HAW HAMBURG

BACHELORTHESIS
Miriam Schroth

Testkonzept für automatisierte Tests von internen Webanwendung mit Ranorex und Jenkins

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Miriam Schroth

Testkonzept für automatisierte Tests von internen Webanwendung mit Ranorex und Jenkins

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Bettina Buth
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 24. August 2021

Miriam Schroth

Thema der Arbeit

Testkonzept für automatisierte Tests von internen Webanwendung mit Ranorex und Jenkins

Stichworte

Norddeutscher Rundfunk (NDR), Ranorex, Jenkins, Objekt-Repository, automatisierte Tests

Kurzzusammenfassung

In dieser Arbeit wird ein Testkonzept für automatisierte Tests von Webanwendungen innerhalb des Norddeutschen Rundfunks entwickelt. Bisher erfolgen alle Tests während und nach der Entwicklung manuell. Für die Test- und Produktivumgebung sollen automatisierte Tests eingeführt werden. Das Testkonzept zeigt wie automatisierte Tests mit den Tools Ranorex und Jenkins umgesetzt werden könnten. In Ranorex werden die automatisierte Tests umgesetzt und in Jenkins erfolgt die Überwachung und Steuerung dieser.

Miriam Schroth

Title of Thesis

Test concept for automated tests of internal web application with Ranorex and Jenkins

Keywords

Norddeutscher Rundfunk (NDR), Ranorex, Jenkins, object repository, automated tests

Abstract

In this thesis, a test concept for automated tests of web applications at the Norddeutscher Rundfunk is developed. Until now, all tests during and after development have been carried out manually. Automated tests are to be introduced for the test and production environment. The test concept shows how automated tests could be implemented with

the tools Ranorex and Jenkins. The automated tests are implemented in Ranorex and monitored and controlled in Jenkins.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Öffentlich-Rechtlicher Rundfunk (ÖRR) verbreitet für alle Bundesbürger Hörfunk- und Fernsehprogramme. Die Verbreitung erfolgt über Kabelnetze, Satelliten und terrestrische Sendenetze. Die Programme sind unabhängig von Staat und Wirtschaft. Um die gesamte Bevölkerung zu versorgen wird der ÖRR in das ZDF, die Landesrundfunkanstalten der ARD und dem gemeinsamen DeutschlandRadio aufgeteilt. Eine dieser Landesrundfunkanstalten der ARD ist der Norddeutscher Rundfunk (NDR). Dieser deckt die Bundesländer Hamburg, Mecklenburg-Vorpommern, Niedersachsen und Schleswig-Holstein ab und produziert das ARD-Format Tagesschau. Die Sendungen werden aus unterschiedlichen Bereichen für Fernsehen, Hörfunk und Internet produziert. Für Teile der Produktionen und der Verwaltung von betriebsinternen Abläufen werden Verwaltungstools benötigt. Viele dieser Verwaltungstools sind in eigenen Webanwendungen realisiert, auf die nur Zugriff im Intranet des NDRs besteht.

1.1 Motivation

Die Funktionalität der Verwaltungstools sind für den NDR Grundvoraussetzung, um betriebsinterne Abläufe durchzuführen und die damit verbunden Produktionen umsetzen zu können. Im Rahmen des Praxissemesters wurde eine Webanwendung bereits überarbeitet. Das Testen der Webanwendung erfolgte manuell im Browser. Die manuelle Form des Testens ist für einmalige Tests praktikabel, allerdings nicht für die gesamte Entwicklung von Webanwendungen. Werden Tests wiederholt durchgeführt, so steigt der zeitliche Aufwand. Ist die Entwicklung abgeschlossen, werden Fehlfunktionen oder auch Unerreichbarkeit der Anwendung meist durch die Anwender festgestellt. Es kommt zu Einschränkungen in den Betriebsabläufen und gegebenenfalls zu Verzögerungen bei den Produktionen. Der Systemservice und die Anwendungsbetreuung werden über fehlerhafte Funktionalität informiert. Die Anwendungsbetreuung reproduziert den Fehler, findet die Fehlerursache heraus und beseitigt diese.

Automatisierte Tests erleichtern die Entwicklung und Fehler in den Anwendungen werden im laufenden Betrieb automatisch erkannt und gemeldet. Die Entwicklungsabteilung kann frühzeitig auf Probleme reagieren, diese beheben und die Anwender informieren.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, zu zeigen, wie Webanwendungen während der Entwicklung und im laufenden Betrieb automatisiert getestet und überwacht werden können. Die Tests selbst umfassen Oberflächentests, welche die Funktionalität einer Anwendung prüfen. Auf Basis der erstellten Tests soll es möglich sein, weitere zu erstellen oder die bestehenden zu erweitern. Die automatisierten Tests werden mit Hilfe des Tools Ranorex erstellt. Die Überwachung und Steuerung der Tests erfolgt über das Tool Jenkins.

1.3 Aufbau der Arbeit

Die Arbeit ist in fünf Kapitel gliedert. Das Erste Kapitel beinhaltet eine Einleitung, die Motivation zu dieser Arbeit sowie die Zielsetzung. Durch diese Gliederung bekommt der Leser Informationen über das bestehende Problem. Das zweite Kapitel umfasst die Grundlagen und Tools, die für das Verständnis der Arbeit benötigt werden. Im dritten Kapitel wird das Testkonzept für die Webanwendungen beschrieben. Zu erst wird der Ist-Zustand in der Entwicklung im NDR beschrieben. Danach wird auf die Anforderungen und das Design für das Testkonzept eingegangen. Im Anschluss werden technischen Aspekte von Ranorex beschrieben. Im vierten Kapitel erfolgt die Beschreibung der Umsetzung des Testkonzeptes mit Ranorex und Jenkins. Im fünften Kapitel erfolgt eine Evaluierung und das Fazit der Arbeit. Im sechsten Kapitel erfolgt eine Zusammenfassung und ein Ausblick auf mögliche Erweiterungen der Arbeit.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für die Arbeit vorgestellt. Im ersten Abschnitt werden die Grundbegriffe für das Testen definiert, die im Rahmen dieser Arbeit zum Einsatz kommen und Hintergründe erläutert. Im zweiten Abschnitt werden die Technologien Jenkins, Jira, Git, Bitbucket und Ranorex vorgestellt, wenn sie diese bereits kennen, können Sie im Kapitel drei weiterlesen.

2.1 Testgrundlagen

In diesem Abschnitt werden wichtige Begriffe für die Arbeit aus dem Bereich des Testens vorgestellt. Die Definition der Grundbegriffe aus dem Bereich des Testens orientieren sich an dem International Software Testing Qualifications Board Standard (IST QB-Standard) und den Folien zur Vorlesung Certified Tester. [4] [6]

2.1.1 Testprozess

Ziel des Testprozesses ist es Tests in Softwareprojekten automatisiert und strukturiert durchzuführen. Ein Testprozess besteht aus einer Menge von Testaktivitäten, die sich je nach Situation des Projektes unterscheiden. Diese sind in der Teststrategie des Projektes oder Unternehmens festgelegt. Der Testprozess umfassen im Normalfall die nachfolgenden Aktivitäten. [4, Kapitel 2.3]
In der *Testplanung* wird ein Testkonzept auf Basis der Teststrategie erstellt. Das Testkonzept ist die Beschreibung des Testprozesses, mit der Vorgehensweise, den benötigten Ressourcen und der Zeitplanung.
Bei der *Testüberwachung und Steuerung* wird die aktuell ausgeführte Testaktivität beobachtet und Abweichungen zur Planung berichtet. Auf Grundlage der Situation wird die Planung angepasst, um die Ziele zu erreichen.

Die *Testanalyse* umfasst die Untersuchung der Testbasis (alle Informationen und Dokumente), wozu die Prüfung der Testbasis, die Analyse von Dokumenten, die Testbarkeit des Testobjektes und Berichte zur Risikoanalyse in Betracht gezogen werden.
Der *Testentwurf* legt fest, wie Tests durchgeführt werden. Die Spezifikation erfolgt abstrakt und konkret. Abstrakte Testfälle enthalten keine konkreten Ein- und Ausgabewerte, sondern logische Operatoren für die Beschreibung. Die Testfälle können für mehrere Testzyklen mit unterschiedlichen konkreten Werten (Eingaben) verwendet werden.
Die *Testrealisierung* ist die Vorbereitung für die Testdurchführung. Die Infrastruktur mit dem Testrahmen und der Testumgebung wird überprüft und bereitgestellt. Die Ausführungsreihenfolge der Testfälle mit den entsprechenden Testdaten wird festgelegt und vorbereitet.
Die *Testdurchführung* wird akkurat und komplett protokolliert. Das Protokoll dient als Nachweis für die geplante Teststrategie und muss für den Kunden nachvollziehbar sein. Die Durchführung erfolgt manuell oder mit Werkzeugen. Unter gleichen Bedingungen soll sich der Test zu einem späteren Zeitpunkt wiederholen beziehungsweise reproduzieren lassen. Abweichungen zu den Zielen werden ebenfalls im Protokoll festgehalten.
Der *Testabschluss* erfolgt je nach Entwicklungsmodell zu einem anderen Zeitpunkt, wie zum Beispiel beim Beenden des Testobjektes oder beim Abschluss einer Testaktivität auf einer Teststufe. Alle Testergebnisse und -aktivitäten werden in einem Testabschlussbericht zusammengefasst. Auf Basis der Testergebnisse erfolgt eine Bewertung gegenüber den festgelegten Endkriterien im Testbericht. Zur Wiederholung der Test werden die Testumgebung, Testdaten, Testinfrastruktur und Testmittel archiviert. Den Stakeholdern (Anwendungsbetrieb) werden der Testbericht und die Testmittel übergeben. Eine Verbesserung des Testprozesses wird durch die Umsetzung der Erkenntnisse erreicht.

Im Testprozess ist es wichtig, dass zwischen der Testbasis und den Testergebnissen eine Rückverfolgbarkeit besteht. Die Überwachung und Steuerung sind durch die Analyse der Auswirkungen und Änderungen auf den Prozess effektiver. Die Berichte zum Fortschritt und Abschluss sind anschaulicher und nachvollziehbarer. Die Testüberdeckung ist bewertbar. Technische Informationen werden für die Beurteilung des Testobjektes gegenüber den Zielen des Geschäftes aufbereitet.

