Manuell*	`Drucken Sie die Seite mithilfe des Betriebssystems`	`Die Seite wird gedruckt`
`Leere Tintenkartusche erkennen`	*Manuell*	`Eine leere Tintenkartusche einlegen`	`Das Druckerdisplay zeigt die Warnung Patrone leer`

Sie sollten nun drei erstellte Testfälle sehen, einen mit zwei Testschritten und einen mit drei Testschritten.

2.5. Testsuiten anlegen

Der letzte Schritt bevor es mit dem Testen losgehen kann, ist das Anlegen von Testsuiten. In werden Testfälle einer Testsuite zugeordnet, um sie später gemeinsam ausführen zu können. Eine Testsuite kann einen oder mehr Testfälle enthalten und Testfälle können in mehreren Testsuiten vorkommen.

1. Wählen Sie unter *Definieren* den Eintrag *Testsuiten* in der seitlichen Menüleiste aus.
2. Klicken Sie auf `Neu`.

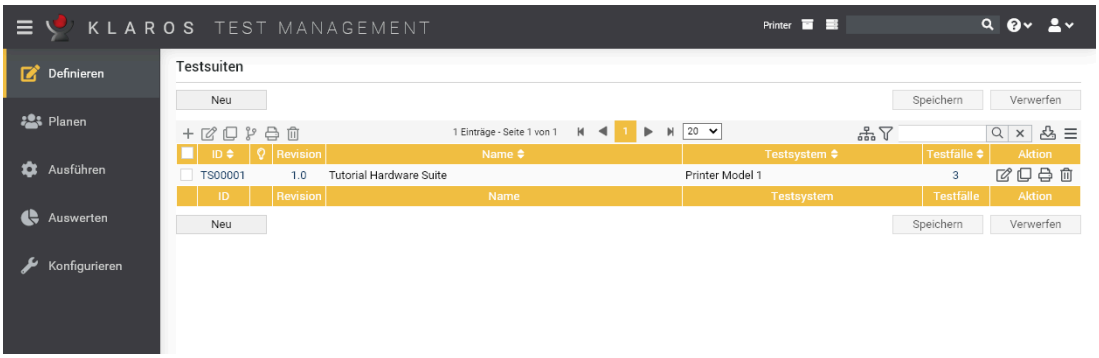


Abbildung 2.6. Die Seite Testsuiten

3. Fügen Sie `Tutorial Hardware Suite` in das Feld *Name* ein.
4. Klicken Sie auf `Speichern`.
5. Klicken Sie auf das Icon .
6. Klicken Sie auf die Ansicht *Eigenschaften*.
7. Klicken Sie auf das Icon Hinzufügen bei `Leere Tintenkartusche erkennen` und `Test ob der Drucker mindestens 10 Seiten pro Minute druckt` bei den Testfällen.

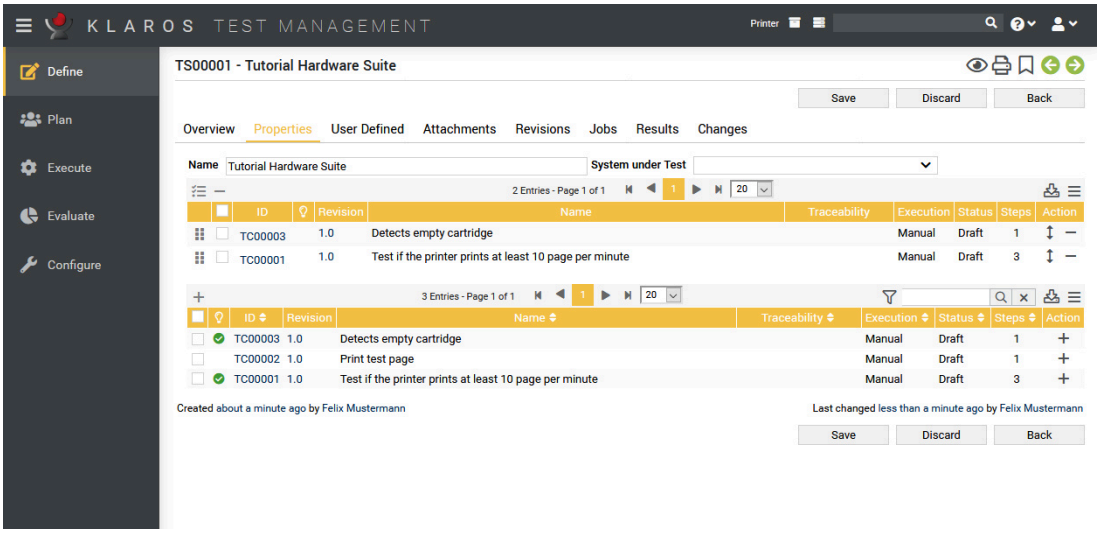


Abbildung 2.7. Die Seite Testsuite

8. Klicken Sie auf Speichern.

Unser Projekt ist nun fertiggestellt, und wir können mit der Ausführung der Testfälle beginnen.

2.6. Testfälle ausführen

Wir haben nun einige Testfälle und eine Testsuite erstellt und sind wir bereit für die erste Testausführung.

2.6.1. Einen einzelnen Testfall ausführen

Nun werden wir einen einzelnen Testfall mehrfach mit verschiedenen Parametern ausführen.

1. Wählen Sie unter *Ausführen* den Eintrag *Testfälle ausführen* in der seitlichen Menüleiste aus.

2. Nun ist die Seite *Testfälle ausführen* zu sehen. Sie zeigt für die vorhandenen Testfälle in der Spalte *Aktion* ein Ausführungsicon an. Das Aussehen des Icons kann variieren, für manuell ausführbare Tests wird hier das angezeigt.

Wichtig

Ein Testfall ist genau dann manuell ausführbar, wenn seine *Ausführung* auf *Manuell* oder leer gesetzt wurde, der Testfall mindestens einen Testschritt besitzt und sein *Status* entweder auf *Draft* oder *Genehmigt* steht. Zudem muss eine Testumgebung und ein Testsystem vorhanden sein. Ist dies nicht gegeben, erscheint das Icon ausgegraut. Die Gründe werden in der Spalte "Info" im Detail aufgeführt.

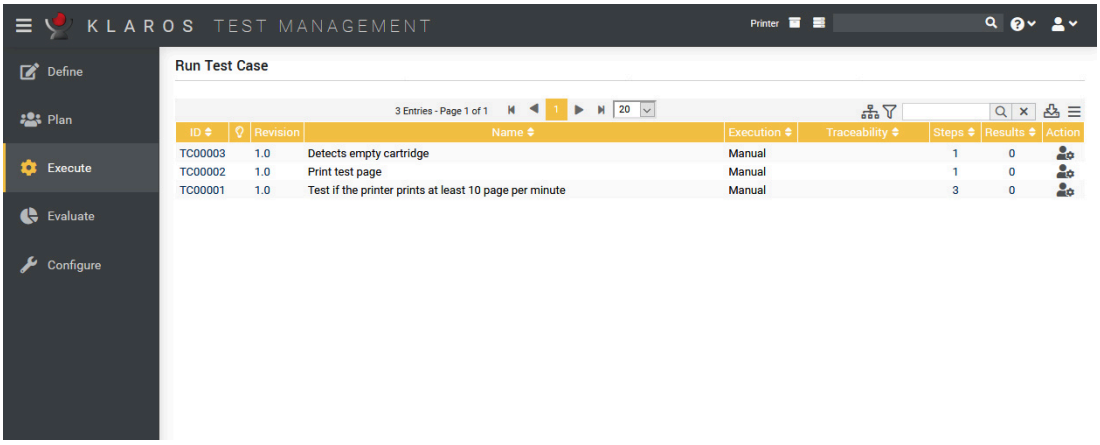


Abbildung 2.8. Die Seite Testfälle ausführen

Klicken Sie auf das Icon beim Testfall `Test ob der Drucker mindestens 10 Seiten pro Minute druckt` .

3. Auf dem nächsten Bildschirm wählen Sie `Durchschnittliche Raumtemperatur` als *Testumgebung* aus.

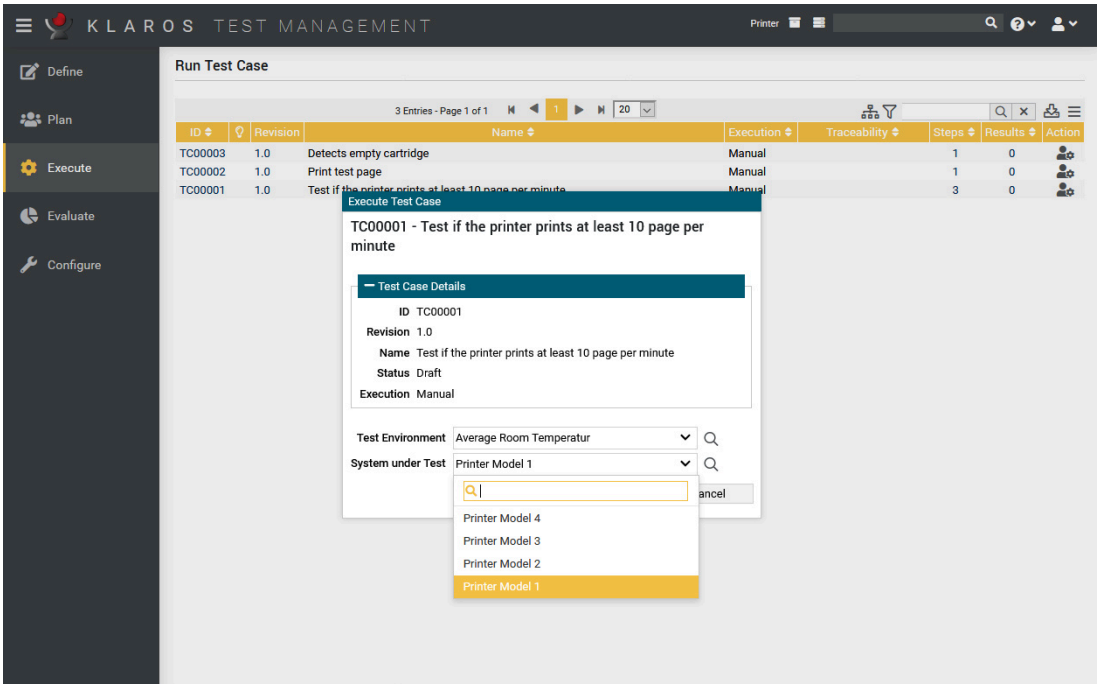


Abbildung 2.9. Die Seite Testfall ausführen

4. Wählen Sie `Drucker Modell 1` als *Testsystem* aus.

5. Klicken Sie auf `Ausführen` .

6. Nach einem Hinweis auf die laufende Ausführung öffnet sich ein neues Browser-Fenster. Falls dies nicht geschieht, überprüfen Sie, ob Ihr Browser einen Pop-up-Blocker verwendet und fügen Sie ggf. die Web-URL zu den Ausnahmen hinzu.

Im neuen Fenster wird der Tester schrittweise durch den Testablauf geführt. Auf der ersten Seite sehen Sie den Testfall im Überblick. Das bisherige Browser-Fenster wird für die Testdurchführung nicht benötigt und kann minimiert werden.

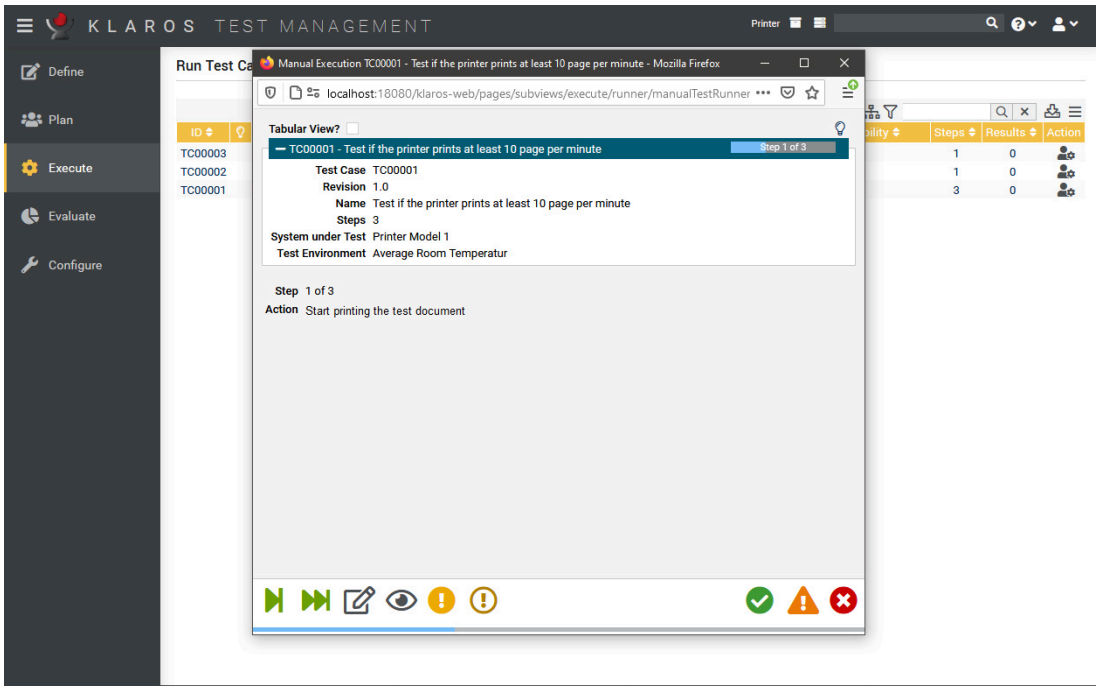


Abbildung 2.10. Der Testfall-Runner

7. Der manuelle Test-Runner zeigt nun den ersten Testschritt. Wir nehmen an, dass der Tester die Testseite ordnungsgemäß gedruckt hat, also ist der erste Schritt bestanden.

   Klicken Sie auf das grüne Bestätigungsicon .

8. Nun wird der zweite Schritt angezeigt. Markieren Sie auch die weiteren Schritte als Bestanden. Es erscheint ein weiterer Dialog, in dem Sie bestätigen, dass Sie die Testausführung beenden möchten. Klicken Sie auf den Button OK.

9. Es erscheint jetzt die Übersicht über den abgeschlossenen *Testlauf*. Da alle Schritte ohne Komplikationen bestanden sind, ist es nicht nötig einen Issue/Defekt zu erstellen.

   Klicken Sie auf den Button Beenden. Kehren Sie jetzt wieder auf der Übersichtsseite mit den drei Testfällen zurück.

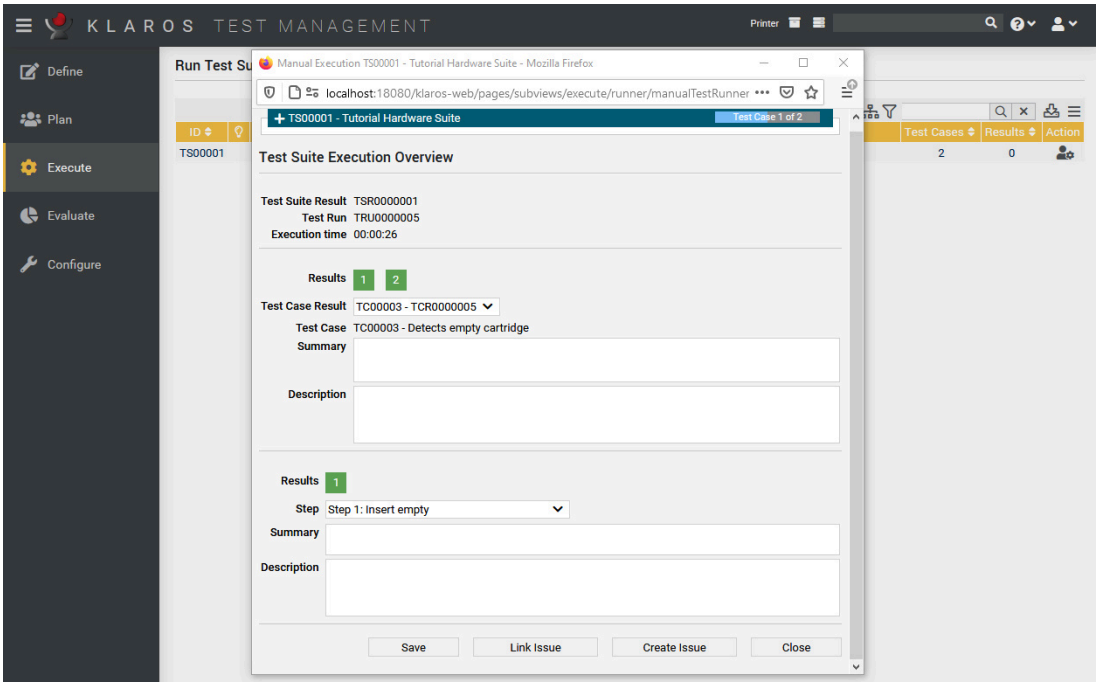


Abbildung 2.11. Der Ergebnisüberblick im Test-Runner

Jetzt werden wir denselben Testfall mit einem anderen Testsystem und in einer anderen Testumgebung ausführen.

1. Klicken Sie erneut auf das Icon bei den Testfällen `Leere Tintenkartusche entdecken` und `Test ob der Drucker mindestens 10 Seiten pro Minute druckt`.

2. Im folgenden Dialog wählen Sie *Maximale Arbeitstemperatur* als *Testumgebung* aus.

3. Wählen Sie *Drucker Modell 2* als *Testsystem* aus.

4. Klicken Sie auf Ausführen.

5. Klicken Sie auf das grüne Bestätigungsicon .

6. Klicken Sie erneut auf das grüne Bestätigungsicon .

7. Das Druckermodell 2 benutzt die Model 2 Druckerköpfe. Diese haben ein Überhitzungsproblem, deshalb wird dieser Testfall in einer Umgebung mit hoher Temperatur fehlschlagen.

   Klicken Sie auf das orange Icon.

8. Geben Sie im Textfeld *Kurzbeschreibung* für das Testfallergebnis den Text `Zu viele Pausen aufgrund von Überhitzung` ein.

   Diese Kommentare helfen, den Fehler zu reproduzieren.

9. Klicken Sie auf den Button Test beenden. Ein Dialog erscheint, in dem Sie das Beenden der Testfallausführung mit OK bestätigen.

10. Wenn Sie bereits ein Issue Management System angebunden haben, können Sie nun einen Issue direkt aus dieser Seite heraus erzeugen. Dieser gesamte Prozess wird in Abschnitt 3.6, „Issues“ ausführlich beschrieben.

11. Klicken Sie auf Beenden .

2.6.2. Testsuiten ausführen

Als Nächstes werden wir die Testsuite ausführen. Der Prozess ist dem Ausführen eines Testfalls sehr ähnlich. Zusätzlich erscheint zu Beginn noch ein Überblick-Bildschirm.

1. Wählen Sie in der seitlichen Menüleiste unter *Ausführen* den Eintrag *Testsuiten ausführen* aus.
2. Klicken Sie auf das Icon bei der Testsuite **`Tutorial Hardware Suite`** .

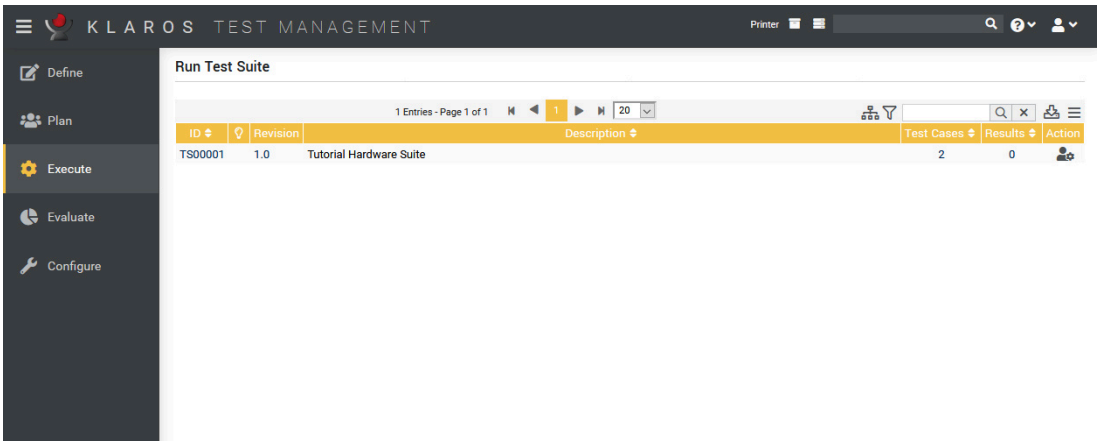


Abbildung 2.12. Die Seite Testsuite ausführen

3. Wählen Sie **`Durchschnittliche Raumtemperatur`** als *Testumgebung* aus.

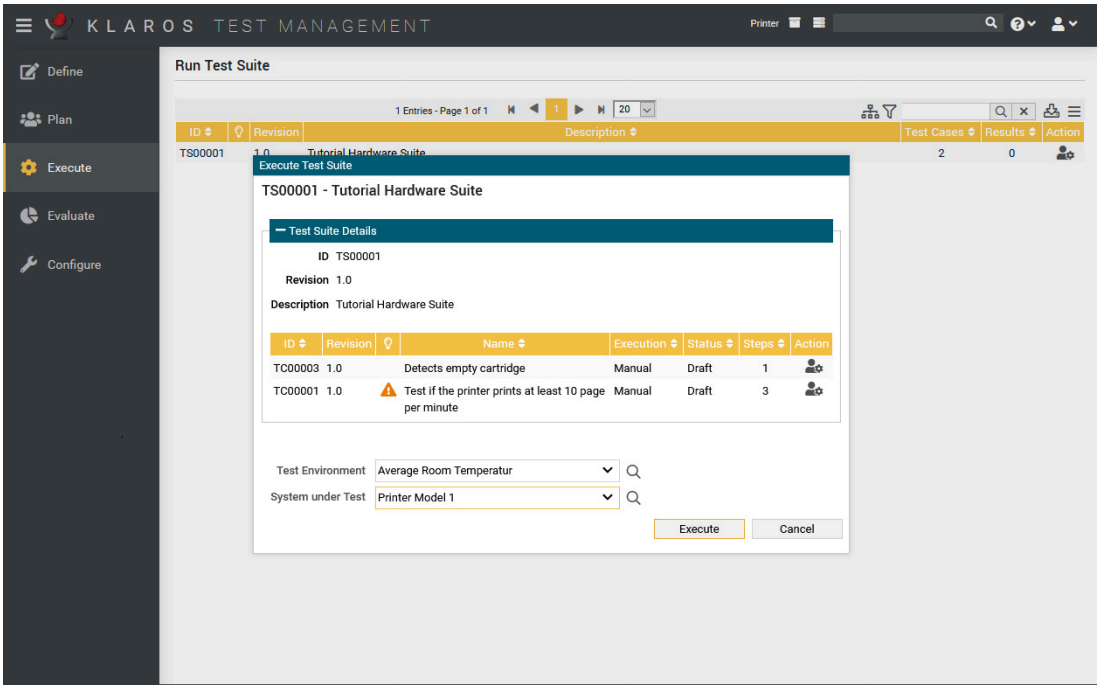


Abbildung 2.13. Die Seite Testsuite ausführen

4. Wählen Sie *Drucker Modell 1* als *Testsystem* aus.
5. Klicken Sie auf den Button Ausführen .

6. Klicken Sie auf den Button Beginnen im manuellen Test-Runner.

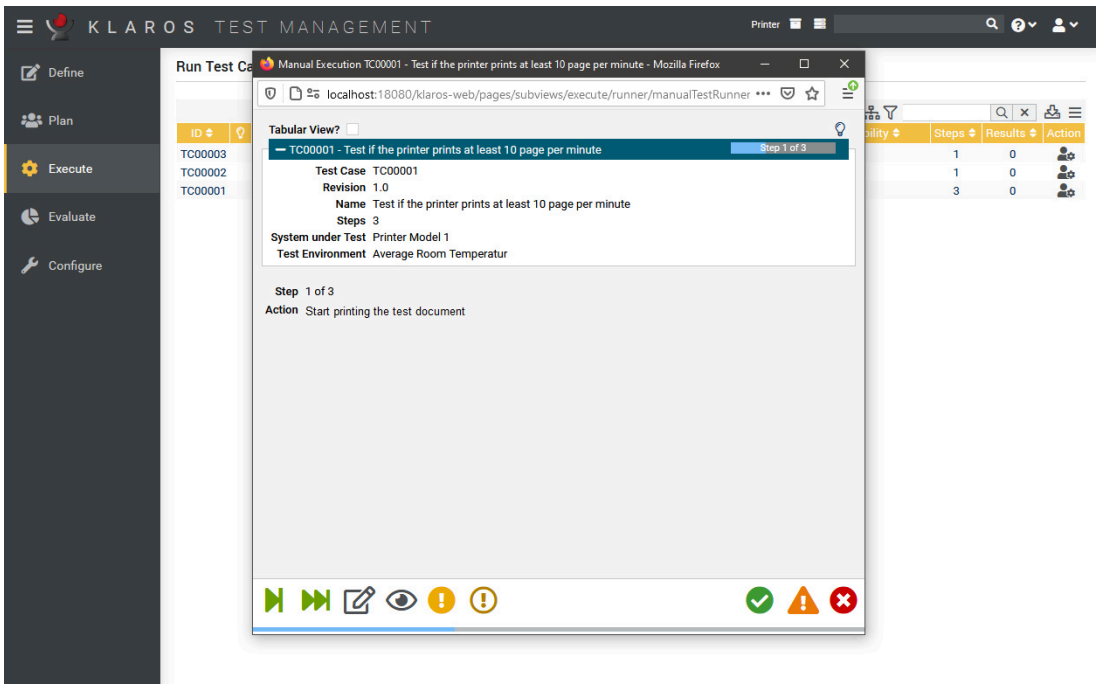


Abbildung 2.14. Der Testsuite Runner

7. Klicken Sie auf das grüne Bestätigungsicon für alle Schritte.

8. Bestätigen Sie den folgenden Dialog durch Klicken auf den Button OK.

9. Wiederholen Sie dies für den zweiten Testfall.

10. Klicken Sie auf den Button Schließen.

2.7. Ergebnisse und Berichte

Jetzt können wir uns die Testergebnisse ansehen. verfügt über ein umfangreiches und ausführliches Berichtswesen.

1. Loggen Sie sich mit dem Account `manager` ein.

2. Wählen Sie das Projekt *Drucker*, falls es nicht bereits ausgewählt ist.

3. Klicken Sie auf **Auswerten** in der seitlichen Menüleiste.

4. Nun wird das Dashboard für das Projekt *Drucker* angezeigt. Sie sehen zahlreiche relevante Informationen wie die Anzahl der Testfälle und Testsuiten im Projekt, die Anzahl der bestandenen und fehlgeschlagenen Testläufe sowie die Testaktivität der letzten 30 Tage.

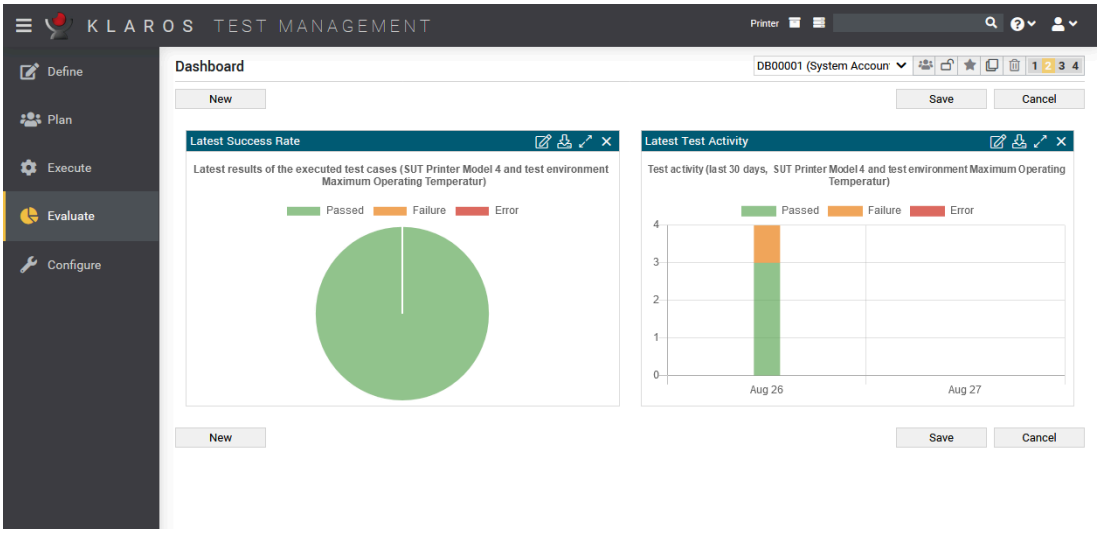


Abbildung 2.15. Das Dashboard

5. Den Bericht „Letzte Erfolgsrate“ können Sie über einen Klick auf das Icon anpassen. Im folgenden Dialog können Sie dann das Testsystem und die Testumgebung einstellen.

6. Im Bereich **Auswertung** können Sie sich individuelle Ergebnisse für Testfälle und Testsuiten anzeigen lassen.

   Wählen Sie unter *Auswerten* den Eintrag *Testfallergebnisse* in der seitlichen Menüleiste aus.

7. Hier sehen Sie die in diesem Projekt ausgeführten Testfälle, sowie die Anzahl der Testläufe und deren Ergebnisse.

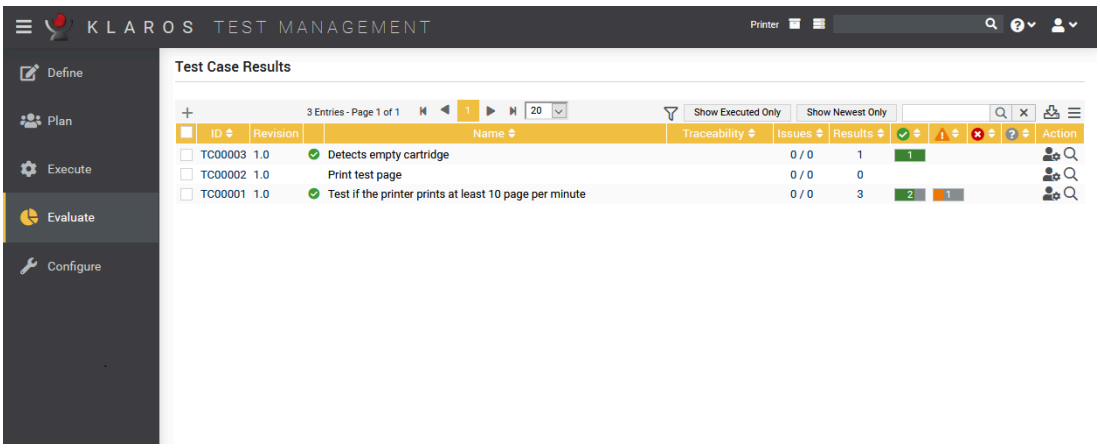


Abbildung 2.16. Die Seite Testfallergebnisse

8. Klicken Sie auf das Icon für den Testfall *Test ob der Drucker mindestens 10 Seiten pro Minute druckt*.

9. Nun sehen Sie eine Ansicht mit einer Zusammenfassung aller Testläufe für diesen Testfall.

   Klicken Sie auf das Icon um einen einzelnen Testlauf näher zu betrachten.

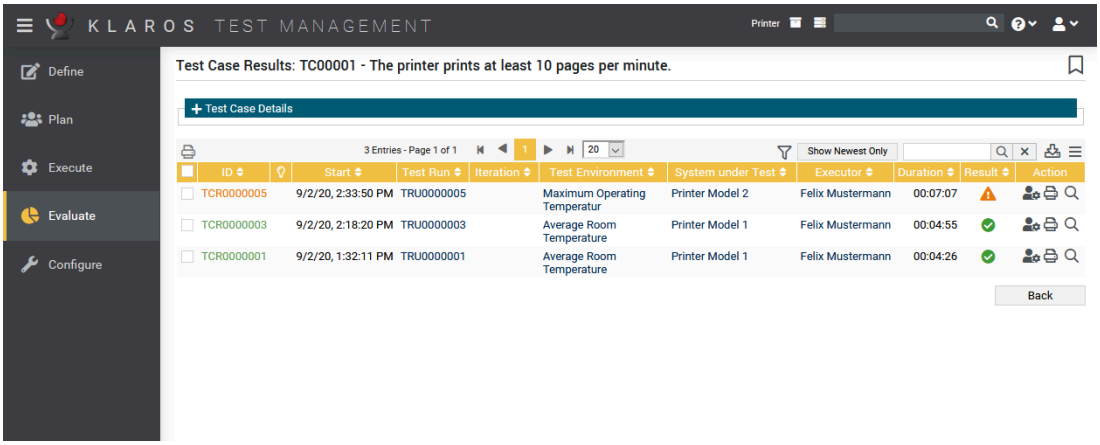


Abbildung 2.17. Die Seite Testfallergebnis

10. Dieser Bildschirm zeigt eine Aufschlüsselung der Ergebnisse der einzelnen Schritte im Testfall sowie die Zusammenfassung und Beschreibung, die der Tester für jeden dieser Schritte eingegeben hat.

    Wählen Sie unter *Auswerten* den Eintrag *Testsuite-Ergebnisse* in der seitlichen Menüleiste aus.

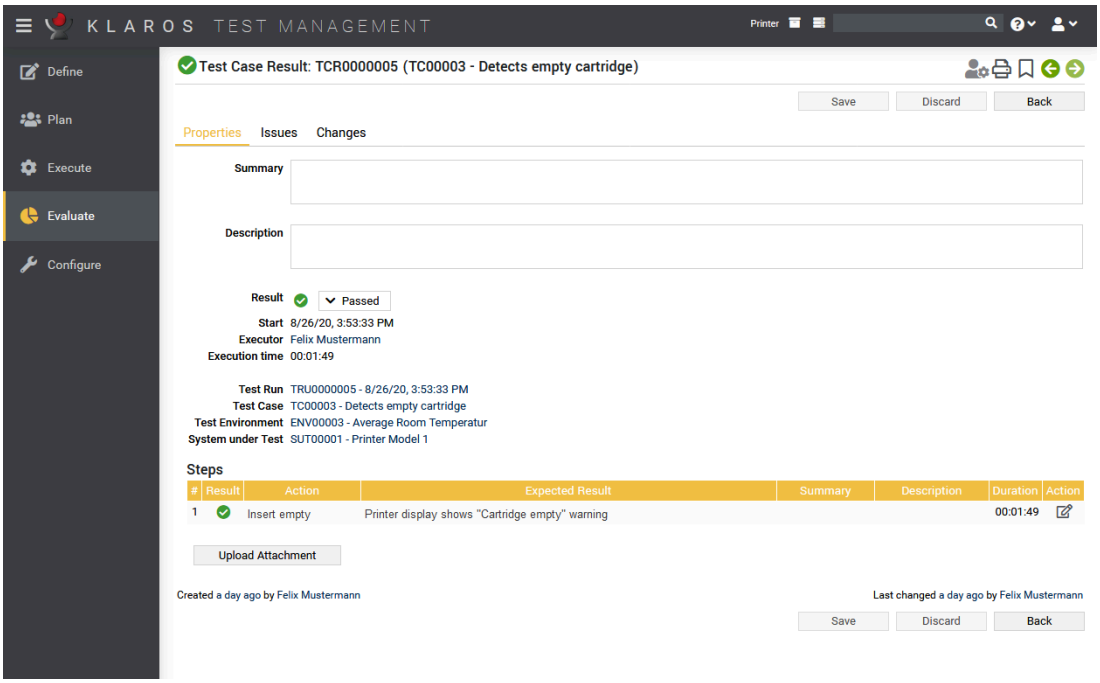


Abbildung 2.18. Die Seite Testfallergebnis

Diese Seite zeigt die Ergebnisse pro Suite in einer ähnlichen Weise wie Sie sie schon bei den *Testfallergebnisse* gesehen haben.

Dies beendet unsere Schnelleinführung. Die folgenden Kapitel enthalten weiterführende Abschnitte, die die Funktionen von detaillierter erläutern.

Kapitel 3. HowTos

Dieses Kapitel enthält einige nützliche Schritt-für-Schritt-Anleitungen für verschiedene Aktionen in Klaros-Testmanagement, die den ersten Einstieg erleichtern. Wir werden unser Drucker-Beispielprojekt aus der Schnellstartanleitung hier weiter verwenden.

Wir empfehlen, zunächst das Tutorial vollständig zu beenden, da wir hier auf die dort angelegten Beispieleingaben zurückgreifen.

3.1. Benutzer Rollen Management

In diesem Kapitel erfahren Sie anhand unseres Beispiels aus dem Tutorial, wie der Zugriff auf ein Projekt funktioniert und wie dieser für bestimmte Anwender eingeschränkt werden kann.

Die Standard-Installation von Klaros-Testmanagement enthält die 3 Anwenderrollen `Administrator`, `Manager` und `Tester` (siehe Klaros-Testmanagement Dokumentation hier und hier für weitere Informationen zu Benutzerrollen). Wenn Sie ein neues Projekt erstellen (siehe Abschnitt 2.1.1, „Ein Projekt anlegen“) haben zunächst alle Anwender von Klaros-Testmanagement Zugriff auf dieses Projekt. Um diesen Zugriff für einzelne oder mehrere Anwender einzuschränken, enthält Klaros-Testmanagement Enterprise Edition 3 verschiedene projektspezifische Rollen.