2.1.2 Softwareentwicklungsmodelle

Die Softwareentwicklung erfolgt oft auf Basis von Modellen. In diesen wird die Software in Abschnitte, Phasen oder Iterationen aufgeteilt. Die Tests unterscheiden sich in Bedeu-

tung und Umfang. Die einzelnen Aufgaben werden in einigen der Modelle bestimmten Rollen zugeordnet. Die Modelle teilen sich in sequenzielle, iterative und inkrementelle Entwicklungsmodelle auf. [4, Kapitel 3.1]

2.1.2.1 Sequenzielle Entwicklungsmodelle

In sequenziellen Entwicklungsmodellen erfolgen die Abläufe der Aktivitäten linear und sequenziell. Die Entwicklung aller Teile der Software erfolgt ohne Überlappungen in Entwicklungsphasen. Am Ende der Entwicklung wird ersichtlich, ob die fertige Software den Vorstellungen des Kunden einspricht. Die bekanntesten Vorgehensmodelle sind das Wasserfall- und V-Modell.
Im *Wasserfallmodell* muss vor dem Beginn eines neuen Entwicklungsschrittes der vorherige abgeschlossen und freigegeben werden. Wird die geforderte Qualität nicht erreicht, kann ein Schritt wiederholt werden. Das Testen und die Prüfung erfolgen erst im letzten Schritt des Modells der Endprüfung.
Das *V-Modell* ist eine Erweiterung des Wasserfallmodels. Zu jedem Entwicklungsschritt existiert eine Teststufe, in der das System auf Verifizierung und Validierung überprüft wird. Mit der *Verifizierung* wird auf die korrekte und vollständige Spezifikation geprüft. Die *Validierung* untersucht den Kontext ob die Software für den Einsatzzweck geeignet ist.

2.1.2.2 Iterative und inkrementelle Entwicklungsmodelle

In der *Iterativen Entwicklung* wird das Softwareprojekt in Zyklen mit festgelegten Schritten realisiert. In jedem Schritt ist eine Änderung vom Umfang oder den Merkmalen möglich um eine neue verbesserte Version zu erreichen.
Die *Inkrementelle Entwicklung* hat das Ziel ein neues Produkt schnell zur Verfügung zu stellen. In jeder neuen Versionen erfolgt die iterative Überarbeitung und Ergänzung.
In der Praxis existieren nur *Iterativ-inkrementelle Entwicklungsmodelle*. Hauptmerkmal ist, dass die Versionen Zwischenstände aufweisen, die in Form von Releases an die Kunden oder Anwender weitergegeben werden. Diese geben ein Feedback, wodurch Fehlentwicklungen schnell erkannt und behoben werden.
Die *Agile Softwareentwicklung* ist Teil der Iterativ-inkrementelle Entwicklungsmodelle. Ziel der agilen Entwicklung ist es, schnell funktionsfähige Software auszuliefern, um Kundenwünsche zu erfüllen. Die Interaktion mit dem Kunden steht im Vordergrund.

Auf Rückmeldungen, Anforderungen und Änderungen an der Software erfolgt eine flexible Reaktion und Handhabung. Die bekanntesten Agile Entwicklungsmodelle sind Scrum und Kanban. Beim Scrum Modell sind die Iterationen und Features klein. Im Kanban Model sind die Längen der Iterationen nicht festgelegt, wodurch in einem Release mehrere Gruppen von Features sein können.

2.1.3 Teststufen

Ein System besteht aus mehreren Teilsystemen und Komponenten. Diese bilden verschiedene Softwarearchitekturen. Die einzelnen Ebenen werden nach Eigenschaft und Verhalten durch Tests überprüft, welche als Teststufen bezeichnet werden. Die Teststufen teilen sich in einzelne Testprozesse auf. Diese prüfen schrittweise das gesamte System, um es abschließend frei zu geben. [4, Kapitel 3.4]

2.1.3.1 Komponententests

Komponententests, auch als Modul-, Klassen- oder Unit-Test bezeichnet, prüfen die kleinsten Softwareeinheiten eines Systems. Ein einzelner Softwarebaustein wird von den anderen Bausteinen des Systems getrennt betrachtet, um die inneren Aspekte ohne Wechselwirkungen zu ermitteln. Aufgabe der Komponententests ist es sicherzustellen, dass die Testobjekte die Funktionalität korrekt und vollständig nach Spezifikation erfüllen. Weitere Aspekte die ein Testobjekt erfüllen muss sind Robustheit, Effizienz und Wartbarkeit. Die Prüfung auf Robustheit erfolgt über unzulässige und unerwartete Eingaben oder Aufrufe. Die Effizienz wird in Bezug zur Wirtschaftlichkeit des Testobjekts und seiner Ressourcennutzung überprüft. Die Wartbarkeit prüft Eigenschaften wie die Codestruktur, Modularität, Kommentare und Verständlichkeit. Komponententests werden meist durch die entwickelnde Person in der Entwicklung durchgeführt. Diese hat die nötige Qualifikation in der Programmierung und in den notwendigen Werkzeugen.

2.1.3.2 Integrationstests

Die nachfolgende Stufe wird als Integrationstest bezeichnet. Es wird das Zusammenspiel von Komponentengruppen in Bezug auf den technischen Entwurf betrachtet. Die Komponenten des Systems werden schrittweise zusammenfasst und die entstehenden Schnittstellen überprüft. Im Anschluss erfolgt die Prüfung zu den Schnittstellen von anderen

Systemen und oder zwischen Hardware und Software. Ziel des Integrationstestes ist es mögliche Schnittstellenfehler aufzudecken.

2.1.3.3 Systemtests

Die dritte Stufe wird als Systemtest bezeichnet. In diesem wird das System auf spezifizierte Anwendungen an das Produkt hin überprüft. Das Testsystem sollte möglichst identisch zur Produktivumgebung sein, um das System als Ganzes betrachten zu können. Bei Systemen, die auf Datenbanken basieren, müssen die Daten eine gewisse Qualität in Bezug auf die Konsistenz, Vollständigkeit und Aktualität aufweisen, um als Testobjekt zu dienen. Ziel der Systemtests ist es die gestellten Anforderungen, welche funktional oder nicht-funktional sein können, zu prüfen. Weiter sollen Mängel und Fehler im System identifiziert werden, die durch unvollständige, widersprüchliche, vergessene und undokumentierte Anforderungen entstehen.

2.1.3.4 Abnahmetests

Abnahmetests prüfen das System aus Kunden- und Nutzersicht in Bezug auf die vereinbarten Leistungsmerkmale. Diese Tests werden auch als Akzeptanztests bezeichnet und gliedern sich in Feldtest, Vertragliche-, Benutzer-, Systembetreiber-Akzeptanz. Beim Test auf Vertragliche-Akzeptanz führt der Kunde auf Basis eines Vertrages die Abnahme durch. Test auf Benutzer-Akzeptanz, auch als Benutzerabnahmetests bezeichnet, kommen bei verschieden Anwendergruppen mit unterschiedlichen Erwartungen zum Einsatz. Akzeptanztests für den Systembetreiber stellen sicher, dass sich das System aus Sicht der Systemadministratoren in die bisherige IT-Umgebung einfügt. Der Feldtest kommt zum Einsatz, wenn keine Testumgebung möglich ist. Auf Basis von vorläufigen Softwareversionen erfolgen Testszenarien oder Tests unter realistischen Bedingungen beim Hersteller (Alpha-Test) oder beim Kunden (Beta-Test). Auf Basis dieses Tests führt der Hersteller Anpassungen durch.

2.1.4 Testarten

Um die Ziele und den Fokus der Teststufen zu bestimmen, existieren verschiedene Testarten. Diese teilen sich in vier grundlegende Testarten auf und kommen in allen Teststufen zum Einsatz. [4, Kapitel 3.5]

2.1.4.1 Funktionale Tests

Mit funktionalen Tests wird ein Testobjekt auf funktionale Anforderungen nach Spezifikation überprüft. Funktionale Anforderungen beschreiben und spezifizieren wie sich das System oder ein Systemteil verhält. Auf Basis der erfüllten Anforderungen wird entschieden, ob ein System produktiv und einsatzbereit ist. Es sind meist mehrere Testfälle für das Testen einer funktionalen Anforderung notwendig.

2.1.4.2 Nicht funktionale Tests

Nicht funktionale Tests prüfen die Qualität des Systems aus verschiedenen Sichten, messen das Verhalten der Software und kommen meist in Systemtest zur Anwendung. Die Qualität der Software beeinflusst maßgeblich die Zufriedenheit des Kunden. Nicht funktionale Tests gliedern sich nach den Merkmalen des Qualitätsmodells IO 25010 siehe Tabelle 2.1 .

Qualitätsmerkmal	**Test**
Performanz & Effizienz Die Performanz beschreibt die Zeit, Ressourcen und Kapazität die ein System oder eine Komponente für die Ausführung von Funktionen benötigt. Die Effizienz vergleicht die Mittel zu den erreichten Zielen.	- Performanztests messen die Antwortzeit in verschiedenen Anwendungsfällen. - Lasttests messen das Verhalten des Systems unter wechselnder Systemlast. - Volumentest prüfen das Systemverhalten in Abhängigkeit zur Datenmenge. - Stresstests beobachten das Systemverhalten bei Überlastung.
Kompatibilität Testen des Zusammenwirkens mit anderen Systemen.	- Konfigurationstests prüfen die unterschiedliche Konfigurationen eines Systems. - Interoperabilitätstests prüfen die Interaktion eines Systems mit Standardsoftware (COTS commercial off-the-shelf).
Benutzbarkeit Festlegen von Zielen im Kontext von Nutzern.	- Gebrauchstauglichkeitstests prüfen die Bedienbarkeit und Verständlichkeit der Systemausgaben in Bezug zu einer Anwendergruppe.

Zuverlässigkeit Ermittlung von Ausfällen pro Betriebsstunde im Dauerbetrieb in einem Profil.	- Robustheitstests prüfen gegenüber Fehlbedienung, Fehlprogrammierung und Hardwareausfall, sowie die Fehlerbehandlung beim Wiederanlauf.
Sicherheit Erfolgt der Zugriff auf Daten und Ressourcen autorisiert.	- Sicherheitstests prüfen gegen unberechtigtem Systemzugang oder Datenzugriff.
Wartbarkeit Aufwand der zur Durchführung von Änderungen nötig ist.	- Wartbarkeitstests prüfen auf Verständlichkeit z.B. der Entwicklungsdokumente oder Systemstruktur. - Statische Analysen erfolgen durch automatische Prüfung gegen Regeln.
Portabilität Übertragbarkeit eines Systems auf andere Umgebungen.	- Portabilitätstests prüfen, ob die Software auf anderen Plattformen und Betriebssystemen downloadbar, installierbar und deinstallierbar sind. - Installationstests

Tabelle 2.1: nicht funktionale Qualitätsmerkmale

2.1.4.3 Strukturbezogene Tests

Strukturbezogene Tests basieren hauptsächlich auf den Whitebox-Verfahren und kommen vorwiegend bei Komponenten- und Integrationstest zum Einsatz. Die innere Architektur der Software wird betrachtet, um die Abdeckung aller Elemente des Quellcodes zu erreichen.