3.1.1. Einschränken des Projektzugriffs

enterprise edition Dieses Feature ist nur in der Klaros-Testmanagement Enterprise Edition verfügbar

Nun werden wir den Zugriff auf das Projekt *Printer* beschränken. Ausführliche Informationen dazu finden Sie unter hier.

Zuweisen projektspezifischer Rollen

Projektspezifische Rollen müssen nicht mit der Anwenderrolle übereinstimmen. Beispielsweise kann ein Anwender der als *Testmanager* angemeldet ist, in einem Projekt als *Tester* und in einem anderen als *Testmanager* fungieren.

1. Melden Sie sich als *Manager* in Klaros-Testmanagement an.
2. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.
3. Klicken Sie auf das Projekt mit der Nummer *P00001*, bzw. das „Printer Test“ Projekt, falls bereits mehrere Projekte angelegt wurden.
4. Wählen Sie den Tab *Zugang* aus.

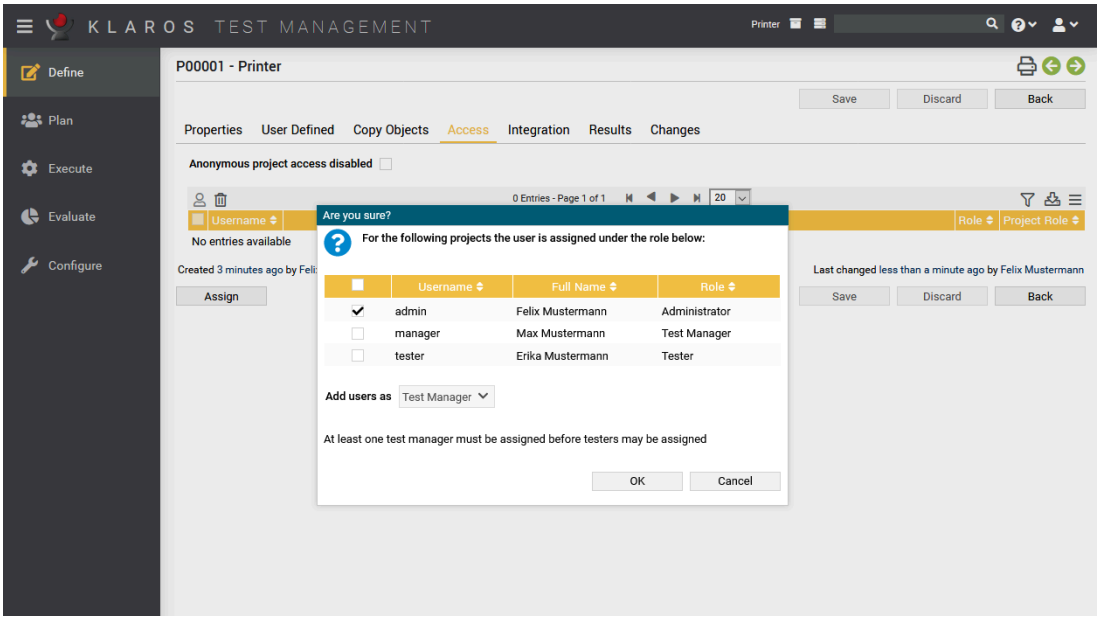
5. 

Abbildung 3.1. Die Ansicht „Zugang“

6. Klicken Sie auf den Zuweisen Button.

7. Wählen Sie in der Spalte *Max Mustermann* aus der Dropdown-Liste die *Projektrolle Testmanager* aus.

8. Bestätigen Sie die Änderungen mit einem Click auf den OK Button.

Der einzige Anwender, der Zugang zum Projekt *Printer* hat, ist jetzt der Anwender *Manager*. Administratoren können das Projekt trotzdem sehen, da sie von den Projektzugriffsregeln ausgenommen sind. Auf der Übersichtsseite aller Projekte zeigt das Symbol in der Spalte *Zusätzliche Hinweise* an, ob ein Projekt zugangsbeschränkt ist oder nicht.

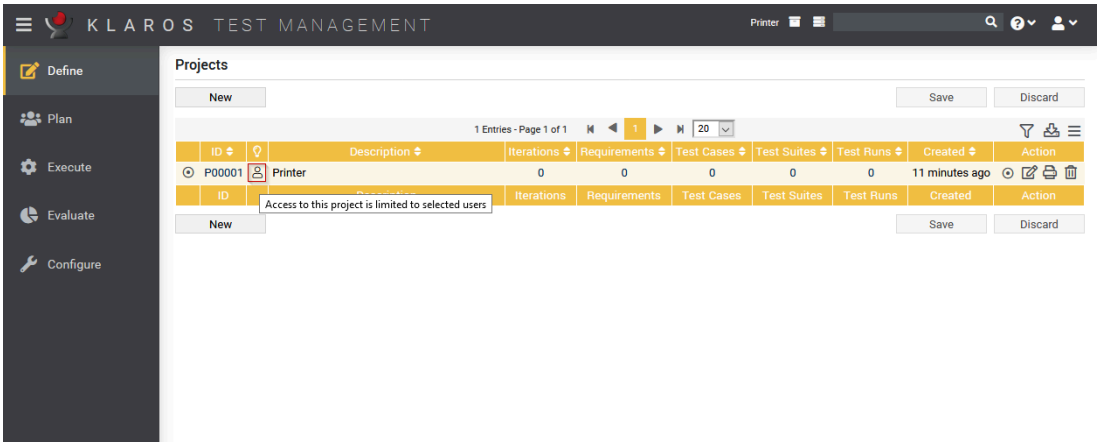


Abbildung 3.2. Ein Projekt mit eingeschränktem Zugang

1. Melden Sie sich als *Manager* in Klaros-Testmanagement an.

2. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.

3. Klicken Sie in der Projektübersicht auf *P00001*, bzw. auf das Drucker-Projekt, falls bereits mehrere Projekte angelegt wurden.

4. Wählen Sie das Tab *Zugang* aus.

5. In der Spalte *Erika Mustermann* wählen Sie aus der Dropdown-Liste die *Projektrolle Tester* aus.

   Klicken Sie auf den OK Button.

Wichtig

Nutzen Sie die projektspezifische Rollen, benötigen Sie mindestens einen Benutzer mit der Rolle "Testmanager" pro Projekt.

3.2. Kategorisierung

enterprise edition Dieses Feature ist nur in der Klaros-Testmanagement Enterprise Edition verfügbar

Wichtig

Um die Beispiele in diesem Kapitel ausführen zu können, sollte zuerst das Kapitel Abschnitt 2.3, „Testumgebungen mit benutzerdefinierten Eigenschaften anlegen“ abgeschlossen werden.

In unserem Beispielprojekt haben wir verschiedene Testumgebungen angelegt, sowie mehrere Testfälle für Hardware und Software. Eine nützliche Funktion zum Aufteilen von Bausteinen in der Klaros-Testmanagement Enterprise Edition ist das Kategorisieren.

In Klaros-Testmanagement können Objekte wie Testfälle, Testumgebungen u.a. selbstdefinierten Kategorien zugeordnet werden. Dies funktioniert ähnlich wie das Ablegen von Dateien in Ordnern eines Dateisystems. Wir werden nun zwei Kategorien erstellen und die angelegten Testumgebungen den jeweiligen Teams zuweisen. Weitere Informationen finden Sie in Abschnitt 3.2, „Kategorisierung“

1. Klicken Sie auf das Icon direkt über der Tabelle im Arbeitsbereich, um das Kategorien-Panel zu öffnen. Wählen Sie *Hier klicken, um Kategorien zu bearbeiten*.

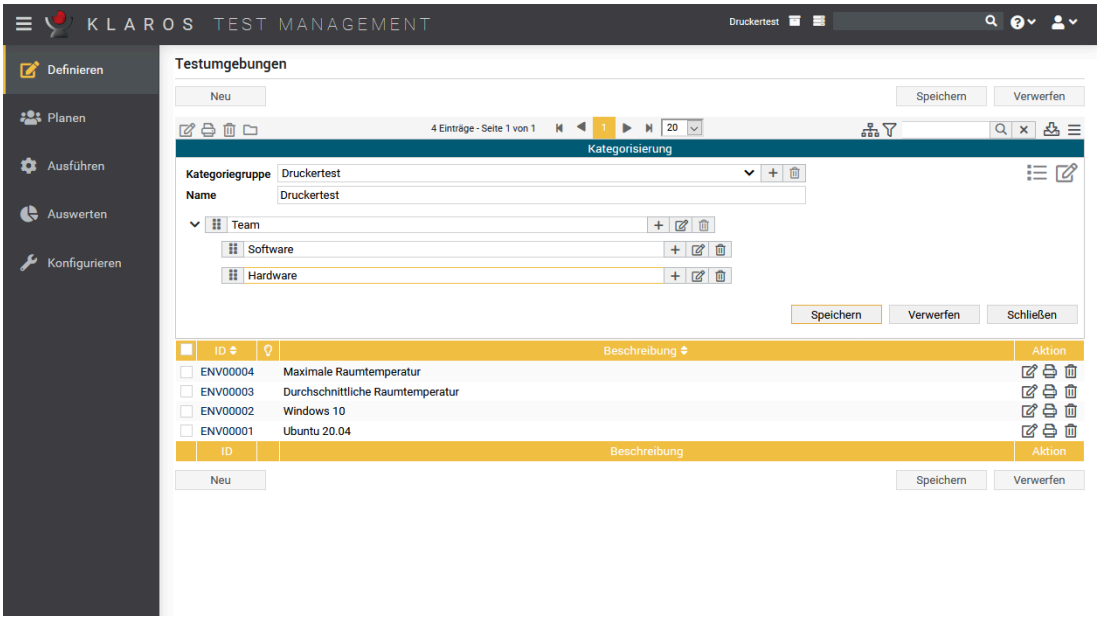


Abbildung 3.3. Kategorien anlegen

2. Klicken Sie auf das Icon um eine neue Kategoriegruppe zu erstellen.

3. Fügen Sie `Druckertest` in das Feld *Name* ein.

4. Nach dem Speichern wählen Sie die Kategorie *Druckertest* im Drop-Down-Menü *Kategoriegruppe* aus.

5. Als Nächstes benennen Sie das Feld *ROOT* in *Team* um.

6. Nun klicken Sie auf das Icon im Feld *Team* und geben `Hardware` in das Textfeld der neuen Unterkategorie ein. Wiederholen Sie den Schritt mit dem Eintrag `Software`.

7. Speichern Sie die Eingaben.

Nachdem die Kategorien erfolgreich angelegt wurden, können jetzt die Testumgebungen den einzelnen Teams zugeordnet werden.

1. Dazu wechseln Sie nun mit dem Icon rechts über der Tabelle die Ansicht.

2. Wählen Sie die Checkboxen links neben den softwarebezogenen Testumgebungen aus (`Windows 10` und `Ubuntu 20.04)` .

3. Klicken Sie auf das Icon links über der Tabelle.

4. Wählen Sie auf den Eintrag `Software` im angezeigten Dialog.

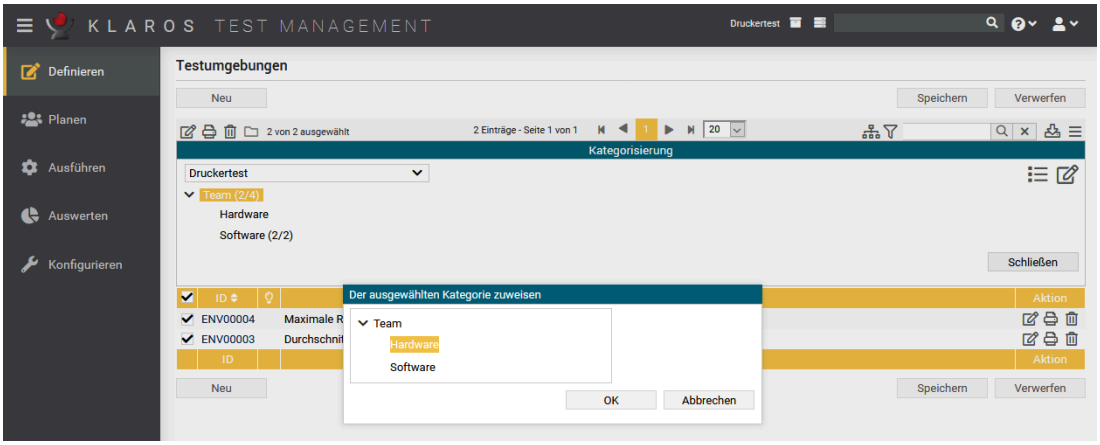


Abbildung 3.4. Testumgebungen an Kategorien zuweisen

5. Klicken Sie auf den Button `OK`. Die beiden Einträge erscheinen nun in der Kategorie *Software*.

6. Wählen Sie nun die Checkboxen neben den hardwarebezogenen Testumgebungen an.

7. Klicken Sie auf das Icon .

8. Klicken Sie nun im Dialog auf den Eintrag **`Hardware`**.

9. Klicken Sie auf `OK`.

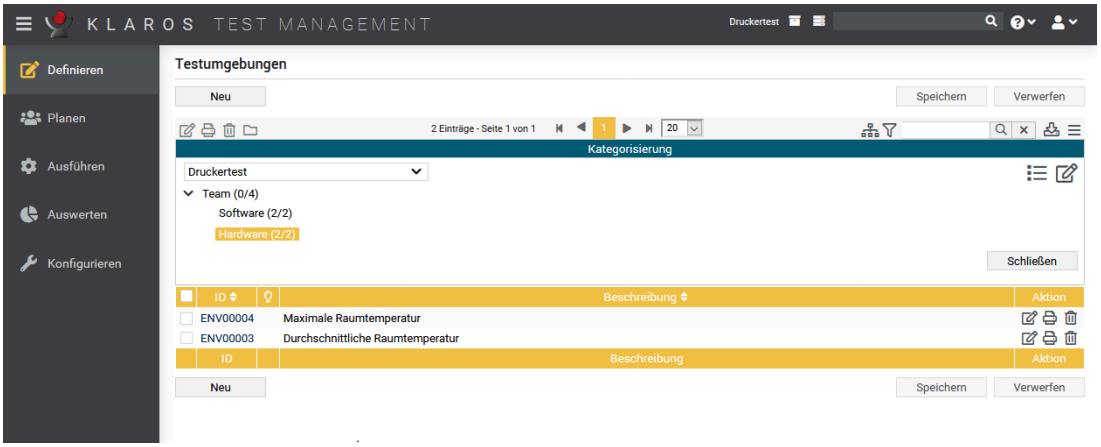


Abbildung 3.5. Kategorien der Testumgebungen

Ist das Kategorien-Panel geschlossen ist, werden alle vorhandenen Testumgebungen angezeigt. Bei geöffneten Panel werden nur die Testumgebungen der ausgewählten Kategorie angezeigt.

3.3. Benutzerdefinierte Eigenschaften

enterprise edition Dieses Feature ist nur in der Klaros-Testmanagement Enterprise Edition verfügbar

Wichtig

Um die Beispiele in diesem Kapitel ausführen zu können, sollte zuerst das Kapitel Abschnitt 2.1.1, „Ein Projekt anlegen“ abgeschlossen sein.

Mit benutzerdefinierten Feldern können die Bausteine in Klaros-Testmanagement um eigene Felder erweitert werden, beispielsweise um das Filtern und die Suche zu erleichtern oder um auf externe Systeme und Dokumente zu verlinken.

In diesem Kapitel werden wir benutzerdefinierte Felder für unsere Testsysteme erstellen. Da die Testsysteme verschiedene Druckermodelle sind, werden wir diese um Felder anreichern, die das Filtern und die Suche erleichtern. Eines dieser Eigenschaften ist die Firmware Version, dargestellt durch eine alphanumerische Zeichenfolge.

Die in unserem Beispiel verwendeten Drucker haben unterschiedliche Eigenschaften. Diese können in der Ansicht *Benutzerdefiniert* eingerichtet werden. Eine ausführliche Beschreibung hierzu finden Sie unter Abschnitt 3.3, „Benutzerdefinierte Eigenschaften“. Hier definieren wir die Firmwareversion und die Druckkopfmodelle.

1. Klicken Sie in die Projekt-ID oder in das Icon in der Aktionsspalte.
2. Wählen Sie das Ansicht *Benutzerdefiniert* und darunter *Testsystem* aus.
3. Klicken Sie auf Neu.

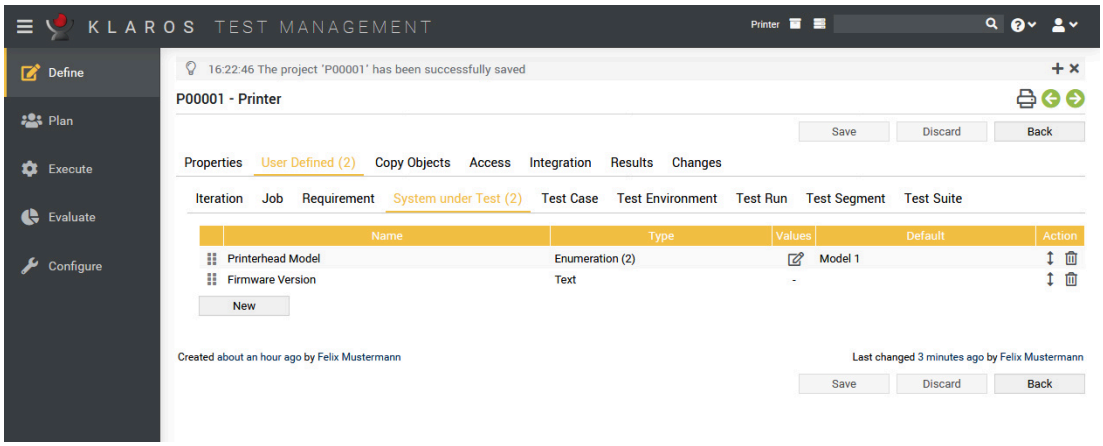


Abbildung 3.6. Hinzufügen von benutzerdefinierten Eigenschaften

4. Fügen Sie in das Feld *Name* den Eintrag `Firmware-Version` hinzu. Das Feld *Typ* ist bereits mit dem gewünschten Wert *Text* vorbelegt.
5. Klicken Sie auf *Speichern*.

Als Nächstes legen wir 2 Druckkopfmodelle an und definieren dafür einen Aufzählungstyp.

1. Klicken Sie auf Neu.
2. Fügen Sie in das Feld *Name* den Eintrag `Druckkopfmodell` hinzu.
3. Wählen Sie in der Spalte *Typ* den Eintrag *Aufzählung* aus. Klicken sie auf das Icon das in der Spalte "Werte" erscheint. Ein Dialog öffnet sich.

4. Klicken Sie auf den Button Neu und fügen Sie den Eintrag `Modell 1` hinzu.
5. Klicken Sie erneut auf den Button Neu, fügen Sie den Eintrag `Modell 2` hinzu.
6. Klicken Sie auf den Button Ok und anschließend im Hauptbildschirm auf Speichern.

1. Klicken Sie auf die ID für *Druckermodell 1* oder wählen Sie das Icon in dessen Aktionsspalte aus.
2. Wählen Sie die Ansicht *Benutzerdefiniert* aus.
3. Fügen Sie den Wert `V23.41.06` in das Feld *Firmware Version* ein.
4. Wählen Sie *Modell 1* aus der *Druckkopfmodell* Dropdown-Liste.
5. Klicken Sie auf Speichern.

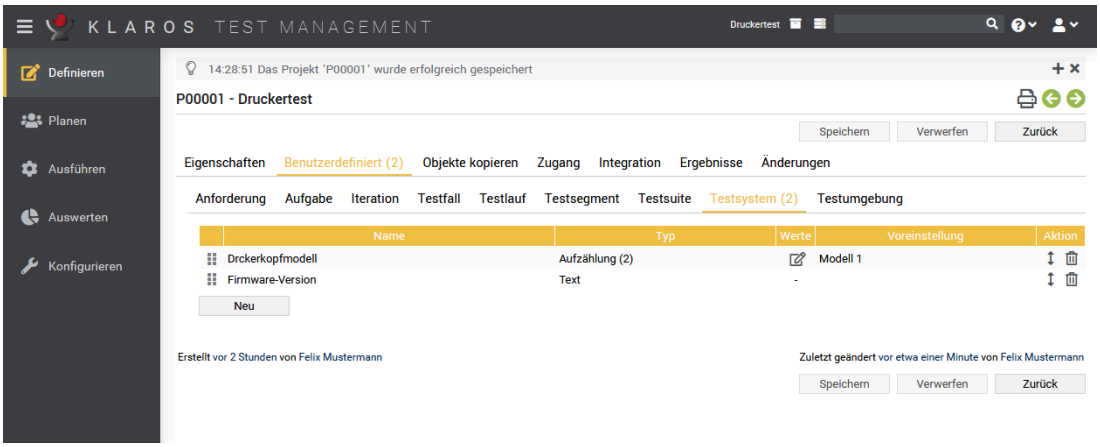


Abbildung 3.7. Die Ansicht Benutzerdefiniert / Testsysteme

6. Klicken Sie nun auf den grünen Pfeil rechts oben um auf das nächste Element umzuschalten. Nun wird das Druckermodell 2 angezeigt. Wiederholen Sie jeweils den Prozess für die verbliebenen drei Testsysteme.

Version	Firmware Version	Druckkopfmodel
`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. Pausieren und Fortsetzen von Testfällen und Testläufen

Wichtig

Um die Beispiele in diesem Kapitel ausführen zu können, sollte zuerst das Kapitel Abschnitt 2.5, „Testsuiten anlegen“ abgeschlossen sein.

Klaros-Testmanagement unterstützt das Pausieren und Fortsetzen von Testfällen und Testsuiten. Wenn mindestens ein Testschritt bei der Ausführung durchgeführt wurde, kann der Testlauf zu

einem späteren Zeitpunkt wiederaufgenommen werden. Dieser findet sich dann unter *Testläufe fortsetzen*.

1. Melden Sie sich als *Tester* in Klaros-Testmanagement an.
2. Wählen Sie das Projekt *Printer* aus.
3. Wählen Sie unter *Ausführen* den Eintrag *Testsuite Ausführen* in der seitlichen Menüleiste aus.
4. Wählen Sie die Testsuite *Tutorial Hardware* aus.
5. Schließen Sie den ersten Testfall der Testsuite wie in Abschnitt 2.6.2, „Testsuiten ausführen“ beschrieben ab.
6. Klicken Sie auf den Button Verwerfen .

   Der manuelle Testrunner schließt sich nun und Sie befinden sich wieder im Hauptfenster von Klaros-Testmanagement .
7. Wählen Sie den Eintrag **Testlauf fortsetzen** in der seitlichen Menüleiste aus.

   Die unterbrochene Ausführung der Testsuite wird hier angezeigt. Ebenso ist zu sehen, dass einer der beiden Testfälle abgeschlossen wurde.
8. Sie können auf das Icon klicken, um die Ausführung der Testsuite fortzusetzen oder auf das Icon um die Testsuite und die zugehörigen Ergebnisse zu löschen.

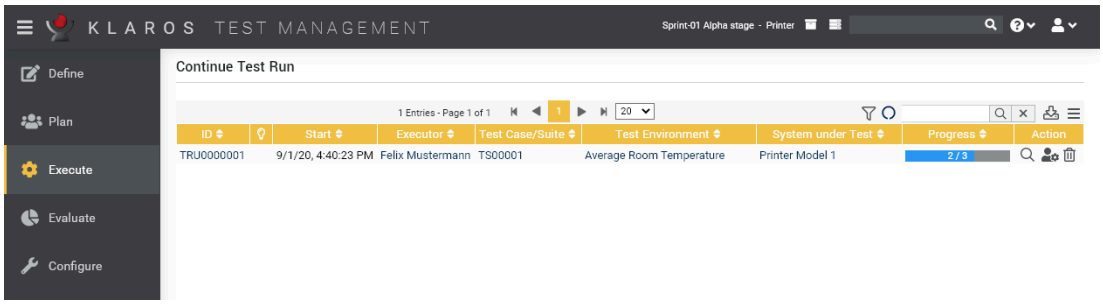


Abbildung 3.8. Fortsetzen eines Testlaufs

3.5. Konfigurieren der Issuemanagement Systeme

Zu Anfang loggen Sie sich bitte mit der Rolle "Administrator" ein. Wir beginnen mit dem Anlegen eines neuen Projektes und der Zuordnung des Issue-Managementsystems.

Anmerkung

Username	Password
admin	admin
manager	manager
tester	tester

Tabelle 3.1. Benutzerrollen

1. Loggen Sie sich ein und beginnen Sie, indem Sie durch Klicken auf den Button Neu ein neues Projekt anlegen. Wählen Sie dieses Projekt nun aus und klicken auf "Bearbeiten". Nun erscheint die Seite **Konfigurieren** - **Integrieren** - *Issue Management*.

2. Klicken Sie auf den Button Neu .

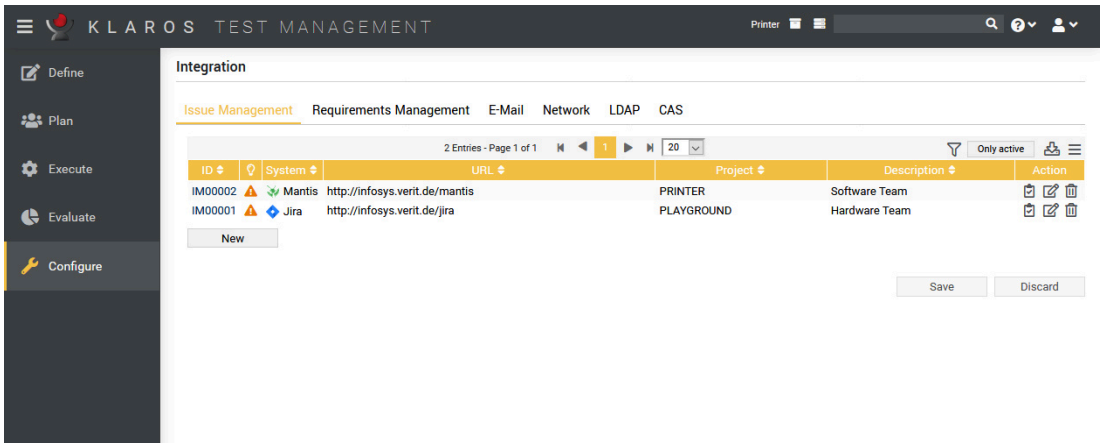


Abbildung 3.9. Die Seite Issue Management

3. Wählen Sie JIRA in der Drop-Down-Liste *System* aus.

4. Fügen Sie `PRINTER` in das *Projekt* Textfeld (Oder den Namen Ihres Testprojektes, z.B. `PLAYGROUND`)

5. Fügen Sie `Hardware Team` in das Textfeld *Beschreibung* ein.

6. (Optional) Wenn Sie JIRA verwenden fügen Sie die Adresse in das *URL* Textfeld und klicken Sie auf das Icon .

7. Klicken Sie auf den Button Speichern um Ihr neues Issue Management System zu sichern.

Wiederholen Sie den Prozess, wählen Sie Mantis statt JIRA aus und fügen Sie `Software Team` in die Beschreibung ein.

Nun können diese beiden Issue Management Systeme Projekten in Klaros-Testmanagement zugewiesen werden.

Damit sind die Arbeiten, welche die Administrator-Rolle in Klaros-Testmanagement erforderlich machte abgeschlossen. Sie können sich nun ausloggen und erneut in der Rolle `Manager` anmelden.

3.6. Issues

Wichtig

Um die Beispiele in diesem Kapitel ausführen zu können, sollte zuerst das Kapitel Abschnitt 2.6.1, „Einen einzelnen Testfall ausführen“ abgeschlossen werden.

Klaros-Testmanagement supports the creation of issues in issue management systems (IMS) (see Abschnitt 3.5, „Konfigurieren der Issuemanagement Systeme“) 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 *Issue verlinken* 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 *Evaluieren - Issue verlinken* section of Klaros-Testmanagement. Each *Ergebnisüberblick* page in the manual test runner has the Issue Verlinken and Issue Erzeugen buttons.

3.6.1. Issues verlinken

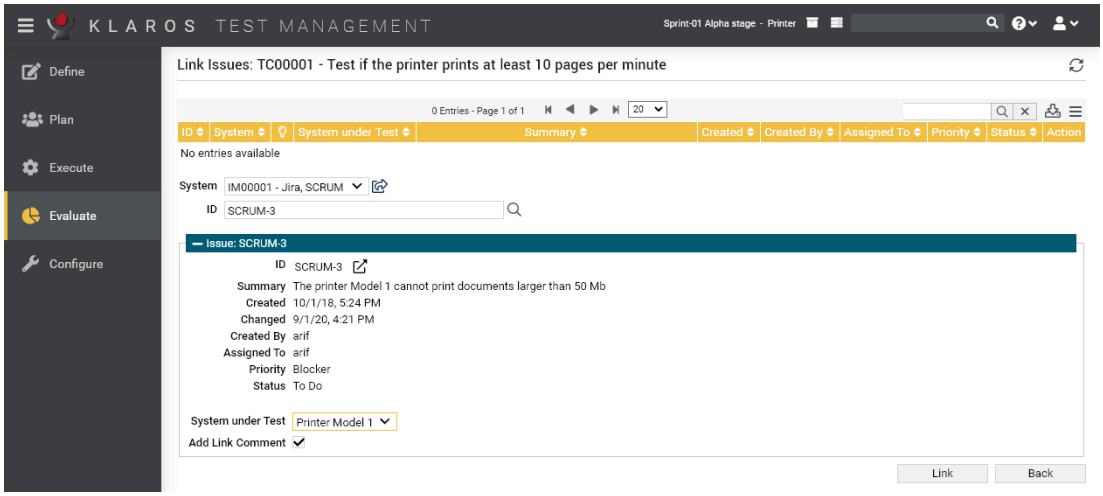


Abbildung 3.10. Issues verlinken

1. Melden Sie sich als *Tester* in Klaros-Testmanagement an.
2. Wählen Sie unter *Auswerten* den Eintrag *Issues* in der seitlichen Menüleiste aus.
3. Klicken Sie auf das Icon in der Aktionsspalte des Testfalls *Test if the printer prints at least 10 pages per minute*.
4. Klicken Sie auf den Button **Neu**.
5. Wählen Sie das Issue Management System für das *Hardware Team* aus der *Issue Management System* Dropdown-Liste.
6. Fügen Sie eine gültige ID für das Issue in das Feld *ID* ein. Diese ID muss identisch mit der existierenden ID im IMS sein.
7. Klicken Sie auf das Icon neben dem Feld *ID*. Dies kann einen Moment dauern, da das Issue erst vom IMS geholt wird.
8. Wählen Sie das Testsystem *Printer Model 1* aus der Dropdown-Liste *Testsysteme* aus.
9. Klicken Sie auf den Button Verlinken.

Die Liste der verknüpften Issues des Testfalls enthält nun einen einzigen Eintrag.

3.6.2. Issues anlegen

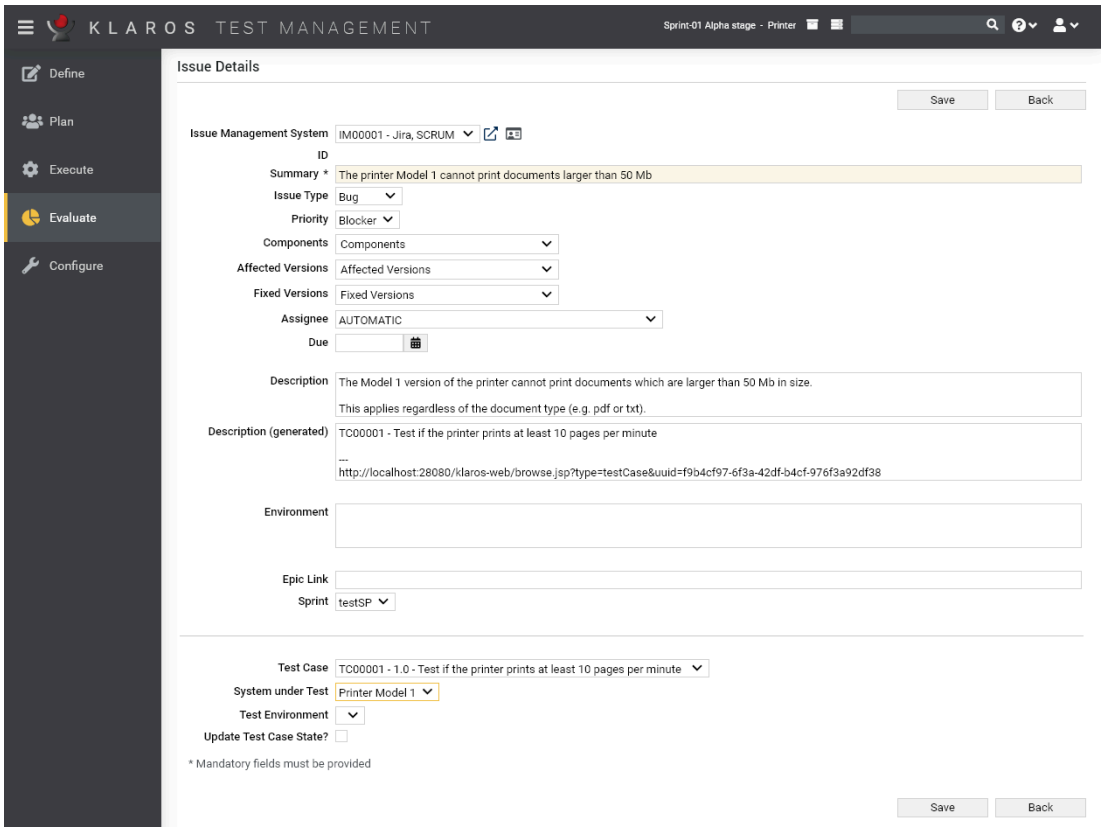


Abbildung 3.11. Issues anlegen

1. Melden Sie sich als *Tester* in Klaros-Testmanagement an.
2. Wählen Sie den Bereich **Evaluieren** in der seitlichen Menüleiste aus.
3. Wählen Sie den Eintrag **Issues** in der seitlichen Menüleiste aus.
4. Klicken Sie auf den Button **Neu**.
5. Wählen Sie das Issue Management System für das *Software Team* aus der *Issue Management System* Dropdown-Liste.
6. Fügen Sie `The printer Model 1 cannot print documents larger than 50 Mb` in das Feld *Zusammenfassung* ein.
7. Fügen Sie in das Feld *Beschreibung* `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).` ein.
8. Wählen Sie den Testfall *Test if the printer prints at least 10 pages per minute* aus der Drop-down-Liste *Testfälle* aus.
9. Klicken Sie auf den Button *Save*.

Jetzt sollten Sie die Nachricht *The issue was successfully created in Mantis with ID XYZ.* im Log-Panel oben auf dem Bildschirm sehen.

3.7. Revisionen

Wichtig

Um die Beispiele in diesem Kapitel ausführen zu können, sollte zuerst das Kapitel Abschnitt 2.4, „Testfälle anlegen“ abgeschlossen werden.

Testanforderungen ändern sich während eines Produktlebenszyklus, aufgrund einer Vielzahl interner und externer Änderungen. Nehmen wir zum Beispiel an, dass bei unserem *Printer* statt 10 Seiten pro Minute 20 Seiten der neue Status quo sind. Anstatt den Testfall *Test if the printer prints at least 10 page per minute* zu kopieren und anzupassen kann stattdessen einfach eine neue Revision erstellt werden.

Anmerkung

Mit Klaros-Testmanagement Können Sie Revisionen von Testfällen, Testsegmente, Testsuiten und Anforderungen verwalten.

1. Melden Sie sich als *Manager* in Klaros-Testmanagement an.
2. Wählen Sie das Projekt *Printer* aus.
3. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.
4. Wählen Sie den Eintrag **Testfälle** in der seitlichen Menüleiste aus.
5. Klicken Sie auf das Icon im Testfall *Test if the printer prints at least 10 page per minute*.
6. Klicken Sie auf das Tab *Revision*.
7. Klicken Sie auf den Button `Neue Revision`.
8. Fügen Sie `requirements have changed` in das Feld *Comments* ein.
9. Klicken Sie auf den Button `OK`.
10. Klicken Sie auf das Tab *Eigenschaften*.
11. Ändern Sie die *Beschreibung* des Testfalles zu `Test if the printer prints at least 20 page per minute`.

Sie haben nun erfolgreich eine neue Revision eines Testfalles angelegt.

Anmerkung

Wenn Sie eine neue Revision anlegen, wird diese automatisch zur aktiven Revision. Sie können über das Revisionen-Tab jederzeit zu einer anderen Version wechseln.

3.8. Anforderungen

Dieses Feature ist nur in der Klaros-Testmanagement Enterprise Edition verfügbar

Anforderungen sind Bedingungen, die erfüllt sein müssen, damit ein zu testendes Produkt als fertiggestellt gilt. Eine Anforderung in Klaros-Testmanagement kann mit mehreren Testfällen verknüpft sein, die alle erfolgreich ausgeführt sein müssen, damit die Anforderung als erfüllt gilt.