2.1.4.4 Anforderungsbezogene Tests

Anforderungsbezogene Tests basieren auf dem Blackbox-Verfahren und kommen hauptsächlich in System- und Abnahmetests zum Einsatz. Als Testbasis dienen Spezifikationen, wie Use Cases oder Use Stories, welche das Verhalten der Software aus externer Sicht beschreiben.

2.1.5 Wartungs- und Änderungstests

Wartungs- und Änderungstests kommen bei einer Änderung oder Weitereinwicklung zum Einsatz. [4, Kapitel 3.6]

2.1.5.1 Softwarewartungstests und -pflegetests

Bei der *Softwarewartung* werden lange bekannte Fehlerzustände in einem Produkt beseitigt und anschließend mit der Strategie Fehlernachtest überprüft. *Fehlernachtests* prüfen das System auf einen bestimmten Fehlerzustand hin. Für die eigentlichen Änderungen sind weitere Testfälle notwendig.
In der *Softwarepflege* erfolgt die Anpassung auf neue Einsatzbedingungen. Bestehende Datenbestände müssen migriert und konvertiert werden. Das Testen auf Akzeptanz vom Systembetreiber und auf nicht funktionale Anforderungen ist notwendig, um das Verhalten an die neue Umgebung zu überprüfen. Im Anschluss erfolgen Fehlernachtests und Testfälle auf Änderung.

2.1.5.2 Weiterentwicklungstests

Bei einer Weiterentwicklung oder Änderungsarbeiten werden neue und alte Funktionen überprüft. Folgt stattdessen eine Stilllegung der Software, sind die Datenbestände zu archivieren oder zu übernehmen.

2.1.5.3 Regressionstests

Regressionstests testen das System nach einer Modifikation auf unbeabsichtigte Seiteneffekte und neue Fehlerzustände. Um sicherzustellen, dass die Modifikationen kein anderes Verhalten gegenüber der vorherigen Version aufweist, werden die bestehenden Tests erneut ausgeführt. Der Umfang dieser Tests hängt von der Testautomatisierung, den Kosten und der Zeit ab. Erfolgen die Tests manuell, werden Tests mit hoher Priorität, bestimmten Konfigurationen, Teststufen oder Teilsystemen bevorzugt durchgeführt.

2.1.6 Statische Tests

Bei statischen Tests wird gegen Regeln ohne die Ausführung des Prüfobjektes geprüft. Eine Prüftechnik ist die statische Analyse, diese erfolgt mit Werkzeugen und Dokumenten mit formaler Struktur. Die zweite Prüftechnik ist das Review, in diesem erfolgt die Prüfung manuell durch Personen. [4, Kapitel 4]

2.1.6.1 Review

Das Review dient zur Kontrolle und Qualitätsüberwachung und kann informell oder formell erfolgen. Beim formellen Review ist im Gegensatz zum informellen Review der Prozess definiert, das Ergebnis und die beteiligten Personen festgelegt. Der Prozess von Reviews besteht aus der Planung, der Initiierung, dem individuellen Review, der Diskussion der Befunde, dem Bericht und der Fehlerbehebung. Die beteiligten Personen können die folgenden Rollen einnehmen: Manager, Leiter, Moderator, Autor, Gutachter (Reviewer) und Protokollant.
Reviews werden in die Arten informelles Review, Walkthrough, technisches Review und Inspektion aufgeteilt. Die Auswahl richtet sich nach dem Ergebnis, den verfügbaren Ressourcen, den möglichen Risiken, dem Geschäftsbereich und der Unternehmensstruktur. Ein mögliches Ergebnis ist, dass sich das Review auf Dokumente, die während des Entwicklungsprozesses erstellt werden, bezieht. Bei dem zweiten Ergebnis analysiert das Review den Projektablauf oder Entwicklungsprozess selbst.
Im informellen Review wird dem Autor ein Feedback mit potenziellen Fehlern gegeben. Das Walkthrough ist eine informelle Vorgehensweise für kleine Teams, in dem mit einem Gutachten gearbeitet wird und die Ergebnisse dokumentiert werden. Beim technischen Review sind alle Teilnehmer vorbereitet und Experten. Der Ablauf ist festgelegt, im Protokoll steht priorisiert die Bewertung der Qualität, Spezifikation und Eignung für den Einsatz. Die Inspektion ist formal, die Rollen vorgeschrieben, der Ablauf und die zu Prüfenden Schritte sind definiert. Ziele sind die Qualität zu kontrollieren, das Ergebnis zu verbessern und zu optimieren. [4, Kapitel 4]

2.1.7 Dynamische Tests

Das Testobjekt wird bei dynamischen Tests während oder durch Ausführung auf Funktionalität hin überprüft. Ziel ist es nachzuweisen, dass festgelegte Anforderungen erfüllt

werden. Ein weiteres Ziel sind Abweichungen und Fehlerwirkungen aufzudecken. Zum Erreichen der Ziele erfolgt die Testerstellung in Schritten. Zu Beginn werden Bedingungen und Voraussetzungen festgelegt. Im Anschluss werden die Testfälle spezifiziert und zum Schluss die Testdurchführung festgelegt. Systematische Testfälle werden mit dem Whitebox- oder Blackbox-Verfahren erstellt. Das Whitebox-Verfahren analysiert und prüft die innere Struktur und Abläufe des Testobjekts. Beim Blackbox-Verfahren wird das Testobjekt als schwarzer Kasten angesehen und das Verhalten von außen betrachtet. Für beide Verfahren ist eine Testbasis erforderlich. Whitebox-Verfahren kommen vorwiegend für die Teststufen der Komponenten- und Integrationstest zum Einsatz, diese beziehen sich meist auf den Programmtext. Existiert keine oder nur eine mangelhafte Testbasis, so erfolgt die Erstellung der Tests durch die erfahrungsbasierte Testfallermittlung. [4, Kapitel 5]

2.1.7.1 Blackbox-Testverfahren

Blockbox-Testverfahren können auf funktionale und nicht funktionale Tests angewendet werden. Der innere Aufbau des Testobjekts ist nicht bekannt, weshalb sich die Testfälle auf die Ein- und Ausgaben des Testobjektes beziehen. Die Testfälle selbst werden aus Spezifikationen abgeleitet oder sind Teil dieser. Nun erfolgt eine kurze Beschreibung der Blackbox-Testverfahren. [4, Kapitel 5.1]
Bei der *Äquivalenzklassenbildung* wird ein Testobjekt mit vielen unterschiedlichen un/-gültigen Ein- und Ausgabebedingungen (Wertebereichen) auf definierte Ausgaben hin überprüft.
Die *Grenzwertanalyse* baut auf der Äquivalenzklassenbildung auf und untersucht Wertebereichsgrenzen.
Der *zustandsbasierte Test* prüft Zustandsübergänge bei komplexen Zuständen. Beim Systemtest einer grafischen Bedienoberfläche können die Navigationsmöglichkeiten mit Zustandsautomaten realisiert werden.
Beim *Entscheidungstabellentest* wird das Verhalten eines Testobjektes abhängig von Kombinationen von Bedingungen in einer Tabelle mit Regeln, logischen Bedingungen und Aktionen zugeordnet.
Beim *anwendungsfallbasierten Test* wird ein Anwendungsfall (Use Case) in einem bestimmten Szenario betrachtet. In diesem führen Vorgänge in einem Dialog zwischen Akteur und Testobjekt zu einem eindeutigen Ergebnis. Durch die Außensicht sind Use Cases gut für System- und Akzeptanztest geeignet.

Im *Syntaxtest* basieren die Testfälle auf formalen Spezifikationen der Syntax für Eingangsdaten.
Beim *Zufallstest* werden die Testfälle aus einem Nutzungsprofil mit pseudozufälligem Algorithmus ausgewählt.
Der *Smoke-Test* beinhaltet eine Teilmenge aller Testfälle, welche die Hauptfunktionalität (wichtigsten Funktionen) des Testobjektes umfassen und ausführen.

2.1.7.2 Erfahrungsbasierte Testverfahren

Die erfahrungsbasierte Testfallermittlung ergänzt die systematischen Testverfahren durch die Aufdeckung von Fehlerwirkungen. Die Testfälle werden auf Basis von Erfahrung, Wissen und Intuition der Tester abgeleitet und in drei Verfahren aufgeteilt. [4, Kapitel 5.3]
Die *induktive Testfallermittlung* basiert neben der Erfahrung und den intuitiven Fähigkeiten auf einem Fehlerkatalog. Der Fehlerkatalog enthält mögliche Fehlerhandlungen, Fehlerzustände und Fehlerwirkungen.
Eine Checkliste dient dem *checklistenbasierten Testen* als Grundlage für die Testfälle, welche immer aktuell sein sollte und komplett abgearbeitet werden muss.
Das *explorative Testen* ist informell und es können mangelhafte oder keine Spezifikation vorhanden sein. Die Testaktivitäten Testfallanalyse, Testentwurf, Testrealisierung und Testdurchführung erfolgen praktisch parallel und nicht wie beim Testprozess strukturiert. Mögliche Aufgaben und Funktionen des Testobjektes werden ermittelt und Charta festlegt. Ein Charta ist ein Testziel oder eine Testidee die getestet werden soll. In einem Test werden die Charta überprüft und anschließend analysiert. Auf Basis der gesammelten Informationen und Auffälligkeiten werden neue Testfälle erstellt.

2.1.8 Teststrategie

In der Teststrategie werden die Ziele und Rahmenbedingungen eines Projektes festgelegt. Bei der Strategiefindung gibt es unterschiedliche Herangehensweisen.

Im Entscheidungs- und Gestaltungsspielraum wird der Zeitpunkt spezifiziert, in dem der Tester in ein Projekt involviert wird, um die Tests durchzuführen. Diese teilt sich in zwei Situationen auf.
Beim *Vorbeugenden Ansatz* wird der Tester ab Projektbeginn in das Projekt involviert. Der Tester hat dadurch die Möglichkeit das Projekt mit zu gestalten, zu optimieren,

einzugreifen und Maßnahmen bei Fehlern umsetzen. Dieser Ansatz ist vor allem bei sicherheitskritischen Systemen notwendig.
Beim *Reaktiven Ansatz* bekommt der Tester die Software oder das System eines Projektes nach Fertigstellung. Der Tester muss sich der Situation anpassen und die Tests planen, entwerfen, erstellen und durchführen. In der Praxis für die Umsetzung dieser Teststrategie wird oft die Vorgenweise des ‘explorativen Testens‘ verwendet.