In diesem Kapitel werden wir eine Anforderung erstellen, mit einem Testfall verknüpfen und diesen ausführen.

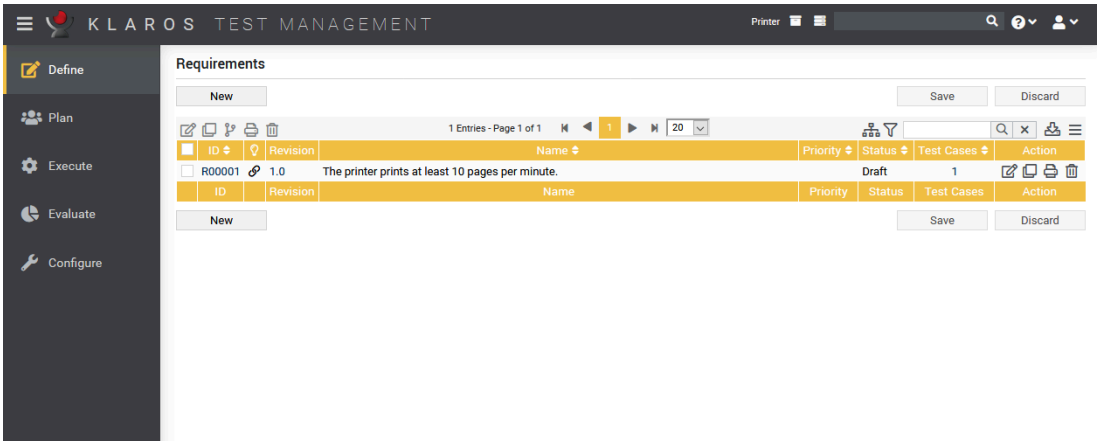


Abbildung 3.12. Die Anforderungen Seite

1. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.
2. Wählen Sie den Eintrag **Anforderungen** in der seitlichen Menüleiste.
3. Klicken Sie auf den Button Neu .
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. Klicken Sie auf den Button Speichern .
7. Klicken Sie auf das Icon icon of the requirement *The printer prints at least 10 pages per minute*.
8. Wählen Sie das Tab *Testfälle* tab.
9. Klicken Sie auf den Button New .
10. Select the test case *Test if the printer prints at least 10 page per minute*.
11. Klicken Sie auf den Button Zuweisen .

Multiple Requirements

Sie können derselben Anforderung mehrere Testfälle zuweisen.

Sie können ebenfalls Testfälle mehreren Anforderungen zuweisen!

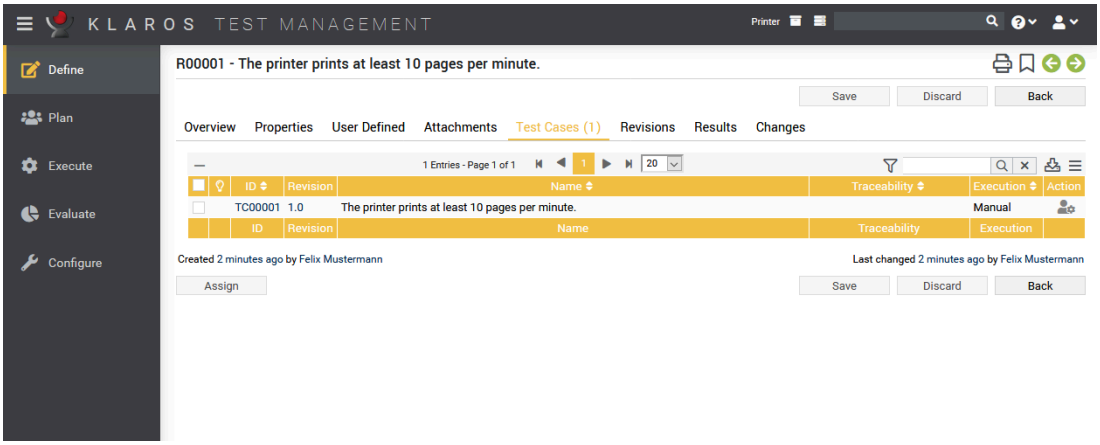


Abbildung 3.13. Die Anforderungen Detail Seite

Nun werden Sie den Testfall mit der verlinkten Anforderung ausführen.

1. Wechseln Sie zu dem Bereich **Ausführen** in der seitlichen Menüleiste aus.
2. Wählen Sie den Eintrag **Testfälle ausführen** in der seitlichen Menüleiste aus.
3. Klicken Sie auf das Icon.
4. Klicken Sie auf den Button Ausführen .
5. Klicken Sie auf das grüne Bestätigungsicon .
6. Klicken Sie auf den Button OK .

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. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.
2. Wählen Sie den Eintrag **Anforderungen** in der seitlichen Menüleiste.
3. Klicken Sie auf das Icon icon of the requirement *The printer prints at least 10 pages per minute*.
4. Wählen Sie das Tab *Ergebnisse* aus.

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. Iterationen

enterprise edition Dieses Feature ist nur in der Klaros-Testmanagement Enterprise Edition verfügbar

Eine Iteration verkörpert einen Testzyklus innerhalb eines Projektes und unterstützt damit das Synchronisieren des Testprozesses mit einem agilen Entwicklungsprozess, z.B. Scrum. Eine Iteration kann einen Meilenstein in einem Projekt darstellen, beispielsweise kann in einem Software-

entwicklungsprojekt der Meilenstein *Core Funktionalität stabil* durch eine Iteration abgebildet werden.

In diesem Abschnitt erzeugen Sie eine Iteration, lernen wie Sie Iterationen aktivieren und deaktivieren und wie Sie Anforderungen einer Iteration zuordnen können.

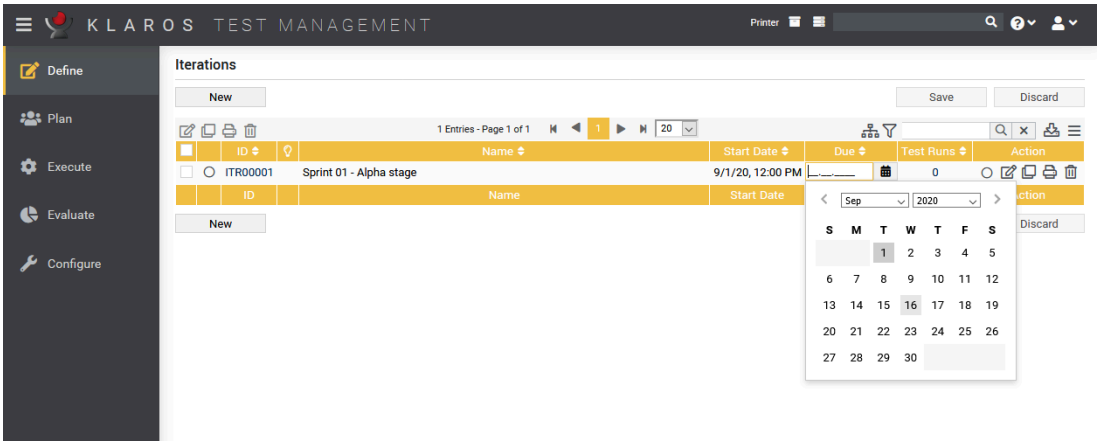


Abbildung 3.14. Die Iterationen Seite

1. Wählen Sie den Bereich **Definieren** in der seitlichen Menüleiste aus.
2. Wählen Sie den Eintrag **Iterationen** in der seitlichen Menüleiste.
3. Klicken Sie auf den Button Neu .
4. Fügen Sie `Sprint 01 - Alpha stage` in the *Name* text field.
5. Klicken Sie auf das Icon .
6. Wählen Sie den *Eigenschaften* Tab an.
7. Geben Sie `Alle Drucker Modelle drucken die Testseite ohne Fehler.` und `Alle Drucker Modelle drucken mindestens 10 Seiten pro Minute.` in das mit *Erfolgskriterien* bezeichnete Textfeld ein.
8. Klicken Sie auf den Speichern Button.

Sie haben nun erfolgreich eine erste Iteration angelegt. Aktivieren Sie diese mit dem Radio-Button links neben der Projekt-ID oder rechts in der Aktionsspalte. Sie können die aktive Iteration in der oberen Leiste sehen (siehe Abschnitt 3.9, „Iterationen“).

Abbildung 3.15. Die ausgewählte Iteration in der oberen Menüleiste

Tipp

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

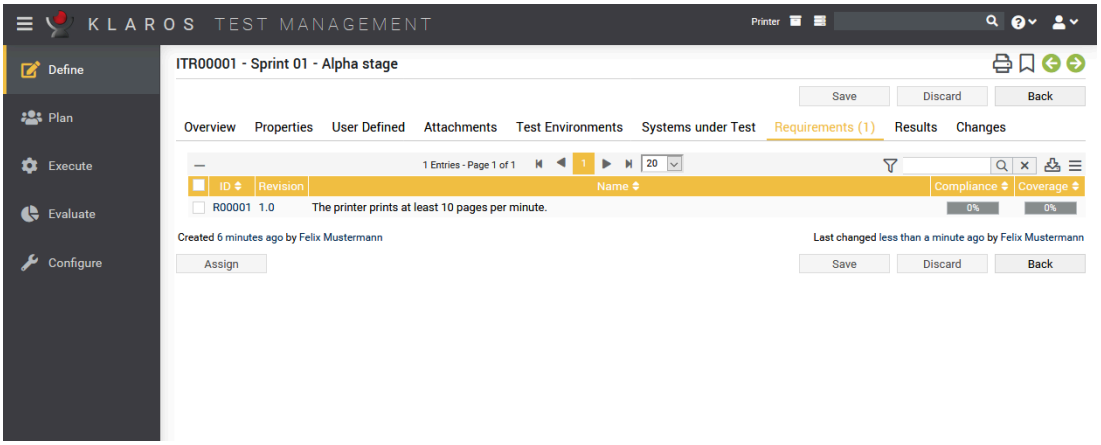


Abbildung 3.16. Die Iteration Detail Seite

1. Wählen Sie den Eintrag *Anforderungen* in der seitlichen Menüleiste aus.
2. Zuweisen Button.
3. Wählen Sie die Anforderung *Der Drucker druckt mindestens 10 Seiten pro Minute* aus.
4. Klicken Sie auf den Zuweisen Button.

Sie haben nun erfolgreich eine Anforderung mit einer Iteration verknüpft.

Sobald eine Iteration ausgewählt wird, werden nur zu dieser Iteration gehörige Artefakte angezeigt. Um zu einer anderen Iteration zu wechseln oder die aktuelle Auswahl aufzuheben folgen Sie bitte den nächsten Schritten.

1. Klicken Sie auf das Icon in der Topbar.
2. Wählen Sie den leeren Eintrag in der Iterationen Dropdown Box aus.

Kapitel 4. Erstellen von benutzerdefinierten Berichten

Dieses Tutorial gibt einen Überblick über die Erstellung von benutzerdefinierten Berichten in Klaros-Testmanagement. Bitte beachten Sie, dass benutzerdefinierte Berichte der Klaros-Testmanagement Enterprise Edition vorbehalten sind. Grundkenntnisse in Java Programmierung und XML sind erforderlich, um diesem Tutorial zu folgen. Den in diesem Tutorial erstellten Report finden Sie hier. Den Sourcecode können Sie sich hier herunterladen und nach Ihren Vorstellungen modifizieren.

Um benutzerdefinierte Berichte anzulegen, navigieren Sie über den Eintrag *Konfigurieren* und *Berichtsvorlagen* in der seitlichen Menüleiste zur Übersicht der benutzerdefinierten Berichte der Klaros-Testmanagement Enterprise Edition.

Dort können Sie über einen Klick auf den Button Neu einen neuen Bericht erstellen. Den Bericht können Sie dann später über den Eintrag *Auswerten* in der oberen Menüleiste und den Eintrag *Berichtsvorlagen* in der seitlichen Menüleiste aufrufen.

4.1. Entwicklungsumgebung

Als ersten Schritt zeigt dieses Tutorial wie Sie die Entwicklungsumgebung aufsetzen. In diesem Tutorial setzen wir Eclipse als Entwicklungsumgebung ein.

4.1.1. Anlegen eines Berichts-Projektes

- Starten Sie Eclipse und wählen Sie *File -> New -> Other...*.
- Im folgenden Dialog wählen Sie wie im Bild unten dargestellt *Java -> Java Project* und klicken auf *Next*.

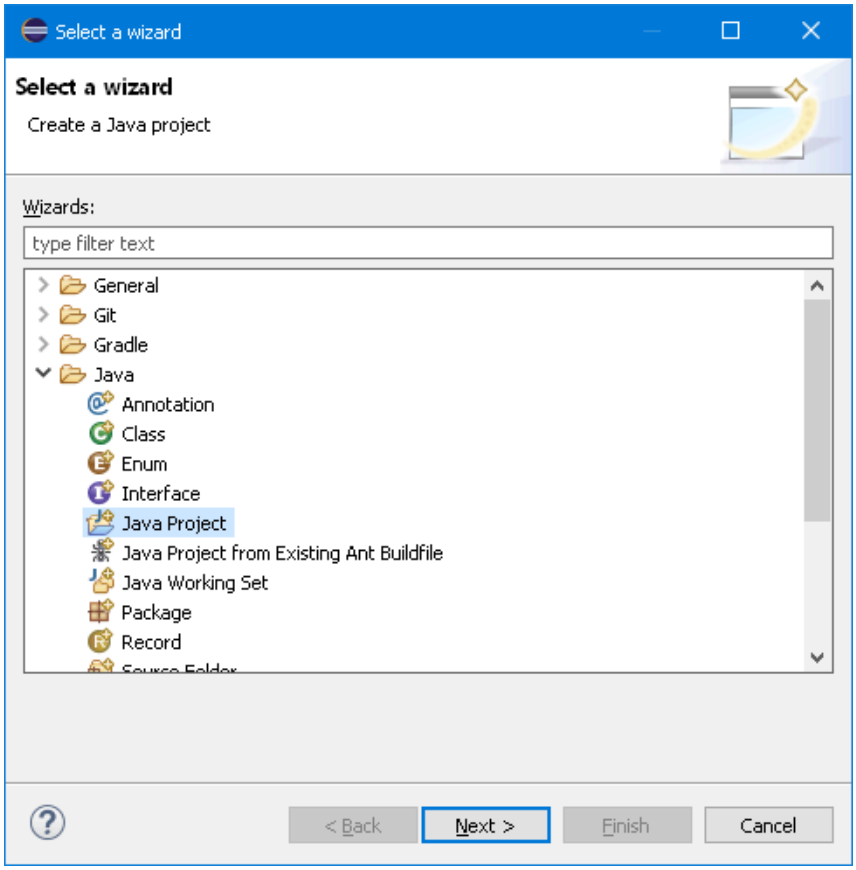


Abbildung 4.1. Die Dialog-Auswahl

- Im nächsten Dialog vergeben Sie den Namen für das Projekt (z.B. `ReportTutorial`) und klicken auf *Finish*.

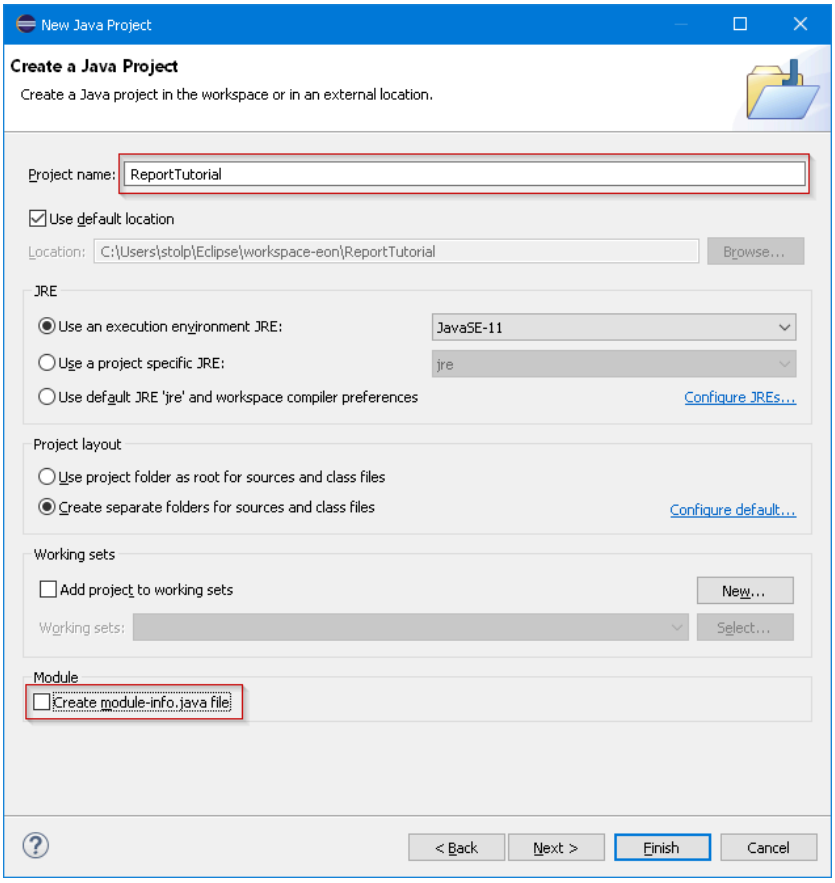


Abbildung 4.2. Der Dialog ""New Project"

- Der *Package Explorer* in Eclipse sollte jetzt das neue Projekt anzeigen.

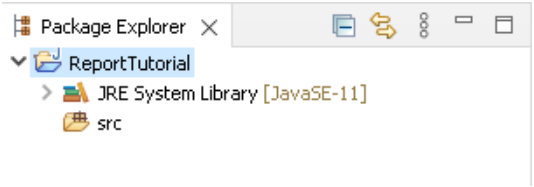


Abbildung 4.3. Der Paket-Explorer

4.1.2. Einrichten des Projektes

- Zuerst legen Sie einen Ordner an, welcher die XML Dateien aufnimmt. Klicken Sie mit der rechten Maustaste auf das Projekt und wählen Sie *New -> Folder*. Im folgenden Dialog geben Sie `xml` in das Feld *Folder name* ein und klicken auf Finish.