In der zweiten Herangehensweise, gibt es zwei Ansätze, welche die Verfügbarkeit an Wissen über ein Projekt betrachten.
Der *Analytischen Ansatz* basiert auf Daten und deren systematische Analyse.
Der *Heuristische Ansatz* basiert auf Erfahrungswissen oder Faustregeln. Die Faustregeln werden verwendet wenn keine Daten vorliegen oder das Know-how fehlt.

2.2 Verwendete Tools

Die beschriebenen Technologien und Tools dienen in der Arbeit zur Erklärung und werden für die Umsetzung benötigt.

2.2.1 Git

Git ist ein Open-Source-Versionsverwaltungssystem und der moderne Standard für Softwareentwicklung. Versionsverwaltungssysteme teilen sich in zentrale und verteilte Systeme auf. Ein verteiltes Versionsverwaltungssystem bietet im Gegensatz zum zentralen System die Möglichkeit Änderungen ohne eine Server-Verbindung lokal durchzuführen. Es kann parallel an den gleichen Dateien gearbeitet werden und im Repository ist der gesamte Verlauf enthalten.
Die Versions- und Entwicklungskontrolle erlaubt eine stetige Protokollierung von Änderungen der Dateien eines Softwareprojektes. Das Verzeichnis für Dateien wird auch als Repository bezeichnet. Das Repository kann im Verlauf über Branches verzweigt, zusammengeführt und umgearbeitet werden. Die Entwickler haben lokal auf ihrem Rechner den Bearbeitungsverlauf eines Git-Repositories. In Pull-Requests werden Branches zusammengeführt, wodurch ein gemeinsames Arbeiten in einem Repository möglich ist. [7]

2.2.2 Bitbucket

Bitbucket ist ein webbasiertes Tool zur Versionsverwaltung von Softwareprojekten auf Basis von Git. Es bietet Teams einen Ort, um Projekte zu planen, testen, verteilen und am Code arbeiten zu können. Tools wie das Projektorganisationstool Jira können in Bitbucket integiert werden. Die Prüfung von Code (Code-Reviews) kann während der Beurteilung von Pull-Requests erfolgen. Das Mergen in den produktiven Zweig ist in Checklisten mit festgelegten Gremien (Personen) möglich. CI/DC (Continuous Integration/Continuous Delivery) ermöglicht in Pipelines das Projekt automatisch zu bauen, zu testen und anschließend zu installieren (deployen). [5]

2.2.3 Jira

Jira ist eine Software zum Planen, Verfolgen und Freigeben von Projekten. Ein Projektteam kann eigene oder vorgefertigte Arbeitsabläufe (Workflows) verwenden. Die Planung erfolgt in User-Storys, Vorgängen und Sprints. Die einzelnen Aufgaben werden Projektmitgliedern zugeordnet. In Boards, wie Kanban, ist eine Übersicht über die Schritte im Projekt ersichtlich. In Veröffentlichungen (Releases) wird der aktuelle Stand einzelner Arbeitsschritte und des Projektes angezeigt und aktualisiert. Es ist eine Integration mit anderen Tools, wie Bitbucket, möglich. Diese Integration erlaubt eine automatische Aktualisierung von Vorgängen und Veränderung von Aufgaben. [11]

2.2.4 Jenkins

Jenkins ist ein Open-Source-Automatisierungsserver in dem Plugins installiert werden können, um Funktionalitäten zu erweitern, hinzuzufügen und zu integrieren. Die Plugins ermöglichen es Projekte individuell zu erstellen, anzupassen, bereitzustellen, zu testen und zu automatisieren. Über eine Weboberfläche erfolgt die Konfiguration und Einrichtung von Jenkins. In dieser wird der Status von Projekten angezeigt, welcher eine Übersicht über alle erfolgreichen und nicht erfolgreichen Tests (Error Checks) mit dem zeitlichem Verlauf enthält. Sich wiederholende Aufgaben, wie automatisierte Tests, können automatisiert und überwacht werden. Wird ein Problem oder Fehler festgestellt, können die Verantwortlichen benachrichtigt werden. [8]

2.2.5 Ranorex

Ranorex ist ein Windows-Server-Tool welches End-to-End-Tests für Desktop-, Web- und mobile Apps anbietet. Die Tests können anschließend lokal oder fern (remote) auf Geräten oder in einer Simulation ausgeführt werden. In Browsern wie Chrome, Firefox und Microsoft Edge sind parallele Tests möglich. In der IDE Ranorex Studio werden die Testfälle entwickelt und Reports angezeigt. Die Reports sind konfigurierbar und lassen sich direkt als PDF-Datei anzeigen oder automatisch als E-Mail versenden. In Video-Reports kann der Zugriff direkt auf einen fehlerhaften Testabschnitt aus einem Testreport erfolgen, um eine Korrektur des Tests im Wartungsmodus durchzuführen oder um das Debugging zu erleichtern. Weiter ist es möglich JUnit-kompatible Reports zu generieren, um diese in andere Tools zu integrieren. Die Testfälle können durch Programmieren oder Aufnahmen von Testaktionen erstellt werden. Aus den Testfällen generiert Ranorex Automatisierungsskripts die effizient, wartungsfreundlich und modular sind. Ranorex bietet durch die einfache Erstellung von Tests schnelle Automatisierungsziele und Zeit für die Verbesserung der Qualität von einem Produkt. [13]

3 Testkonzept

In diesem Kapitel wird ein Testkonzept für automatisierte Tests von internen Webanwendung erstellt. Zu Beginn werden die bisherigen Zuständigkeiten, Prozesse und verwendeten Tools bei der Entwicklung im NDR erläutert. Im Anschluss werden die Anforderungen an das Testkonzept benannt, das Design festgelegt und technischen Aspekte von Ranorex beschrieben.

3.1 Ist-Zustand bei der Entwicklung im NDR

In der Fachgruppe IT-Service im NDR liegt die Verantwortung für PC-Arbeitsplatzsysteme und dezentrale Server in Hamburg. Zum Aufgabengebiet gehört die Betreuung der Hardware und Software des NDR-Standard-Clients mit dazugehörigen Managementsystemen. Weitere Aufgaben sind die Anwendungsbetreuung und die Service-Verantwortung für betriebswirtschaftliche administrative Systeme im NDR, wie zum Beispiel SAP. Die IT-Service Dienstleistungen werden über zwei Competence Center ‘CC Client‘ und ‘CC Anwendungen‘ erbracht.
Das Center ‘CC Client‘ verantwortet die PC-Arbeitsplatzsysteme, die Entwicklung und Administration des NDR Standardclients, Windows Administration, Anwendungsbetreuung, Betreuung dezentraler Server, das Patch-Management von Software und Arbeitsplatzsystemen, Softwaretests und die Betreuung des Testcenters.
Das Center ‘CC Anwendungen‘ betreut und entwickelt Anwendungen. Für die Software SAP übernimmt es den Anwendungsbetrieb, sowie das Benutzer- und Berechtigungsmanagement.

Das Competence Center ‘CC Anwendungen‘ nutzt für die Koordinierung der Arbeitsschritte in der Entwicklung und Betreuung von Anwendungen das Managementtool Jira. In diesem werden die Arbeitsschritte in Vorgängen beschrieben und einer Person für die

Bearbeitung zugewiesen. Für die Übersicht sorgt ein Kanbanboard, dieses zeigt alle Vorgänge zu einem Projekt an und in welchem Entwicklungsschritt sich dieses gerade befindet. Die Abbildung 3.1 gibt eine Übersicht über die derzeitigen Entwicklungsschritte. Zu jedem Vorgang wird in der Übersicht der Projektsitzungsschlüssel, die Beschreibung des Vorganges, das Datum der letzten Bearbeitung, die Abteilung der bearbeitenden Person und die Priorität angezeigt. Die Entwicklungsschritte in dem sich ein Vorgang befindet werden nun einzeln erläutert. 'Neu' steht für einen neu erstellter Vorgang. 'Freigegeben/Wartend' für einen zugeordneten in Bearbeitung befindlichen Vorgang. 'Freigabe für internen Test' besagt, dass ein Vorgang für einen Test in einer Testumgebung bereit ist. 'interner Test erfolgreich' zeigt, dass ein Vorgang in einer Testumgebung erfolgreich getestet wurde. 'Freigabe für Abnahmetest' der Vorgang kann von Kunden angenommen

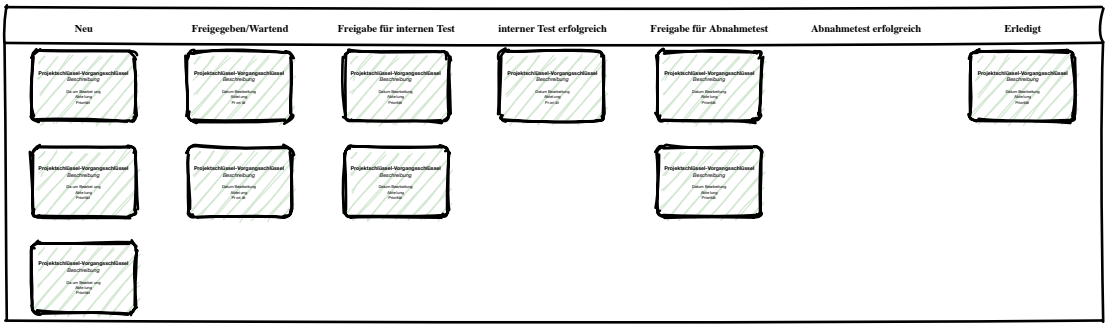


Abbildung 3.1: Kanbanboard

Webanwendungen in NDR

Im NDR werden viele betriebsinterne Abläufe über Webanwendungen verwaltet. Der Zugriff auf diese besteht nur im Intranet des NDR. In dieser Arbeit erfolgen die automatisierten Tests und Beschreibungen für der Webanwendung 'NDR Yii_basic'. Diese dient im NDR als Basisanwendung für alle Webanwendungen und beinhaltet ein vorgegeben Webseiten-Aufbau mit grundlegenden Funktionen. An einem Beispielobjekt werden aus der Benutzeransicht die grundlegenden Funktionen gezeigt. Über die Navigationsbar kann ein neues Beispielobjekt angelegt oder die Übersicht über alle Beispielobjekte angezeigt werden. In der Übersicht können angelegte Beispielobjekte, angezeigt, bearbeitet oder gelöscht werden. Für den Administrator existiert ein zusätzlicher Bereich auf den im Rahmen dieser Arbeit nicht eingegangen wird.

Entwicklungsschritte bei Webanwendungen

Wie beim Wasserfallmodel erfolgt die Erweiterung oder Neuentwicklung einer Anwendung in festgelegten Schritten nacheinander, siehe Tabelle 3.1. Jedem Entwicklungsschritt sind eine oder mehrere Rollen zugeordnet. Haben Kunden eine Idee oder einen Projektvorschlag, so teilen sie diese dem Anforderungsmanagement mit. Es erfolgt eine Aufnahme der Anforderungen und eine Prüfung einer möglichen Umsetzung. Existiert bereits eine Software, welche die Anforderungen abdecken könnte, wird eine Integration in diese geprüft und ggf. realisiert. Bei einer Neuentwicklung erfolgt eine Analyse von der Projektsteuerung (Projektleitung) in der Entwicklungsabteilung, die eine Kosten- und Aufwandsschätzung durchführt. Im Anschluss werden die Spezifikationen zwischen dem Kunden und der Projektleitung im Entwurf festgelegt. Das Design bzw. die Gestaltung der Webanwendungen ist über die Webanwendung 'NDR Yii_basic' vorgegeben. Nach der Realisierung, welche die Entwicklung oder Erweiterung beinhaltet, kommt die Prüfung. Die Prüfung wird im Moment manuell auf den Entwicklungsrechnern und in der Testumgebung durchgeführt. Die Abnahme der Anwendung erfolgt durch den Kunden ebenfalls auf der Testumgebung. Ist die Abnahme erfolgreich wird die Webanwendung in die Produktivumgebung übernommen. Der Anwendungsbetrieb übernimmt die Wartung und den Support.

Entwicklungsschritte	**Rollen**				
	Kunden	Anforderungsmanagement	Entwicklungsprojektleitung	Entwicklung	Anwendungsbetrieb
Idee / Projektvorschlag	X				
Anforderungen		X			
Analyse & Entwurf	X		X		
Realisierung				X	
Prüfung			X	X	
Abnahme & Auslieferung	X				X
Wartung & Support					X

Tabelle 3.1: Entwicklungsschritte bei Webanwendungen

Deployment-Chain

Die Deployment-Chain beschreibt den Ablauf von einer Entwicklung, über das Testen und Ausführen einer Webanwendung mit Bitbucket und anderen Systemen. Die Abbildung 3.2 zeigt die Deployment-Chain einer Webanwendung. Für alle Anwendungen stehen eine Test- und eine Produktivumgebung auf virtuellen Linux Servern zur Verfügung. Die Bereitstellung der Webseiten, den eigentlichen Anwendungen, erfolgt über Apache-Webserver. Die persistenten Daten der Anwendungen liegen zumeist in einer Datenbank, die auf MariaDB basiert. Über verschiedene Hostnamen auf dem Apache-Webserver werden unterschiedliche Webseiten zur Verfügung gestellt. Virtuelle Hosts, auch als Virtual Hosting oder VHost bezeichnet [2], die namenbasiert sind, werden mehrere Domains einer IP-Adresse zugeordnet [1]. Der DNS Server sorgt für die Zuordnung der Hostnamen und der Apache HTTP Server erkennt diese.

Die Verwaltung des Codes für die Anwendungen erfolgt über Git und den Onlinedienst Bitbucket. Bitbucket hat für jede Anwendungen verschiedene Repositories mit jeweils drei Braches. Der Masterbranch ist der Entwicklungs-Branch, der Testbranch ist für die Testumgebung und der Prodbranch für die Produktivumgebung.

Im Arbeitsverzeichnis des Entwicklungsrechners, auch als Sandbox [15] bezeichnet, erfolgt die Modifikation des Codes. Der Code wird in dem Webframework Yii [17] realisiert, welches auf der Programmiersprache PHP basiert. Für die lokale Entwicklung wird das Tool XAMPP [16] installiert, welches einen lokalen Apache-Webserver, die Programmiersprache php und die Datenbank MariaDB bereitstellt. Im Browser kann lokal auf die Webseite und die Datenbank zugegriffen werden, um die lokalen Tests durchzuführen.

Änderungen aus dem Arbeitsverzeichnis werden über den Befehl `git add` hinzugefügt. Die Übername der gesammelten Änderungen erfolgt über einen Commit. In der Nachricht (CommitMassage) werden die Änderungen beschrieben. Über den Befehl `git push` werden die Referenzen und Commits aus dem Arbeitsverzeichnis in den Master-Branch hochgeladen.

Ist ein Teil-Feature der Anwendung fertig wird vom Entwickler ein Pull-Request erstellt und automatisch ein anderes Mitglied der Entwicklungsabteilung per Mail informiert. Dieser führt ein Code-Review durch und kann über Feedback einzelne Zeilen oder Abschnitte kommentieren und bearbeiten. Ist das Review abgeschlossen erfolgt eine Genehmigung (Approve) und ein Merge in den Testbranch oder eine Ablehnung (Reject) mit der Bitte um Nacharbeiten. [12]

Aus dem Test-Branch wird der Code über ein Skript automatisch in regelmäßigen Abständen auf die Testumgebung deployed. Nach jeder Feature-Entwicklung führt die Entwicklung einen internen Test durch. Ist die Entwicklung abgeschlossen, erfolgt durch die

Projektleitung ein Test und im Anschluss daran der Abnahmetest durch die Kunden. Nach der Abnahme wird der Code über einen Pull-Request vom Anwendungsbetrieb in den Prod-Branch übernommen und anschließend auf die Produktivumgebung deployed.

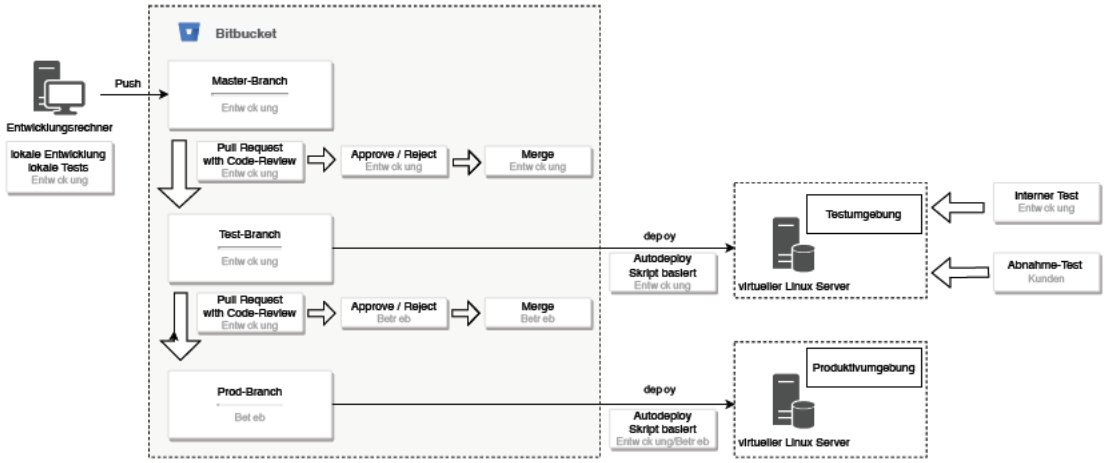


Abbildung 3.2: Ist Deployment-Chain

Verwendete Browser im NDR

Im NDR werden die Webanwendungen je nach Betriebssystem und Zugang in unterschiedlichen Browsern ausgeführt. Der Google Chrome Browser wird als Standard für Windows angesehen, auch die Browser Firefox und Edge (Chrome basiert) kommen oft zum Einsatz. Der Internet Explorer steht im Moment noch zur Verfügung und wird langfristig nicht mehr unterstützt. Im Betriebssystem macOS wird der Browser Safari verwendet.

Dokumentation der Webanwendung

Für die bestehenden Webanwendungen existieren nur geringe Dokumentationen oder Spezifikationen. Erfolgt nun eine neue Entwicklung oder Erweiterung dieser, werden die Anforderungen aufgenommen und umgesetzt. Die Anforderungen selbst liegen meist nur als Notizen vor, die sich an vorhandenen Webanwendungen orientieren.

Automatisierte Tests im NDR

Im NDR werden in Ranorex bereits Smoke-Tests für Anwendungen wie AdobeReader, Word, Excel und Outlook durchgeführt. Die Smoke-Tests beinhalten einfache Funktionen wie das An- und Abmelden, Anzeigen eines Willkommensbildschirms und Datenaktualisierung. Diese Tests werden durch Jenkins überwacht und gesteuert.

Im Rahmen dieser Arbeit sollen für Webwendungen automatische Tests in Ranorex erzeugt und mit Jenkins überwacht und gesteuert werden. Aus diesem Grund erfolgt nun eine Beschreibung, welche Lizenzen von Ranorex beim NDR bereits verfügbar sind und wie diese angewendet werden. Nachfolgend werden die grundlegenden Eigenschaften und Funktionen von Ranorex beschrieben, die bei der Umsetzung von automatischen Tests für Webanwendungen sinnvoll sind. Aus diesen werden die Anforderungen an das Testkonzept abgeleitet.
Für Jenkins, das beim NDR für die Überwachung und Steuerung der Tests eingesetzt wird, erfolgt eine Beschreibung. Diese umfasst wie Jenkins bisher im NDR eingesetzt wird und welche Möglichkeiten es bei der Umsetzung von automatischen Tests von Webanwendungen mit Ranorex gibt.
Das Testkonzept wird mit seinen Anforderungen auf Basis der Möglichkeiten, Funktionen und Eigenschaften die Ranorex und Jenkins zur Verfügung stellen, erstellt.

3.1.1 Ranorex Studio

Die Testautomatisierung erfolgt über das Tool Ranorex Studio. Für den Einsatz werden Lizenzen benötigt. Der NDR verfügt über mehrere Ranorex Enterprise- und Runtime-Lizenzen. Diese Lizenzen, auch als Mehrfachlizenzen (Premium Floating) bezeichnet, können auf mehreren physischen oder virtuellen Rechnern gleichzeitig verwendet werden. Ein Lizenzmanager überwacht alle genutzten Lizenzen und dient im lokalen Netzwerk als Lizenzserver. Ein Rechner oder Server stellt eine Anfrage an den Lizenzserver, sobald Ranorex Studio ausgeführt wird. Die Lizenzvergabe erfolgt bis keine Lizenzen mehr vorhanden sind. Mit Add-on Lizenzen (Runtime Floating Lizenz) werden die Enterpriselizenzen erweitert. Die Tests können dadurch auf mehreren Umgebungen parallel ausführt werden. Für das Testen unter realen Bedingungen wird auf Standardclients, die mit dem Betriebssystem Windows 10 ausgestattet sind, Ranorex installiert.

Es erfolgt nun die Beschreibung der Eigenschaften und Funktionen, die bei der Umsetzung von automatisierten Tests für Webanwendungen sinnvoll sind.