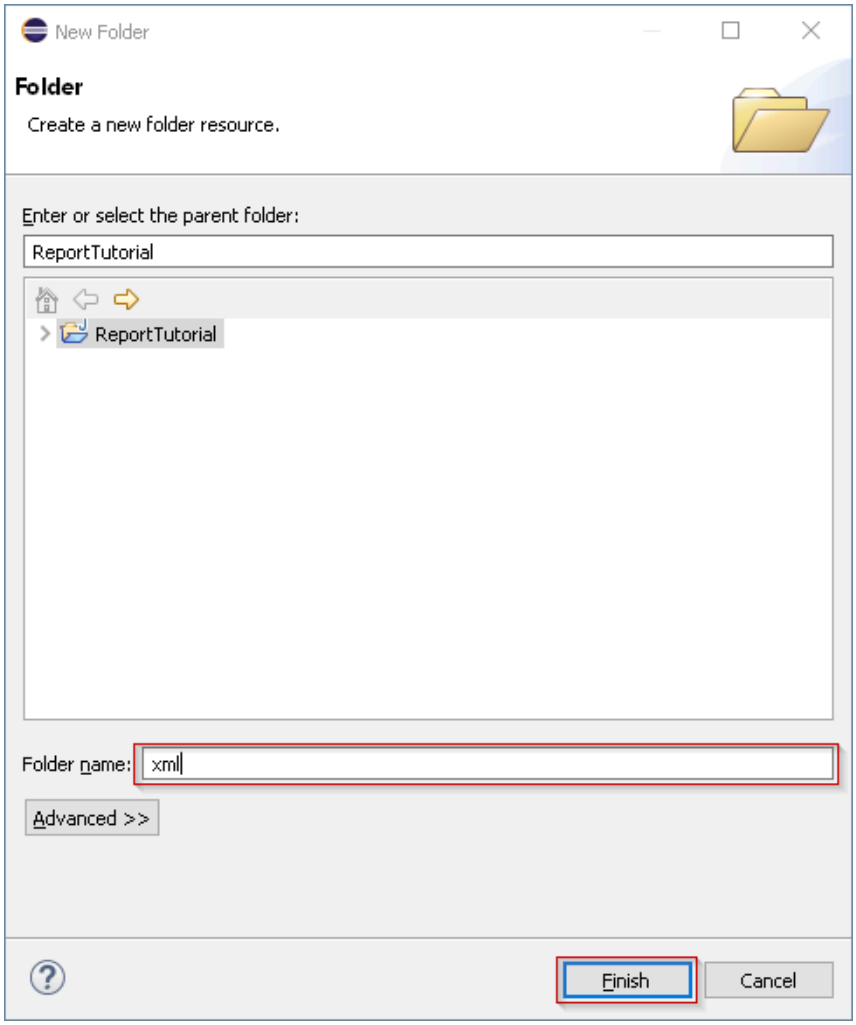


Abbildung 4.4. Anlegen eines neuen Ordners

- Fügen Sie die benötigten Bibliotheken zum Build Pfad des Projekts hinzu. Dazu klicken Sie mit der rechten Maustaste auf das Projekt und wählen Sie *Properties* im Menu aus. Im folgenden Dialog klicken Sie auf *Java Build Path*(1) und wählen den Tab *Libraries*(2) aus. Dann klicken Sie auf *Add External JARs...*(3).

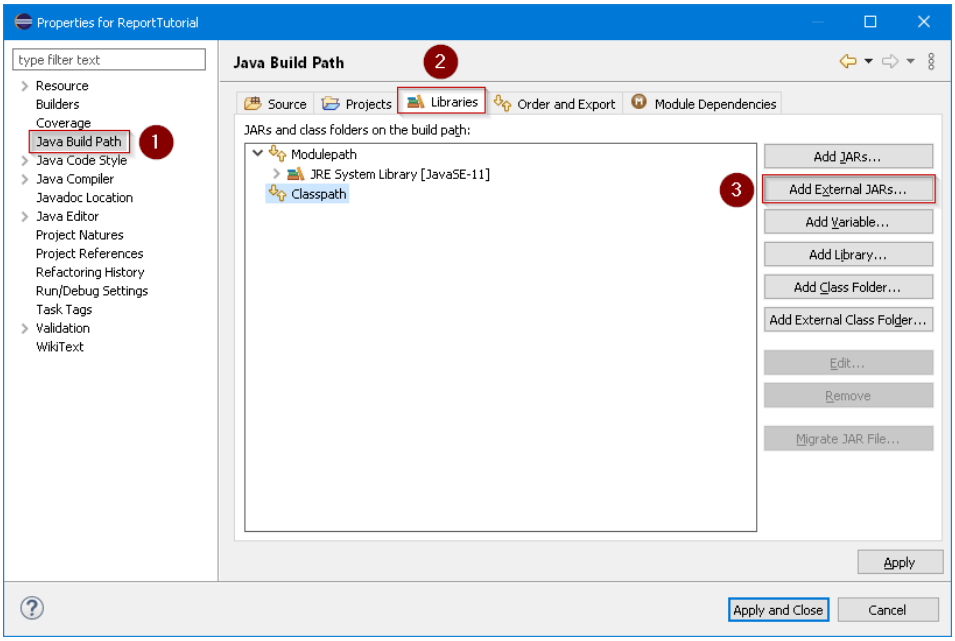


Abbildung 4.5. Die Build Path Einstellungen

- Im folgenden Dialog navigieren Sie zum Klaros-Testmanagement Installationsordner. Von dort navigieren Sie weiter zum *webapps/klaros-web/WEB-INF/lib* Ordner und wählen die Jar Dateien *klaros-commons-x.y.z.jar*, *klaros-core-x.y.z.jar*, *klaros-model-x.y.z.jar* und *klaros-scripting-x.y.z.jar* aus, dann klicken Sie auf den Open bzw. Öffnen Button.

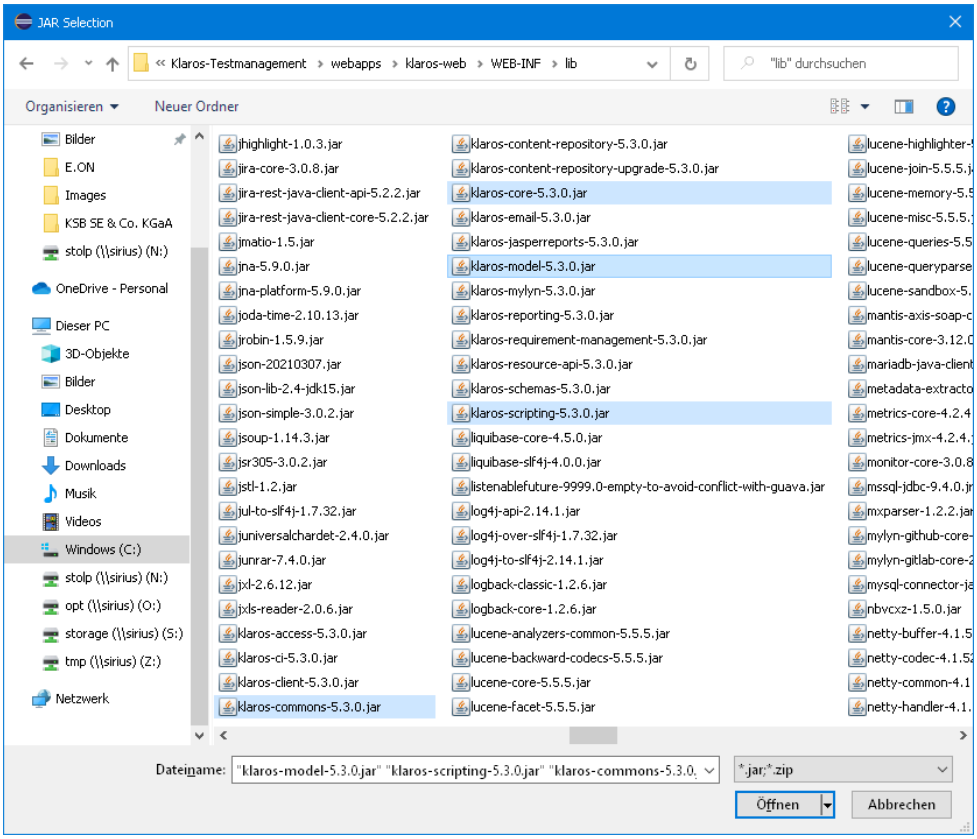


Abbildung 4.6. Auswahl einer externen Jar-Datei

- Zum Schluss klicken Sie auf den Apply and Close Button des *Build Path* Dialogs. Sie sollten jetzt einen weiteren Eintrag namens *Referenced Libraries* in Ihrem Projekt vorfinden.

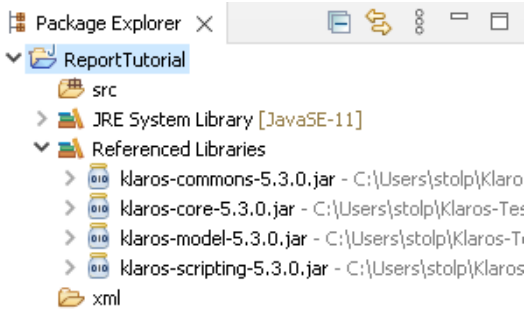


Abbildung 4.7. Der Package Explorer

4.2. Datenaufbereitung für den Report

Im nächsten Schritt sehen Sie, wie Sie Ihre Daten für Ihren Bericht aufbereiten können. Der Bericht in diesem Tutorial zeigt eine Zusammenfassung aller Testläufe für alle Testfälle einer Testsuite an. Weiterhin zeigt er eine Beschreibung aller ausgeführten Testschritte pro Testlauf, sortiert nach Testfall. In der Klaros API Dokumentation finden Sie eine Übersicht über das zur Verfügung stehende Datenmodell.

Das folgende Bild zeigt wie Sie eine KlarosScript Klasse anlegen.

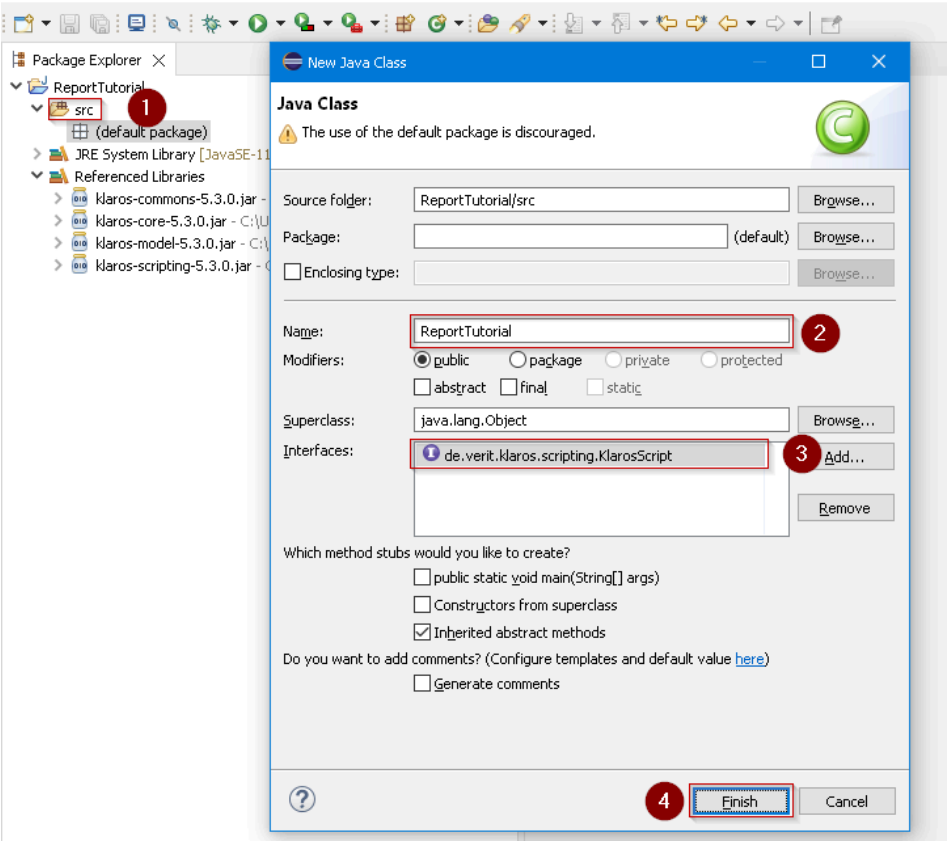


Abbildung 4.8. Anlegen der KlarosScript Klasse

- Zuerst legen Sie eine Klasse an, welche das KlarosScript-Interface implementiert. Dazu klicken Sie mit der rechten Maustaste auf den *src* Ordner (1) in Ihrem Projekt und wählen *New -> Class*.
- Im folgenden Dialog geben Sie einen Namen für Ihre Script Klasse ein (2)
- Klicken Sie dann auf die Schaltfläche `Add` (3) und geben Sie **`KlarosScript`** ein, um das Interface KlarosScript für die Klasse festzulegen. Drücken Sie anschließend auf `OK`, um den Dialog zu schließen.
- Schließlich klicken Sie auf den `Finish` Button (4). Nach kurzer Zeit ist die Klasse angelegt und Eclipse zeigt diese an.

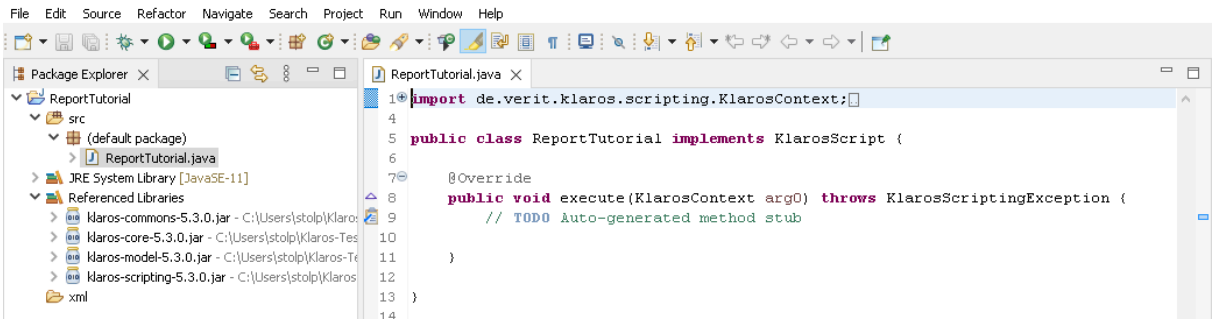


Abbildung 4.9. Die neu erzeugte Klasse

- Bevor Sie Java Code eingeben können, muss die Klasse noch mit den folgenden Import Statements versorgt werden.

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

- Der auszuführende Code muss in der Methode *execute* der Klasse KlarosScript eingetragen werden. Die Testsuite, für die der Report generiert werden soll, wird als Parameter an die Klasse KlarosScript übergeben. Wir werden später in diesem Tutorial sehen, wie dies im Detail funktioniert. Erreicht wird dies durch die folgende Codezeile:

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

- Als nächstes wird auf die Datenbank zugegriffen, um die Daten für den Bericht abzurufen. Das folgende Snippet zeigt, wie der Zugriff auf die Datenbank erfolgt. Dieser Code wählt alle Einträge aus der Tabelle KlarosTestSuite aus, deren ID mit der an KlarosScript übergebenen Test-Suite-ID übereinstimmt und die im aktuell aktiven Projekt vorhanden sind. Das Ergebnis wird als Liste zurückgegeben, wobei wir nur einen einzigen Eintrag zurückerwarten.

```
        // 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);
```

- Um Fehler bei der Ausführung des KlarosScript zu vermeiden, wird folgender Code für den Fall hinzugefügt, dass keine passende Testsuite gefunden wurde.

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

```
 ...
}
```

- Jetzt fügen wir die Testsuite dem Context hinzu, damit die Template-Engine später darauf zugreifen kann.

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

- Jetzt werden die Daten für das Tortendiagramm aufbereitet. Der Report soll ein Diagramm anzeigen, welche die Anzahl der Zustände der Testfall-Ergebnisse widerspiegelt. Es gibt vier mögliche Ergebnistypen: `Passed`, `Error`, `Failure` und `Skipped`. Hierzu wird eine Liste von Klaros-TestCaseResult für jeden möglichen Ergebnistyp angelegt. Auch dieser Code wird innerhalb der if-Anweisung angelegt.

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

- Nun werden die Listen gefüllt. Zuerst wird über die Liste der Ergebnisse der Testsuite iteriert.

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

- Innerhalb der Testsuite-Ergebnis-Schleife wird über die Testergebnisse jedes Testfalls iteriert. Der folgende Code wird dazu in die im vorhergehenden Schritt angelegte for-Schleife eingefügt.

```
for (KlarosTestCaseResult testCaseResult : testSuiteResult.getResults()) {
...
}
```

- Nun haben Sie die Testergebnisse vorliegen und können sie abhängig vom Ergebnis auf die vier Listen verteilen. Der folgende Code wird in die im vorhergehenden Schritt erstellte for-Schleife eingefügt.

```
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);
```

- Wir sind fast fertig. Im letzten Schritt müssen wir die vier Listen, welche die KlarosTestCaseResults beinhalten, noch im Context ablegen, damit die Template-Engine darauf zugreifen kann. Der folgende Code wird innerhalb der if-Anweisung, jedoch außerhalb der zwei for-Schleifen eingefügt.

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

- Die vollständige Klasse finden Sie in folgendem Archiv.

4.3. Definieren des Berichts-Layouts

Klaros-Testmanagement verwendet das *JBoss Seam PDF* Framework um den Bericht zu erzeugen und die bereitgestellten Daten in das Layout-Template einzufügen.

1. Wir starten mit einem leeren Dokument, das nur eine Kopf- und eine Fußzeile enthält.

```
<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>
```

Dieses Codefragment erzeugt eine Kopfzeile, welche auf jeder Seite angezeigt wird. Diese beinhaltet das Datum der Erstellung und den Namen des Klaros-Testmanagement Benutzers, welcher den Report erzeugt hat. Die Fußzeile enthält die Seitennummer und einen Text, dass der Report mit Klaros-Testmanagement erstellt wurde. Für die Kopfzeile wurde das Attribut `borderWidthBottom` verwendet um die Kopfzeile vom restlichen Inhalt der Seite abzusetzen. Entsprechend wurde für die Fußzeile das Attribut `borderWidthTop` gesetzt.

2. Als nächstes erstellen wir eine Titelseite, auf der die wichtigsten Informationen über den Bericht zusammengefasst sind.

   Für die Formatierung des Deckblattes verwenden wir hauptsächlich `<p:paragraph>` Elemente. Diese Elemente beschreiben die Textposition über ihre Attribute `alignment` und `spacing`. Um die Schriftart der Paragraphen zu wechseln, können Sie das `<p:font>` Element benutzen. Um Text hervorzuheben, können Sie die `style` und `size` Attribute des `<p:font>` Elements benutzen.

```
  <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 />
```

Das resultierende Deckblatt finden Sie nachfolgend.

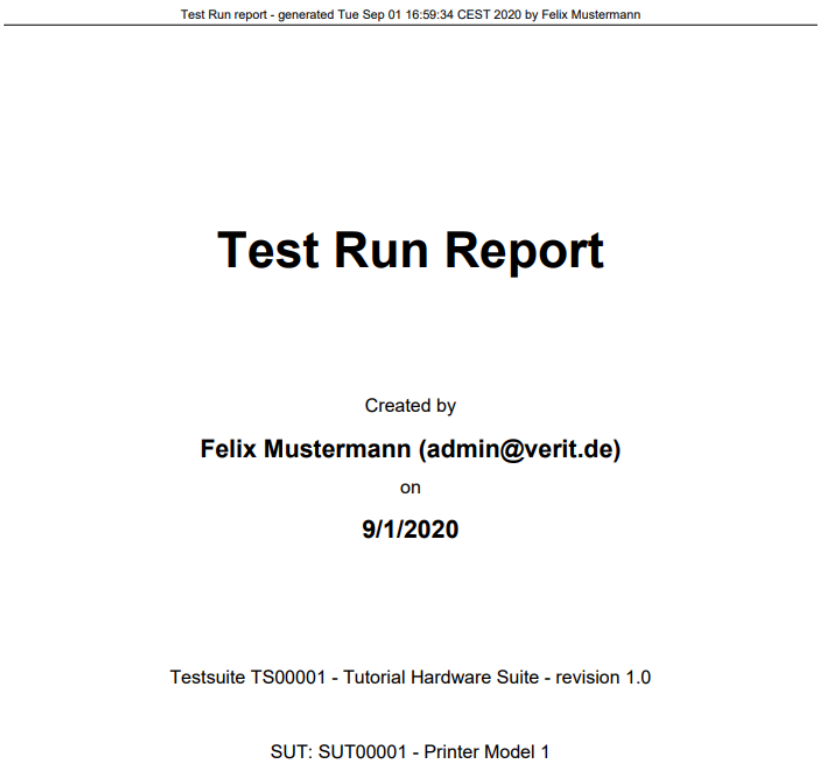


Abbildung 4.10. Beispiel Deckblatt für einen Report

Wie bei der Kopf- und Fußzeilendefinition gesehen, kann auf die im Kontext gespeicherten Werte zugegriffen werden indem man der in geschweiften Klammern eingeschlossenen Kontext-

variablen ein führendes # Zeichen vorangestellt wird, z. B. *#{Benutzername}*. Wir haben dies bereits verwendet, als wir die Daten für diesen Bericht vorbereitet und die Testsuite dem Kontext hinzugefügt haben. Die Kontextvariable für den Benutzer wird durch Klaros-Testmanagement automatisch in den Kontext eingefügt.

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

Für das Deckblatt werten wir einige Attribute der Testsuite aus. Diese werden entsprechend dem Benutzer im letzten Schritt eingefügt. *#{testSuite.name}*, *#{testSuite.shortname}*, und *#{testSuite.revisionId}*. Die benutzbaren Attribute der Testsuite finden Sie in der Klaros-Testmanagement API Documentation in der Online Dokumentation. Auf alle Attribute, die eine get-Methode haben, kann zugegriffen werden. Für `#{testSuite.name}` finden Sie die Methode `getName()` in seiner abgeleiteten Schnittstelle IKlarosLabeledObject.

Um einen Seitenumbruch in das Dokument einzufügen, können Sie folgendes verwenden:

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

3. Das Deckblatt ist fertig, jetzt geht es an die Aufbereitung der Daten. Um einen schnellen Überblick zu bieten wird zuerst ein Tortendiagramm für die Ergebnisse der Testsuite eingefügt.

```
<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>
```

Ein Beispiel für das erzeugte Tortendiagramm finden sie unten.

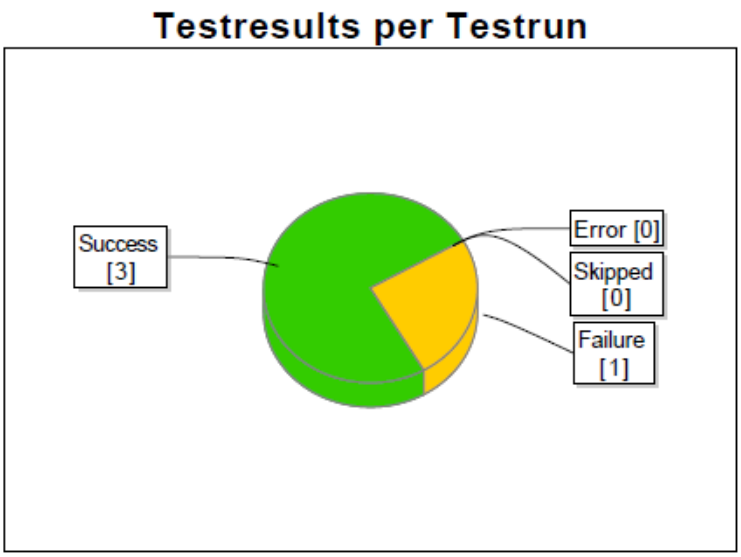


Abbildung 4.11. Das erzeugte Tortendiagramm

Eine detaillierte Anleitung zum Layout des Tortendiagramms und zu anderen Charts finden Sie in der seam-pdf documentation. Erinnern Sie sich noch daran, dass wir vier `List<KlarosTestCaseResult>` Objekte in den Context eingefügt haben:

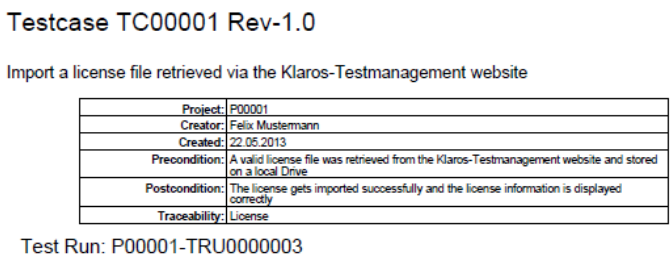
```
context.add("error", error);
context.add("failure", failure);
context.add("success", success);
context.add("skipped", skipped);
```

Diese vier Listen repräsentieren die Ergebnisse innerhalb der Testsuite. Um das Tortendiagramm zu berechnen, verwenden wir die Anzahl der Elemente der einzelnen Listen:

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

4. Als nächstes wollen wir für jeden in der Testsuite enthaltenen Testfall eine eigene Seite erzeugen und dessen Ergebnisse in einer Tabelle darstellen.

   Da dies ein wenig langatmig werden könnte, werden hier nur Fragmente des Codes vorgestellt. Ein Beispiel finden sie unten.



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

Abbildung 4.12. Der Test Case Report

Zuerst wird eine Schleife über alle Testcases der Testsuite benötigt:

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

Eine Schleife in der Template-Engine erzeugen wir mittels des *<ui:repeat>* Elements. Das Attribut `value` bekommt eine Collection zugewiesen, in unserem Fall die Liste der Testfälle. Das Attribut `var` definiert eine Variable, welche innerhalb der Schleife zur Adressierung des aktuellen Wertes verwendet werden kann, z.B. `testCase`. Mit jedem Durchlauf der Schleife bekommt die Variable `testCase` den nächsten Wert aus der Collection.

Nun zur Tabelle. Wir benötigen eine zweispaltige Tabelle und die Breite der zweiten Spalte sollte die dreifache Größe der ersten Spalte haben. Der Code dafür sieht aus wie folgt:

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

Nun wird die Tabelle mit Daten gefüllt. Dazu muss definiert werden, wie jede Tabellenzelle auszusehen hat. Der folgende Code wird zwischen die `<p:table>` Elemente eingebettet.

```
<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>
  ...
```

Die Tabelle soll zwei Spalten haben, somit müssen auch zwei Zellen pro Zeile angegeben werden. In obigem Code beinhaltet die erste Zelle einer Zeile den Namen des Testprojektes, während die zweite Zelle den Namen des Benutzers, der den Testcase erstellt hat, beinhaltet. Es ist wie am `<p:font>` Element ersichtlich möglich, den Text innerhalb einer Zelle zu formatieren.

5. Als nächstes sollen die Testcase Ergebnisse angezeigt werden. Hier kommt jetzt ein weiteres Element ins Spiel. Wenn für einen Testfall kein Ergebnis vorliegt, so soll ein Standardtext anstatt einer leeren Zelle angezeigt werden.

```
<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>
```

Achten Sie auf das `<ui:fragment rendered="...">` Element. Dieser Teil des Dokumentes wird nur ins PDF aufgenommen wenn die Bedingung im `rendered`-Attribut zu `true` ausgewertet wird. Diese Anweisung ist mit if-Anweisungen in anderen Programmiersprachen vergleichbar. Im Ausdruck `#{testCase.results.isEmpty()}` wird die `isEmpty()` Methode der Testfall-Ergebnisliste aufgerufen, welche zu einem boole'schen Wert ausgewertet wird. Als Nächstes wird ein Block definiert, der die Daten aufbereitet, wenn die Ergebnisliste nicht leer ist:

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

Beachten Sie bitte das `!` welches den Ausdruck, den wir zuvor benutzt haben, negiert. Der Code innerhalb dieses Blocks wird ausgeführt, wenn die Liste der Ergebnisse mindestens einen Eintrag besitzt.

6. Jetzt wird über die Testcase Ergebnisse iteriert, welche alle Testläufe eines Testcase beinhalten. Ein Beispiel für einen Testlauf finden Sie unten.

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>
```

Dieser Code liefert ein Testergebnis in der Variable `testResult`. Das nächste Codefragment beinhaltet die Anzeige einer Zusammenfassung des Testergebnisses, die dem Code zur Anzeige der Testfallzusammenfassung sehr ähnlich ist. Den gesamten Code finden Sie beigefügten Quellcode.

Next, the test steps and their results are displayed in the current test run.

Als nächstes werden die Testschritte und deren Ergebnisse im aktuellen Testlauf angezeigt.

7. Dazu muss eine weitere Schleife innerhalb der Testergebnis-Schleife angelegt werden.

```
<!-- 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>
```

Innerhalb des `<ui:repeat>...</ui:repeat>` Blocks wird eine Tabelle angelegt. Diese Tabelle soll angezeigt werden, sobald es mindestens ein Ergebnis zu einem Testschritt gibt. Die Tabelle zeigt die Vorbedingung, Aktion und Nachbedingung des Testschritts an, ebenso wie das Testschrittergebnis, die Ergebniszusammenfassung und die Testschrittbeschreibung. Die Testschrittergebnis-Zelle wird entsprechend dem Ergebnis eingefärbt, z.B. grün, wenn der Testschritt erfolgreich absolviert wurde. Jede Tabelle hat die Testschrittanzahl als Überschrift.

8. Anzeigen der Testschrittnummer.

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

Hier sehen Sie wie Sie den Index des Testschrittergebnisses aus der Liste der Testergebnisse abfragen können. `testStepResult` ist die Variable der innersten Schleife, während `testResult` die Schleifenvariable der äußeren Schleife bezeichnet. Da die Zählung bei null beginnt, müssen wir den ermittelten Wert inkrementieren, sonst würde der erste Schritt als Schritt 0 angezeigt werden.

9. Die Tabelle soll nur dann angezeigt werden, wenn der Testfall mindestens einen Testschritt beinhaltet.

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

10. Abfragen der Testschritt Eigenschaften (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>
```

Um die Testschritteigenschaften anzusprechen wird der Testcase benötigt. Über den Index des Testschrittergebnisses innerhalb der Testschritte wird der entsprechende Testschritt geholt und die Eigenschaften wie `precondition`, `action` und `postcondition` können abgefragt werden.

11 Einfärben einer Tabellenzelle abhängig vom Testschrittergebnis.

```
<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>
```

Die Zellen werden abhängig vom Status der Testschritte eingefärbt. Dazu werden die Methoden `isError()`, `isFailure()` etc. ausgewertet. Die Zellen werden eingefärbt, indem das `backgroundColor`-Attribut auf den gewünschten RGB Wert gesetzt wird.

Glossar

A

Abdeckung, Anforderungsabdeckung

Die Anforderungsabdeckung stellt das Verhältnis der durch Tests abgedeckten Anforderungen zu der Gesamtmenge der definierten Anforderungen dar.

In Klaros-Testmanagement können Anforderungen direkt erfasst und Testfällen zugeordnet werden. Dabei können einzelne Anforderungen durch mehrere Tests abgedeckt sein und ein Test kann mehr als eine Anforderung abdecken.

Administrator

Eine Benutzerrolle, die vollständigen Zugriff auf alle Funktionen in Klaros-Testmanagement hat.

Dazu gehören u.a. Installation und Wartung der Software, Updates, Systemeinstellungen, Benutzerverwaltung, Backups und alle Rechte um Tests anzulegen, auszuführen und auszuwerten.

Änderungsverfolgung

Lückenloses Nachvollziehen aller Änderungen an den verwalteten Objekten durch automatisches Protokollieren der Aktivitäten.

In Klaros-Testmanagement werden alle Änderungen auf der Detailseite "Änderungen" des entsprechenden Objektes farbig und zeitlich dargestellt.

Anforderung

Eine vom Benutzer benötigte Eigenschaft oder Fähigkeit, die eine Software erfüllen oder besitzen muss, um einen Vertrag, einen Standard, eine Spezifikation oder ein anderes formales Dokument zu erfüllen. [Nach IEEE 610].

Anforderungen lassen sich entweder in Klaros-Testmanagement direkt verwalten oder mit externen Quellen wie z.B. JIRA synchronisieren. Ein Verweis auf die entsprechende Anforderung, z.B. auf ein Dokument, kann Testfällen und Testsuiten hinzugefügt werden (Verfolgbarkeit).

Arbeitsprotokoll

Das Arbeitsprotokoll zeigt den zeitlichen Verlauf und die Dauer der Aktivitäten zur Bearbeitung einer Aufgabe an. Die für die gesamte Testausführung aufgewendete Zeit ergibt sich aus der Summe der Ausführungszeiten.

Aufgabe

Mittels Aufgaben werden Testaktivitäten geplant und protokolliert. Die Ausführung und die Ergebnisse der ausgeführten Aufgaben werden automatisch mitprotokolliert und zur Auswertung herangezogen.

Authentifizierung

Die Authentifizierung der Benutzer erfolgt in der Regel über Benutzername und Passwort.

In Klaros-Testmanagement kann das Verwalten der Zugangsdaten direkt in der Software oder über ein externes LDAP- oder Active Directory-Verzeichnis erfolgen.

Automatisierte Testfälle

Die Testergebnisse aus Automatisierungstools werden in Form einer Ergebnisdatei in Klaros-Testmanagement eingespielt. Liegen sowohl manuelle als auch automatisierte Testergebnisse vor, können sie gemeinsam ausgewertet werden.

B

Benachrichtigungsereignis

Benachrichtigungsereignisse sind Auslöser für eine automatische E-Mail-Benachrichtigung an bestimmte Benutzer. Diese werden verschickt, sobald das vorgesehene Ereignis eingetreten ist.

Mögliche Benachrichtigungsereignisse sind z.B. „Aufgabe zugewiesen“, „Benutzerkonto angelegt“, „Testausführung fehlgeschlagen“, „Benutzerkonto-Passwort geändert“ und „Aufgabe bereit zur Ausführung“.

Berichte (Reports)

Berichte geben Auskunft über den Stand der Testaktivitäten, den Zustand der Software, Hinweise auf kritische Komponenten und ermöglichen eine Vorausplanung.

Dazu werden die Daten aus der Klaros-Testmanagement-Datenbank nach bestimmten Kriterien geordnet und in textuelle und grafische Übersichten aufbereitet. Beispiele für Berichte sind „Projektübersicht“ und „Testaktivität“ sowie Übersichten über Testfortschritt, Testergebnisse und Testläufe.

Klaros-Testmanagement enthält zahlreiche grundlegende Berichte bereits vordefiniert. Individuelle Berichte können selbst erstellt werden. Alle Berichte sind in verschiedene Formate wie PDF oder Excel (auch zu weiteren Bearbeitung) exportierbar.

Berichtsvorlagen

In Berichtsvorlagen sind bereits grundlegende Bestandteile vordefiniert und können vom Benutzer nach Bedarf beliebig angepasst werden.

Bereits integrierte, vordefinierte Berichtsvorlagen in Klaros sind: „Testumgebungsbericht“, „Testsystembericht“, „Testsuitebericht“ und „Testlaufbericht“.

Bestanden (Bewertung)

Ein Test wird als Bestanden bezeichnet, wenn das tatsächliche mit dem vorausgesagten Ergebnis übereinstimmt. (ISTQB)

Bewertung

Als Bewertung wird das Ergebnis des ausgeführten Testfalls, Testschritts oder der Testsuite bezeichnet. Mögliche Bewertungen sind „Bestanden“, „Fehlschlag“, „Fehler“, „Unklar“, „Übersprungen“.

Bugzilla

Bugzilla ist ein Open Source-Issue Management System. Ausführliche Informationen sind auf der Bugzilla-Webseite zu finden.

Klaros-Testmanagement verfügt über eine Anbindung an Bugzilla.

C

Community Edition

Die kostenlose Edition von Klaros-Testmanagement. Sie hat einen eingeschränkten Feature-Umfang, kann aber frei auch für kommerzielle Zwecke eingesetzt werden. Sie kann von der Webseite unter Download heruntergeladen werden.

Continuous Integration

Kontinuierliche Integration (Continuous Integration, CI) ist die automatisierte Integration von Codeänderungen von mehreren Mitwirkenden in ein einzelnes Softwareprojekt. Es ist eine wichtige DevOps-Best-Practice, die es Entwicklern ermöglicht, Codeänderungen regelmäßig in einem zentralen Repository zusammenzuführen, in dem dann Builds und Tests ausgeführt werden.

Ein Plug-in für den Jenkins Continuous Integration Server überträgt automatisch die Testergebnisse eines Builds an Klaros Test Management, wo sie ausgewertet werden können.

D

Dashboard

Ein Dashboard zeigt eine Sammlung von Informationen, die typischerweise in Form von Diagrammen dargestellt werden.

In Klaros-Testmanagement zeigen Dashboards die Auswertungen zu einem bestimmten Testprojekt an. Die angezeigten Diagramme können individuell zu einem Dashboard zusammengestellt werden. Jeder Benutzer kann beliebig viele Dashboards zusammenstellen.

Dashboards können privat (nur für den Ersteller sichtbar) oder öffentlich sein. Administratoren können ein Standard-Dashboard angelegen.

Datenbanksystem

Eine Datenbank ist eine Sammlung von Informationen, organisiert in in Wechselbeziehung stehenden Tabellen von Daten und Spezifikationen von Datenobjekten.

Klaros-Testmanagement benötigt ein Datenbanksystem zur Speicherung der Objekte und wird mit einer vorkonfigurierten Apache-Derby-Datenbank ausgeliefert. Weitere unterstützte Datenbanksysteme sind MariaDB, Microsoft SQLServer, MySQL und PostgresSQL.