Interfaces und Schnittstellen werden für die Arbeit mit Software von Drittanbietern, Hardwaresystem oder verschiedenen Technologien in Ranorex benötigt. Über den Instrumentierungsassistent von Ranorex (Ranorex Instrumentation Wizard) ist es möglich in Technologien, wie dem Browser Chrome oder Adobe Flash ein Add-on zu installieren. Diese werden benötigt um die UI-Elemente in der Anwendung mit

Ranorex-Spy zu identifizieren.
Für die Verwaltung des Codes bietet Ranorex die Integration von zentralen oder verteilten Versionsverwaltungssystemen (VCS) an. Zentrale Versionsverwaltungssysteme die unterstützt werden sind Subversion und Team Foundation Version Control. Als verteiltes Verwaltungssystem steht Git zur Verfügung. Die Schnittstelle TortoiseGit in Ranorex ermöglicht es, den Dateistatus in Git über Icons anzuzeigen.
Ranorex ermöglicht die Integration mit Tools für Testmanagement (TM) und Continuous Integration (CI). Testmanagement- und Anwendungslebenszyklus-Tools (Application Lifecycle Management ALM) speichern für Tests Informationen zur Organisation, Implementierung und Ausführung, sowie eine Historie über ausgeführte Tests. TM- und ALM-Tools wie Jira lassen sich mit Ranorex verbinden. Erfolgt z.B. über Jira das Testmanagement, erlaubt diese Verbindung die Anzeige von Testberichten aus Ranorex in Jira. In das CI-Tool Jenkins lässt sich Ranorex über ein Plugin integrieren.

Code-Editor ermöglicht die Erstellung von flexiblen Skripten mithilfe der Programmiersprachen C# und VB.NET. Für die Entwicklung der Skripten besteht die Möglichkeit eine Entwicklungsumgebung wie Visual Studio, SpecFlow oder Microsoft TFS in Ranorex zu integrieren. Über eine Benutzercodebibliothek können Module und Skripten individuell verwaltet und bearbeitet werden. Dieses ermöglicht einen Zugriff auf Code von verschiedenen Testpersonen und das Einbinden von Code in andere Tests ohne viel Bearbeitung. Für die Erstellung von Skripten und der damit automatisierten Tests sind Programmierkenntnisse erforderlich. Features wie Codevervollständigung, Codevorlagen, Debugging und Refactoring Verfahren erleichtern die Realisierung.

Recorder ist ein Aufzeichnungstool in dem ohne Programmierkenntnisse Benutzereingaben in einem Arbeitsablauf aufgenommen und wiedergegeben werden können. Die Bearbeitung und Erweiterung der Aufzeichnung ist über den Recoder, manuell oder im Code C# möglich. Außerdem können Code-Skripts integriert werden.

Fokussierung wird über die Whitelist ermöglicht, in dem ein Fokus auf bestimmte Prozesse und Anwendungen während der Aufzeichnung mit dem Recorder gelegt wird. Ist diese Option nicht ausgewählt, werden alle Benutzeraktionen der laufenden Prozesse auf dem Rechner aufgezeichnet.

Konzepte in Ranorex gliedern sich in Desktop-, Web- oder Mobile Anwendungen oder einer Kombination aus diesen.

Solution ist die oberste Ebene in Ranorex Studio und enthält alle Testdateien, Projekte usw. Wird eine neue Solution erstellt, wird gleichzeitig ein Testsuite-Projekt erstellt welches direkt ausführbar ist.

Projekte teilen sich in Ranorex sich in die fünf verschiedene Projekttypen auf. In der Tabelle 3.2 sind die Projekttypen mit Eigenschaften beschrieben.

Projekttyp	**Beschreibung und Eigenschaften**
Testsuite-Projekt	Wird bei Erstellung einer Solution automatisch erstellt. Gute Eignung für Tests, die über den Recorder aufgezeichnet werden. - eine oder mehrere Testsuiten - ein oder mehrere Repositories - Aufnahmemodule, Codemodule oder Modulgruppen - direkt ausführbar, Ranorex erstellt ausführbare Datei bei der Erstellung des Projektes
Modulbibliothek-Projekt	Für die Verwaltung, Organisation und Speicherung von Modulbibliotheken und Repositories geeignet. - ein oder mehrere Repositories - Aufnahmemodule, Codemodule oder Modulgruppen - nicht ausführbar, Ranorex Studio erstellt eine DLL-Datei[1]
Klassenbibliothek-Projekt	Wird für den Erhalt von Klassen und Methoden in Programmieranwendungen verwendet. - nicht ausführbar, Ranorex Studio erstellt eine DLL-Datei[1]
Konsolen-anwendungs-Projekt	Klassen und Methoden werden für Befehlszeilenanwendungen programmiert und kompiliert. - direkt ausführbar, Ranorex erstellt ausführbare Datei bei der Erstellung des Projektes
Windows-Anwendungs-projekt	Klassen und Methoden werden für Windows-Anwendungen programmiert und kompiliert. - direkt ausführbar, Ranorex erstellt ausführbare Datei bei der Erstellung des Projektes

[1] DLL (Dynamic Link Library)

Tabelle 3.2: Projekttypen in Ranorex

Testsuite verwaltet die Testszenarien und definiert diese. Es ist die zentrale Stelle für die Testorganisation. Datenverknüpfungen und globale Parameter werden hier übergeordnet festgelegt. Für die Strukturierung erfolgt eine Unterteilung in Testfälle, intelligente Ordner, Module, Modulgruppen und Setup- oder Teardown-Regionen.

Ein Intelligenter Ordner ist ein Strukturelement in der Testsuite, an dem Datenbindungen erfolgen können. Ist er ein Bestandteil eines Testfalls, erhält er Module und Modulgruppen.

Keywordgesteuerte Tests erlauben es einen Testprozess in Testfälle und zusätzlich Testabschnitte zu unterteilten, um einzelne Aktionen in Module zu gliedern. Diese erhalten nachvollziehbare Keywords mit Namen wie 'Anmelden', 'Fenster schließen' oder 'Anwendung Starten'. Schritte zur Ausführung und Automatisierung, wie Validierungen erfolgen in getrennten Modulen. Durch die einfachen Funktionen der Module beschränkt sich der Einsatz nicht auf einen Browser und eine bestimmte Anwendung.

Datengetriebene Tests ermöglichen die automatisierten Tests über Daten zu steuern. Für die Realisierung werden Daten aus Tabellen eingebunden und mit Variablen verknüpft. Die Daten können in einer CSV-Datei, eine Excel-Tabelle oder ein SQL-Datenbank vorliegen. Die SQL-Datenbank kann eine spezielle Testdatenbank oder eine gesamte Produktdatenbank sein.

Cross-Browser-Tests ermöglichen es einen Test mit einem Browser zu erstellen und diesen über den Einsatz von Variablen in den Browsern Chrome, Firefox, Internet Explorer und Microsoft Edge auszuführen.

Validierung erlaubt die Prüfung und Sicherstellung, dass Ansichten und Elemente in einer Anwendung bestimmten Werten oder Vorgaben entsprechen. Die Prüfungen erfolgen oft durch Vergleiche auf die Existenz, einen Text oder der ID eines Objektes.

Ranorex Spy ist ein eigenständiges Testtool für die Objekterkennung und kann eigenständig oder in Ranorex Studio angewendet werden. Es identifiziert UI-Elemente, wie Buttons und Felder, Steuerelemente und Beziehungen wie z.B. Abhängigkeiten von Elementen. Die Zuordnung erfolgt mit der RanoreXPath-Syntax. Die aufgezeichneten Elemente werden in einem zentralen Objekt-Repository gespeichert, verwaltet und zur Verfügung gestellt.

Objekt-Repository speichert automatisch alle Objekte die in einem Testsuite über Ranorex Spy oder den Recoder aufgezeichnet werden. Das Repository erlaubt eine Trennung zu den Transaktionen, wodurch ein Repository mit Objekten für mehrere Codemodule und Projekte verwendet werden kann. Die Organisation des Repositories erfolgt hierarchisch, durch Ordner wird eine Struktur erreicht. Jedes Objekt

wird mit einem eindeutigen aussagekräftigen Namen benannt, hat einen Pfad (das logische Mapping) und Screenshot. Werden in Modulen oder bei der Aufzeichnung Objekte abermals verwendet, erfolgt eine direkte Zuordnung des Objektes zur Aktion.

Report ist ein Testbericht, der bei der Testausführung in Ranorex Studio erstellt wird. Dieser beinhaltet den gesamten Testlauf und enthält gegebenenfalls Fehlermeldungen. Der Bericht kann in ein JUnit-kompatibles Format oder in eine PDF-Datei umformatiert werden. Alle Aktionen in einem Test sind Ebenen zugeordnet, wie Info, Wahrung oder Fehler. Diese dienen als Filter und können individuell für die Analyse angepasst werden. Zusätzlich können Video-Berichte aktiviert werden. Während der Testdurchführung auf einem Agenten kann über einen anderen Rechner die Berichtsausführung angezeigt werden.

Befehlszentrale für Web- und Mobile Tests ermöglicht eine Trennung der Testsuite von der ausgeführten Umgebung. Tests können so auf unterschiedlichen Web- oder mobilen Endpunkten ausgeführt werden.
Für die Skalierung der Tests ist eine parallele Testausführung möglich. Die Umsetzung erfolgt über Ranorex Remote oder einen Ranorex Parallel Runner.

Ranorex Remote Tests können manuell oder mit einem Agenten auf einem entfernten Rechner oder in einer Virtuellen Maschine im Netzwerk ausgeführt werden. Zuvor muss ein ausführbarer Build des Tests erstellt werden. Die manuelle Ausführung erfolgt in einem Terminal über eine RDP-Verbindung (Remote Desktop Protocol). Der Agent ist ein eigenständiges Tool mit einem Kontrollzentrum, dem Remote-Pad. In diesem werden die Builds bereitgestellt, ausgeführt und verwaltet. Nach der Testausführung erfolgt eine Benachrichtigung. Beide Optionen benötigen eine Runtime Floating License, der Agent nur bei Ausführung.

Konfiguration der Testausführung mit intelligenten Ordnern und Testfällen lassen sich für eine individuelle Ausführung von Tests aktivieren und deaktivieren. Für die Unterscheidung von unterschiedlichen Szenarien können unterschiedliche Run-Konfigurationen erstellt werden. Bei der Testausführung werden alle Testsuite-Elemente die aktiviert sind mit eingebunden und ausgeführt. Weitere Einstellungsmöglichkeiten sind die Deaktivierung oder Aktivierung von einzelnen Testsuite-Elementen. Testfälle und intelligente Ordner können in mehreren Iterationen oder bis zur erfolgreichen Durchführung in einer festgelegten Anzahl von Wiederholungen ausgeführt

werden. Weiter ist es möglich für den Report verschiedene Level einzustellen und ein festgelegtes Fehlverhalten zu konfigurieren.

3.1.2 Jenkins

Im NDR wird Jenkins für die Überwachung und Ausführung von automatischen Ranorex-Tests mit Hilfe einer Distributed Build umgesetzt. Um die Test in Jenkins auszuführen werden Jobs erstellt. Diese teilen sich in verschiedene Typen auf. Im NDR kommt der 'Free Style' Job zum Einsatz. In diesem lassen sich die automatischen Ranorex-Tests einbinden. Der Jenkins Host (zentraler Rechner) ist im NDR auf einem Windows-Server 2016 installiert und über eine Weboberfläche erreichbar. In dieser erfolgen alle Konfigurationen und Einstellungen. Für die Integration der Ranorex-Tests wurde das Plugin 'Ranorex Test Execution Plugin' installiert und konfiguriert.
Es erfolgen nun einige Erläuterungen und Beschreibungen. Diese beinhaltet die Erklärung des Distributed Build und die Beschreibung aller Job-Typen im NDR. Im Anschluss werden Funktionen des 'Free Style' Jobs erklärt, die für die Überwachung und Steuerung von automatisierten Ranorex-Tests für Webanwendungen sinnvoll sind.

Beim **Distributed Build** [10], welches eine verteilte Build oder Remote Architektur ist, arbeitet ein zentraler Rechner mit Agenten auf anderen Rechnern zusammen. Es wird ermöglicht eine Vielzahl von Projekten auszuführen, ohne den zentralen Rechner auszulasten. In dieser Arbeit wird der zentrale Rechner auf dem Jenkins installiert ist als Host, die Rechner auf die Jenkins zugreift als Knoten bezeichnet.
Der Host übernimmt alle Aufgaben für das Build-System. Diese beinhalten die Planung von Jobs, die Überwachung der Agenten, das Zuweisen von Build-Jobs an die Agenten, die Ausführung von Build-Aufträgen sowie die Aufzeichnung und Darstellung der Build-Ergebnisse auf einer Konsole. Beim NDR sind die Knoten Windows 10 Rechner, auf denen ein Jenkins Agent installiert wird. Auf diesen ist ein Testprofil hinterlegt auf das Jenkins zugreift und die Ranorex-Tests erstellt werden.
Bei der Konfiguration eines Agenten in Jenkins wurden einige wichtige Einstellungen vorgenommen, auf die nun näher eingegangen wird. Das Verzeichnis auf dem Knoten ist ein absoluter lokaler Pfad, dieser lautet: `C:\Jenkins\Workspace\Webanwendungen`
In diesem liegen die Ranorex-Tests auf die Jenkins zugreift. Die maximale Anzahl der gleichzeitigen Builds auf dem Knoten wurde auf eins begrenzt, da auf einem Rechner immer nur ein Ranorex Test zur Zeit ausgeführt werden kann. Ein Agent kann über verschiedene Methoden gestartet werden. Die Agenten im NDR stellen die Verbindung zum

Host selbst her. Dieses wird über das JNLP-Protokoll (Java Network Launch Protocol) auf Windows-Rechnern erreicht, in dem dieser eine TCP-Verbindung zum Jenkins-Host herstellt. Für diese Methode sind für Jenkins die Ports des Agenten in der ‘Globale Sicherheit‘ konfiguriert worden.
Der Status der Agenten wird durch eine Knotenüberwachung geprüft und der Agent als online oder offline angezeigt. Die Prüfung selbst erfolgt auf den Bedingungen wie der Antwortzeit und dem freien Speicherplatz.

Im installierten Jenkins können die **Job-Typen** ‘Free Style‘ Projekt, ‘Multikonfigurationsprojekt‘ und ‘Pipeline‘ erstellt werden. Jeder Job ist ein Build-Projekt. Das ‘Free Style‘ Projekt erlaubt die Kombination von einem Build-System mit management-Aufgaben. Das ‘Multikonfigurationsprojekt‘ eignet sich für anspruchsvollere Projekte und plattformspezifische Builds. Die ‘Pipeline‘ erlaubt die Orchestrierung von Arbeitsabläufen. Diese werden für Projekte verwendet die nicht in ‘Free Style‘ Jobs passen. Die Pipeline kann z.B. mehrere Phasen enthalten, wie eine Build-, Test- und Deploy-Phase. Für die Umsetzung der automatisierten Ranorex-Tests kommen das ‘Free Style‘ Projekt und ‘Multikonfigurationsprojekt‘ in frage, da in diesen die Ranorex-Tests eingebunden werden können. Das ‘Multikonfigurationsprojekt‘ umfasst mehr Konfigurationen als für die Umsetzung benötigt werden.

Die **‘Free Style‘ Jobs** können auf den Host über die Agenten ausgeführt werden. Dazu muss im Job der absolute Pfad angegeben werden. Die Ausführung der Tests in Jenkins kann in festgelegten Zeiten oder und manuell erfolgen. Es ist möglich den Quellcode aus einer Verwaltung wie Git auszuchecken, wenn das Plugin hierfür installiert wird. In den Build-Schritten müsste der auszucheckte Quellcode über MSBuild gebaut werden. Die Build-Schritte erlauben die Konfiguration und Einbindung von Testsuiten von Ranorex-Tests. In den Post-Build können Feedbacks über E-Mail Benachrichtigungen eingestellt werden.

3.2 Anforderungen

In diesem Abschnitt erfolgt eine Auflistung aller Anforderungen an das Testkonzept, diese sind am Beispiel der Webanwendung ‘NDR Yii_basic‘zu behandeln und umzusetzen. Für die Übersicht erhalten alle Anforderungen eine ID. Diese teilen sich in funktionale (FA) und nicht funktionale Anforderungen (NFA) auf.

ID	Anforderung
1 FA	*Testumfang* Die Tests sollen die Grundfunktionalität, wie An-, Abmelden und das einfache Erstellen, Bearbeiten und Löschen eines Objektes umfassen.
2 FA	*Teststruktur* Die Tests sollen als Grundlage für weitere Tests dienen, deshalb ist bei der Erstellung darauf zu achten, dass die Grundstruktur der Tests logisch, nachvollziehbar und erweiterbar ist.
3 FA	*Datenbasis* Während der Testausführung muss sichergestellt werden, dass die Datenbasis nach der Testausführung den gleichen Umfang hat.
4 FA	*Browser* Die Tests sollen später in den Browsern Chrome, Firefox, Edge und Safari ausgeführt werden. Bei der Erstellung der Tests soll der Chrome Browser verwendet werden.
5 NFA	*Erweiterbarkeit* Auf Grundlage des Testkonzeptes soll es möglich sein weitere Test zu entwickeln.
6 NFA	*Wiederholbar* Die Tests sollen in regelmäßigen Abständen wiederholt werden, um die Funktionalität der Webanwendung für den Nutzer zu gewährleisten. Dazu ist eine mögliche Umsetzung aufzuzeigen.
7 FA	*Automatisierte Tests* Die automatisierten Tests sollen mit Ranorex Studio erstellt werden.
8 FA	*Testdaten* Testdaten für die automatisierten Tests sind durch externe Dateien in diese einzubinden.
9 FA	*Steuerung und Überwachung* Im Rahmen der Umsetzung ist zu zeigen, wie eine mögliche Überwachung und Steuerung der Tests aus Jenkins heraus erfolgt und wie diese umzusetzen ist.
10 FA	*Feedback* Schlägt ein automatisierter Tests fehl, soll eine E-Mail über Jenkins an den Tester oder an den Anwendungsbetrieb versendet werden.

11 FA *Versionsverwaltung*
Das Versionsverwaltungssystem Git soll zum Einsatz kommen. Für jede Webanwendung wird ein eigenes Git-Repository erzeugt, welches lokal ist und über eine Gitignore verfügt. In diesem Git-Repository sollen die Testdaten ohne die Login Daten und die Ranorex Solution mit automatisierten Tests für die jeweilige Webanwendung gespeichert werden.

3.3 Design

Für die Umsetzung der Anforderungen wird nun auf des Design des Testkonzeptes eingegangen und in mehrere Abschnitte unterteilt. Im ersten Abschnitt wird die Rolle des Testers im Entwicklungsprozess erläutert und wie sich die neue Rolle auf das Testkonzept auswirkt. Im zweiten Abschnitt wird eine Übersicht des Testablaufs in den Umgebungen mit den Tools beschrieben. Diese umfasst die Steuerung und Überwachung der Tests durch Jenkins. Im dritten Abschnitt wird auf Testfall Entwicklung eingegangen.

Rollenverteilung bei der Entwicklung
Bei der Entwicklung von Webanwendungen existiert eine klare Rollenverteilung. Diese wird um die Rolle des Testers erweitert, um die Tests zu automatisieren. Dieser kann, muss allerdings nicht zwingend, über Programmierkenntnisse verfügen, da die Tests mit dem Recorder in Ranorex erstellt werden sollen. Die Rolle des Testers kann dementsprechend durch einen Entwickler oder eine andere Person erfolgen.
Betrachten wir nun die Entwicklungsschritte. In der Abbildung 3.4 ist zu sehen, dass der Tester in mehrere Schritte involviert ist. In den Entwicklungsschritten ist der Tester erst spät involviert, dieses wird als ‘Reaktiver Ansatz‘ bezeichnet. Der Tester bekommt in diesem Ansatz das zu testende Objekt, bzw. die Webanwendung erst nach der kompletten Entwicklung, Entwicklung eines Teilfeatures oder einer abgeschlossenen Änderung zum Testen. Aus diesem Grund ist er in den Schritten ‘Anforderungen‘ und ‘Analyse & Entwurf‘ nicht und im Entwicklungsschritt ‘Realisierung‘ nur bedingt beteiligt. Wird eine Webanwendung in mehreren Teil-Features auf die Testumgebung deployed, erhält der Tester die Webanwendung bereits während der Realisierung zum Testen. Erfolgen später Änderungen dieser Features, muss der Tester diese in den Tests mit aufnehmen und anpassen. Der Schritt ‘Prüfen‘ beinhaltet das Testen von Webanwendungen. Sind im Schritt ‘Abnahme & Auslieferung‘ Änderungen durch den Kunden gewünscht, sind

diese in den Tests anzupassen. Wenn Tests fehlschlagen benötigt der Anwendungsbetrieb, im Schritt ‘Wartung & Support‘, gegeben falls Hilfe bei der Anpassung der Tests.

Zusätzlich zum Reaktiver Ansatz und besteht die Dokumentation der Webanwendung meinst nur aus Notizen. Durch diese Einschränkungen kommt für die Umsetzung der Testfälle im Testkonzept nur das explorative Testen in frage.