Detailseite

Jedes Objekt (wie Testfälle, Anforderungen, Iterationen u.s.w.) besitzt eine eigene Detailseite mit verschiedenen weiteren Unteransichten. Diese variieren je nach Objekt und zeigen detaillierte Informationen zu beispielsweise *Eigenschaften*, *Anhängen*, *Änderungen* und *Ergebnissen* an.

Docker

Docker ist eine freie Software zum Isolieren von Anwendungen durch Containervirtualisierung. Die Container enthalten alle nötigen Pakete und lassen sich so leicht als Dateien transportieren und installieren.

Klaros-Testmanagement kann ebenfalls als Container innerhalb einer Docker-basierten Umgebung betrieben werden. Einsatzbereite Docker-

Images für verschiedene Datenbanken sind bereits verfügbar. Eine ausführliche Dokumentation dazu ist unter GitHub zu finden.

Druckansichten

Alle Objekte lassen sich in deiner druckfreundlichen Darstellung ausgeben. Mit verschiedenen Parametern kann der Detailgrad des Ausdrucks festgelegt werden.

E

Enterprise Edition

Die kommerzielle Edition mit vollem Support ist pro Benutzer lizenziert. Die initiale Installation enthält 3 Benutzer. Eine kostenlose 30-Tage-Demo kann unter Demo angefordert werden.

Ergebnisart

Der erwartete Ausgang des Tests: Positiv oder Negativ.

Ergebnis, erwartetes, vorausgesagtes

Das Verhalten eines Systems oder einer Komponente unter festgelegten Bedingungen, das durch die Spezifikation oder durch eine andere Quelle festgelegt ist. (ISTQB)

Evaluierungsart

Die Art der Testergebnisauswertung: Manuell oder Automatisiert.

F

Fehler (Bewertung)

Ein Fehler bei der Testausführung tritt auf, wenn das System den Test nicht korrekt auszuführen kann. Er sollte nicht mit einem Fehlschlag verwechselt werden.

Fehlschlag (Bewertung)

Ein Test schlägt fehl, wenn das aktuelle Ergebnis nicht mit dem vorausgesagten Ergebnis übereinstimmt (ISTQB). Er sollte nicht mit einem Fehler verwechselt werden.

Klaros-Testmanagement unterscheidet die Bewertungen „Bestanden“, „Fehler“, „Fehlschlag“, „Übersprungen“ und „Unklar“.

Funktionales Testentwurfsverfahren

Ein Verfahren zur Herleitung und Auswahl von Testfällen, das auf der Analyse der funktionalen Spezifikation einer Softwarekomponente oder eines Softwaresystems basiert, ohne Bezug auf dessen innere Struktur (ISTQB), auch bekannt als Black-Box- und White-Box-Tests.

G

Gast

Eine Benutzerrolle mit vollständigem, aber nur lesenden Zugriff auf alle Daten. Ein *Gast* kann Berichte in verschiedenen Formaten anzeigen und abspeichern. Diese Rolle ist z.B. für Projektmanager, Kunden oder Reviewer/Gutachter gedacht.

I

ID

Die ID eines Objekts wird beim Erstellen automatisch angelegt und ist nicht änderbar. Alle IDs bestehen aus einem Kürzel für das jeweilige Objekt (z.B. TC für Testfall, ITR für Iteration) und einer aufsteigend vergebenen Zahl. *SEG00025* wäre also das Testsegment 00025.

Integration

Als Integration wird die Anbindung externer Systeme bezeichnet. Klaros-Testmanagement verfügt über Schnittstellen zur Integration mit Issue- und Anforderungsmanagementsystemen, Testautomatisierungs- und Lasttest-Werkzeugen sowie Continuous-Integration-Servern. Außerdem existieren Schnittstellen zu Authentifizierungs- und E-Mail-Servern.

Issue/Problem

Als Issue bezeichnet man einen Teilaspekt, der zur Vervollständigung einer Arbeit erforderlich ist. Dies kann ein Bug sein, ein verlangtes Feature, eine Aufgabe, fehlende Dokumentation o.ä. Ein Issue wird in einem Issue Management System erfasst.

Klaros-Testmanagement besitzt Schnittstellen zu mehreren Issue Management Systemen. Das heißt, gefundene Fehler können direkt an das angeschlossene IMS übergeben werden, ohne die Anwendung verlassen zu müssen.

Issue Management System

Ein Issue Management System (Issue Tracking System) ist eine Software, um Issues zu verwalten.

Issue Tracking System

Siehe Issue Management System.

Iteration

Eine Iteration ist ein vollständiger Entwicklungszyklus, der eine (interne oder externe) Freigabe eines ausführbaren Produkts ergibt. Dieses Produkt ist eine Teilmenge des zu entwickelnden Endprodukts. Die Entwicklung schreitet von Iteration zu Iteration bis zum Endprodukt hin fort (ISTQB).

Klaros-Testmanagement unterstützt agile Entwicklungsprinzipien und verwaltet Iterationen als eigene Objekte.

J

Java

Damit Klaros-Testmanagement ausgeführt werden kann, ist eine Java 64-Bit-Java-11-- Laufzeitumgebung erforderlich. Diese wird mit ausgeliefert und bei der Erstinstallation automatisch verwendet.

JavaScript

JavaScript ist eine in Webbrowsern eingesetzte Programmiersprache und wird für das Ausführen von Klaros-Testmanagement benötigt. Beim Einsatz von JavaScript-Blockern wie NoScript o.ä., muss das Ausführen von JavaScript als Ausnahme zugelassen werden.

JIRA

JIRA ist eine populäre Anwendung zur Fehlerverwaltung, Problembehandlung und dem Projektmanagement der Firma Atlassian Software.

Klaros-Testmanagement verfügt über eine Anbindung an JIRA.

K

Kategorie

Die meisten Objekte können zur besseren Übersicht in Kategorien angeordnet werden. Das Gruppieren der Objekte (Anforderungen, Ite-

rationen, Testsysteme, Testumgebungen, Testfälle oder Testsuiten) in Kategoriebäume kann nach selbst gewählten Kriterien erfolgen. Auch Mehrfachkategorisierungen sind möglich.

Konformität

Die Fähigkeit eines Softwareprodukts, anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen und ähnliche Vorschriften zu erfüllen. [ISO 9126]

Konformitätsrate

In Klaros-Testmanagement zeigt die Konformitätsrate an, wie viele einer Anforderung zugeordnete Testfälle mit dem Ergebnis „Bestanden“ ausgeführt wurden.

L

Layout-Vorlage

Die Layout-Vorlage ist neben dem Skript ein elementarer Bestandteil eines Berichtes. Sie beschreibt den Aufbau des erzeugten Berichtsdokuments in einer XML-basierten Beschreibungssprache. Layout-Vorlagen können an individuelle Anforderungen angepasst werden.

Letzte Erfolgsrate (Berichtsdiagramm)

Der Bericht „Letzte Erfolgsrate“ zeigt die neuesten Ergebnisse der ausgeführten Testfälle für die ausgewählte Kombination aus Testsystem (eins oder mehrere) und Testumgebung (eine oder mehrere) auf dem Dashboard an.

Log-Panel

Das Log-Panel zeigt Statusmeldungen wie Warnungen oder Informationen an. Standardmäßig wird nur die letzte Statusmeldung angezeigt, kann aber mit Klick auf das +-Icon geöffnet werden.

M

Mantis

Mantis (MantisBT, Mantis Bug Tracker) ist eine Open Source-Anwendung zur Fehlerverwaltung und Problembehandlung in der Softwareentwicklung.

Klaros-Testmanagement verfügt über eine Anbindung an Mantis.

N

Nachbedingung

Zustand des Testobjekts/Testsystems (und/oder der Umgebung), in dem sich das Testobjekt/Testsystem (oder die Umgebung) nach Ausführung eines Testfalls oder einer Testsequenz befinden muss. (ISTQB)

O

Objekt

Als Objekte werden in Klaros-Testmanagement Projekt, Testfall, Testschritt, Testsegment, Testschrittergebnis, Testfallergebnis, Testsuite, Testsuiteergebnis, Testumgebung, Testsystem, Testlauf, Anforderung, Iteration und Aufgabe bezeichnet.

P

Projektübersicht (Berichtsdiagramm)

Der Bericht „Projektübersicht“ zeigt die Anzahl der ausgeführten/nicht ausgeführten Testfälle oder Testsuiten und die Anzahl der geplanten/ausgeführten Aufgaben auf dem Dashboard an.

Projekt

Ein Projekt ist der Grundbaustein in Klaros-Testmanagement und enthält alle anderen Objekte, die zur Testausführung notwendig sind.

Q

QFTest

QF-Test QF-Test der Firma Quality First Software ist ein plattformübergreifendes Werkzeug zur GUI-Testautomatisierung.

Klaros-Testmanagement verfügt über eine Anbindung an QF-Test und kann die automatisiert erzeugten Testergebnisse zur weiteren Verarbeitung importieren.

R

Redmine

Redmine ist eine Open Source Anwendung zur Fehlerverfolgung, Problembehandlung und Projektmanagement.

Klaros-Testmanagement verfügt über eine Anbindung an Redmine.

REST (Representational State Transfer)

Representational State Transfer (REST) ist ein Software-Architekturstil, der eine Teilmenge von HTTP verwendet. Er wird häufig verwendet, um Webdienste bereitzustellen.

Klaros-Testmanagement bietet mehrere Integrationsschnittstellen basierend auf REST. Dazu gehören Lesezugriffe auf alle gespeicherten Informationen sowie Importschnittstellen für Testergebnisse, Testfälle und Anforderungen.

Revision

Wird eine größere Änderung an einem Objekt vorgenommen und die vorherige Version wird noch benötigt, sollte eine neue Revsion erstellt werden. Dies wird durch Vergabe einer neuen Revisionsnummer realisiert. Nur ausgewählte Objekte wie z.B. Testfälle oder Anforderungen unterstützen diesen Mechanismus.

Rolle

Eine Benutzerrolle definiert die Rechte des Benutzers. Rollen können global und projektspezifisch festgelegt werden.

In Klaros-Testmanagement gibt es die Benutzerrollen *Administrator*, *Manager*, *Tester* und *Gast*.

S

Selenium

Selenium ist ein Webbrowser-Automatisierungswerkzeug und wird hauptsächlich zum Testen von Web-Apps verwendet.

	Selenium produziert JUnit-XML-konforme Testergebnisse, die in Klaros-Testmanagement zur weiteren Verarbeitung importiert werden können.
Skript	Das Skript ist neben der ein elementarer Bestandteil einer . Es extrahiert die für den Bericht vorgesehenen Daten und liegt als Java-Klasse vor. Skripte können an individuelle Anforderungen angepasst werden.
Sprachen	Die angezeigte Sprache der Benutzer-Oberfläche wird über die vorhandenen Sprachdateien gesteuert. Diese werden im Verzeichnis .klaros/resources/messages gespeichert und unter Konfigurieren / System / Sprachen aktiviert.
SUT	Siehe Testsystem, Zu testendes System, Prüfling, Testobjekt.
System-Zugang	Ein System-Zugang ist für automatisierte Vorgänge wie beispielsweise den Import von Testergebnissen reserviert. Eine Anmeldung über die Login-Seite ist hiermit nicht möglich.

T

Testaktivität (Berichtsdiagramm)	Der Bericht „Testaktivität“ zeigt die Testfallergebnisse für eine ausgewählte Kombination aus einem oder mehreren Testsystemen und Testumgebungen in einem ausgewählten Zeitraum als Histogramm auf dem Dashboard an.
Testart	Eine Gruppe von Testaktivitäten basierend auf spezifischen Testzielen mit dem Zweck, eine Komponente oder ein System auf bestimmte Eigenschaften zu testen (Funktional, Nichtfunktional, Strukturell, Regression, Retest).
Testausführung, Testdurchführung	Der Prozess der Ausführung eines Tests für eine Komponente oder ein System, der Ist-Ergebnisse erzeugt. (ISTQB) Klaros-Testmanagement führt und protokolliert automatisch die Testausführung, siehe Testrunner.
Tester	Eine Benutzerrolle in Klaros-Testmanagement, die Objekte anzeigen sowie Testfälle und Testsuiten ausführen darf.
Testfall	Ein Testfall umfasst eine Menge von Eingabewerten, Ausführungsbedingungen, erwarteten Ergebnissen und Nachbedingungen, welche für ein bestimmtes Testsystem definiert wurden. Sie werden im Hinblick auf ein bestimmtes Ziel bzw. auf eine Testbedingung wie z.B. die Übereinstimmung mit spezifischen Anforderungen entwickelt.
Testfallergebnis	Das Ergebnis der Ausführung eines Tests und seiner Bewertung wie „Bestanden“, „Fehler“, „Fehlschlag“ „Übersprungen“ oder „Unklar“.
Testfall-Priorität	Die Priorität des Testfalls: „Hoch“, „Mittel“ oder „Niedrig“.
Testall-Status	Der Status des Testfalls: „Gesperrt“, „Genehmigt“, „Entwurf“ oder „Auslassen“.

Testfortschritt (Berichtsdiagramm)	Der Bericht „Testfortschritt“ zeigt die Rate der ausgeführten gegenüber den definierten Testfällen eines Projekts für eine bestimmte Menge von Testumgebungen und Testsystemen auf dem Dashboard an.
Testhistorie (Berichtsdiagramm)	Der Bericht „Testhistorie“ zeigt die Rate der definierten Testfälle gegenüber den ausgeführten Testfällen eines Projekts für eine oder mehrere Testumgebungen und ein oder mehrere Testsysteme in einer bestimmten Zeitperiode auf dem Dashboard an.
Testlauf	Die Ausführung eines oder mehrerer Testfälle mit einer bestimmten Version des Testobjekts/Testsystems. (ISTQB) Er bezieht sich immer auf genau ein Testsystem und genau eine Testumgebung.
Testmanager	Die Person, die für das Management der Testaktivitäten, der Testressourcen und für die Bewertung des Testobjekts verantwortlich ist. Zu den Aufgaben gehören Anleitung, die Steuerung, die Verwaltung, Planung und Regelung der Aktivitäten zur Bewertung des Testobjekts. (ISTQB) Eine Benutzerrolle in Klaros-Testmanagement, die vollständigen Lese- und Schreibzugriff auf alle Daten eines Projekts hat. Testmanager können alle Arten von Objekten anlegen, ändern und löschen sowie Projekteinstellungen verwalten.
Testrunner	Klaros-Testmanagement enthält einen webbasierten Client für die Ausführung manueller Tests. Dieser führt den Tester durch die Testschritte, gibt ihm die Möglichkeit Anmerkungen und Anhänge zu erstellen und protokolliert automatisch den Testverlauf und die Testergebnisse.
Testschritt	Ein Testschritt in Klaros-Testmanagement ist die kleinste Aktion, die innerhalb eines Testfalls abgearbeitet wird. Er enthält Ausführungsvorbedingungen, erwartete Ergebnisse und Nachbedingungen der Ausführung für jede einzelne Aktion während der Testausführung.
Testsegment	Ein Testabschnitt ist in Klaros-Testmanagement eine definierte Folge von einzelnen Testschritten, die in einen Testfall eingefügt werden können. Ein Testsegment kann in mehreren Testfällen gleichzeitig verwendet werden. Damit ist es möglich, ein Modulkonzept auf der Basis von Testschritten zu realisieren.
Testsuite	Eine Testsuite in Klaros-Testmanagement ist ein Satz an Testfällen, die zusammen in einer Gruppe ausgeführt werden können.
Testsystem, Zu testendes System, Prüfling, Testobjekt	Ein Testsystem repräsentiert in Klaros-Testmanagement die Softwareversion oder die Produktversion, die getestet werden soll. Je nach Branche sind dafür auch die Bezeichnungen zu testendes System, SUT, Prüfling oder Testobjekt gebräuchlich.
Testsystem (Berichtsdiagramm)	Der Bericht „Testsystem“ zeigt die Erfolgs- und Fortschrittsrate von Testumgebungen unter einem einzigen Testsystem in einem Radardia-

gramm auf dem Dashboard an. Sind weniger als drei Testumgebungen konfiguriert, wird stattdessen ein Balkendiagramm angezeigt.

Testumgebung

Eine Testumgebung repräsentiert die äußeren Umstände, die das Testergebnis beeinflussen können. Beispiele für Testumgebungen sind z.B. das Betriebssystem oder ein Applikations-Server (z.B. Tomcat 9 mit Ubuntu 20.04.2) oder auch Parameter wie Temperatur oder Geschwindigkeit.

Testumgebung (Berichtsdiagramm)

Der Bericht „Testumgebung - Übersicht“ zeigt die Erfolgs- und Fortschrittsrate von Testsystemen unter einer einzigen Testumgebung in einem Radardiagramm auf dem Dashboard an. Sind weniger als drei Testsysteme konfiguriert, wird stattdessen ein Balkendiagramm angezeigt.

Trac

Trac ist eine Open Source-Anwendung zur Fehlerverfolgung und Problembehandlung in der Softwareentwicklung. Klaros-Testmanagement verfügt über eine Anbindung an Trac.

U

Übersichtsseite

Jeder Objekttyp besitzt eine Übersichtsseite. Dort werden die einzelnen Objekte, wie z.B. Testfälle oder Iterationen, tabellarisch angezeigt. Neue Objekte werden ebenfalls hier erstellt.

Übersprungen (Bewertung)

Ein Testfall oder ein Testschritt wurden bei der Ausführung ausgelassen.

Unklar (Bewertung)

Ein Test wird als Unklar bezeichnet, wenn unklar ist, ob das tatsächliche mit dem vorausgesagten Ergebnis übereinstimmt.

Umgebungsvariablen

Umgebungsvariablen sind dynamisch benannte Werte, die das Verhalten von laufenden Prozessen auf einem Computer beeinflussen können. Sie sind Teil der Umgebung, in der ein Prozess läuft.

Durch Setzen der Umgebungsvariablen KLAROS_HOME vor dem Start der Anwendung, kann der Speicherort des Klaros Home-Verzeichnisses an einen anderen Ort verschoben werden.

V

Vorbedingung

Bedingungen an den Zustand des Testobjekts/Testsystems und seiner Umgebung, die vor der Durchführung eines Testfalls oder Testablaufs erfüllt sein müssen. (ISTQB)

W

Webbrowser

Klaros-Testmanagement ist eine Webanwendung und läuft innerhalb eines Webbrowsers. Unterstützte Browser sind Apple Safari, Google Chrome, Microsoft Edge und Mozilla Firefox.

Y

YouTrack	YouTrack ist eine populäre Anwendung zur Fehlerverwaltung, Problembehandlung und dem Projektmanagement der Firma JetBrains.