Entwicklungsschritte	**Rollen**					
	Kunden	Anforderungsmanagement	Entwicklungsprojektleitung	Entwicklung	Tester	Anwendungsbetrieb
Idee / Projektvorschlag	X					
Anforderungen		X				
Analyse & Entwurf	X		X			
Realisierung				X	X	
Prüfung			X	X	X	
Abnahme & Auslieferung	X			X		X
Wartung & Support					X	X

Tabelle 3.4: Entwicklungsschritte bei Webanwendungen mit automatisieren Tests

Übersicht des Testablauf

Die Abbildung 3.3 zeigt die Übersicht des Testablaufes die im Testkonzept umgesetzt werden soll. In diesem sind alle Komponenten, Umgebungen, Tools und beteiligten Personen zu sehen. Bisher werden alle Webanwendungen im NDR manuell getestet. Durch das Testkonzept aus der Abbildung 3.2 soll gezeigt werden, wie die ‘internen Tests‘ durch automatisierte Tests ersetzt und erweitert werden können. Die Erweiterung beinhaltet, dass die Tests nicht nur in der Testumgebung, sondern auch in der Produktivumgebung erfolgen. Die ‘Abnahmetests‘ durch den Kunden und die ‘lokalen Tests‘ in der Entwicklungsumgebung erfolgen weiterhin manuell. Die Abbildung 3.3 zeigt die automatisierten Tests durch Ranorex und manuellen Tests. Die Erstellung und Ausführung der automatisierten Tests erfolgt für die Test- und Produktivumgebung in zwei getrennten Testsuiten auf einem Windows-Rechner (Knoten), auf den Jenkins durch einen Tester

oder Entwickler zugreifen kann. Jenkins überwacht und steuert die Testausführung. Vom Knoten wird während der Ausführung der Tests ein Report an Jenkins übermittelt. Die Ausführung kann auf dem Knoten oder über die Konsolenausgabe von Jenkins im Host verfolgt werden. Je nachdem in welcher Umgebung die Tests ausgeführt wurden, erfolgt eine Mitteilung in Form eines Feedbacks an den Anwendungsbetrieb oder den Tester.

Durch diese Anpassung in der Übersicht werden Fehler in einer Webanwendung in der Produktivumgebung frühzeitig erkannt und der Anwendungsbetrieb automatisch infor-

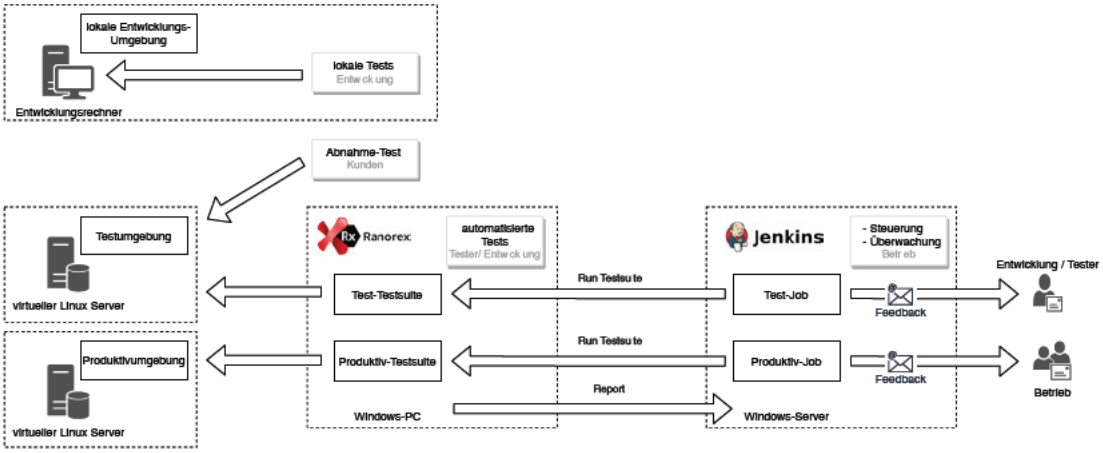


Abbildung 3.3: grundlegender Testablauf

Entwicklung der Testfälle

Im Testkonzept werden die Testfälle in zwei Phasen in den Tests umgesetzt. Anschließend folgt die Steuerung und Überwachung der Tests. Zuerst werden die Testfälle durch das explorative Verfahren erstellt. Im Anschluss erfolgt die Automatisierung der Tests.

Die Umsetzung der Testfälle erfolgt durch Capture und Replay mit dem Tool Ranorex. In diesem werden die Testfälle aufgezeichnet (Capture) und können im Anschluss wiedergegeben werden (Replay). Bei der Aufzeichnung erfolgt die Erfassung der Objekte über den Ranorex Spy und Vergleiche über Validierungen.

Die Testautomatisierung in Ranorex ist durch datengetriebene und schlüsselwortgetriebene Tests möglich. Bei datengetriebenen Tests, auch als Data-Driven Tests bezeichnet, werden die Tests über Testdaten gesteuert. Im Testkonzept ist vorgesehen, dass Daten über Dateien in die Tests eingebunden werden. Es ist nicht vorgesehen, dass die Tests komplett datengetrieben sind. Schlüsselwortgetriebene Tests, auch als Keyword-Driven

Tests bezeichnet, kommen zum Einsatz wenn Testabläufe mehrfach verwendet werden. Der Testablauf wird dazu in einzelne Teile bzw. Module unterteilt. Module mit einfachen grundlegenden Funktionen ohne Validierungen erhalten aussagekräftige Schlüsselwörter. Diese müssen unabhängig von den Tests gespeichert werden, um in unterschiedlichen Tests für Webanwendungen eingebunden werden zu können.

Die Phase des explorativen Testens wird je nach Testbedingung für die Webanwendung in verschiedene Abläufe unterteilt. Bei einer neuen Entwicklung kommt das explorative Testen sofort zum Einsatz. Werden neue Features eingeführt oder Änderungen vorgenommen, müssen bereits bestehende Tests erneut ausgeführt werden. Schlagen die Tests fehl, ist zu ermitteln wo und welche Änderungen in den Tests notwendig sind. Um festzustellen welche Funktionen neu hinzugekommen sind oder verändert wurden, ist eine manuelle Ausführung der Webanwendung, unabhängig von dem Testergebnis, angebracht. Die Webanwendung wird praktisch auf ihre Funktionen hin erforscht und diese mit bestehenden Dokumentationen, falls vorhanden, verglichen. Im Anschluss werden die Charta definiert, welche die Funktionen berücksichtigen.
Müssen für die Umsetzung der Charta Module in Ranorex angepasst werden, ist drauf zu achten, um was für Module es sich handelt und in welchen Tests von Webwendungen diese ausgeführt werden. Bei Schlüsselwortgetriebenen Modulen ist es deshalb sinnvoll neue Module zu erstellen die diese Charta umsetzen oder Module in kleinere aufzuteilen und Teile der Charta getrennt zu behandeln. Module die nur innerhalb der Tests verwendet werden, können erweitert und verändert werden, da diese nicht in anderen Webanwendungen zum Einsatz kommen.
Ist für die Umsetzung der Charta keine Änderung von bestehenden Modulen notwendig, so erfolgt hier ebenfalls der Einsatz von explorativen Tests wie bei neuen Entwicklungen. Bei neuen Entwicklungen ist es sinnvoll, die Webanwendung manuell auszuführen, um eine Übersicht über die Funktionen dieser zu erhalten und diese mit bestehenden Dokumentationen zu vergleichen. Im Anschluss daran werden Charta definiert die zu Beginn Grundelemente enthalten wie Öffnen und Schließen der Webanwendung. Im Anschluss daran das An- und Abmelden eines Benutzers. Als Basis für die Umsetzung sollte die ‘NDR Yii_basic‘ Anwendung verwendet werden, diese beinhaltet bereits Funktionen. Sie sind für die jeweilige Webanwendung nun anzupassen und zu erweitern. Existiert die Testbasis ‘NDR Yii_basic‘ Anwendung nicht, ist diese durch explorative Tests zu erstellen.
Beim explorativen Testen in Webanwendungen werden für definierte Charta in einem Testfall, Modul oder eine Gruppe von Modulen erstellt, ausgeführt, analysiert und in-

terpretiert. Ist das Ergebnis nicht zufriedenstellend, erfolgt ein weiterer Durchlauf für das Charta. Ist das Ergebnis erfolgreich, wird das nächste Charta in Angriff genommen. Dieser Durchlauf wird so lange wiederholt, bis alle definierten Charta getestet oder eine bestimmtes Zwischenziel erreicht wurde. [4, Kapitel 1,2]

Charta, die durch explorativen Tests definiert werden, umfassen die Oberflächentest der Webanwendung. Diese finden aus Sicht eines Anwenders statt, prüfen die Funktionen und sind praktisch Systemtests.

3.4 Technische Aspekte für Ranorex

Für die Umsetzung des Designs und der Anforderungen ist es notwendig Randbedingungen für Ranorex zu spezifizieren. Für Ranorex erfolgt nun eine Auflistung, in dieser werden in einzelnen Punkten Funktionen und Eigenschaften von Ranorex zusammengefasst, die inhaltlich zusammen gehören oder sich ausschließen. Zusätzlich wird drauf hingewiesen, welche im Rahmen der Arbeit nicht zum Einsatz kommen.

ID	Beschreibung
1	*Interfaces und Schnittstellen* Interfaces und Schnittellen sind für die Umsetzung der automatisierten Tests, Ausführung und Überwachung zwingend erforderlich. Für den Zugriff auf Browser bei der Aufnahme und Ausführung der Tests muss über den Instrumentierungsassistenten in den verwendeten Browsern das Add-on Ranorex installiert werden. Außerdem muss die Integration von Ranorex mit dem CI-Tool Jenkins erfolgen.
2	*Recorder und Code-Editor* Für die Erstellung der Testfälle ist der Recorder zu verwenden. Der Code-Editor soll nur zum Einsatz kommen, wenn die Testfälle nicht mit dem Recorder umgesetzt werden können.
3	*Fokussierung* Um bei der Aufnahme von Aktionen keine unbestimmten Prozesse mit aufzunehmen, ist eine Whitelist zu erstellen.
4	*Konzept und Cross-Browser-Tests* Das Konzept Web ist auszuwählen, es bietet sich für das Testen von Webanwendungen mit vorgegebenen Browsern an. Für die Umsetzung von einem Test mit mehreren Browsern sollen Cross-Browser-Tests verwendet werden.

5 *Solution und Projekte*
Die Umsetzung der automatisierten Tests sollen wenn möglich in einer Solution mit zwei Projekten für die Test- und Produktivumgebung realisiert werden. Als Projekttyp ist das Testsuite-Projekt zu bevorzugen.

6 *Testsuite*
Es ist eine beispielhafte Struktur eines Testszenarios mit Testfällen und Ordnern zu erstellen, die als Vorlage für weitere Testfälle und Tests dienen. Dabei ist der Einsatz von grundlegenden Funktionen wie globale Parameter und Datenverknüpfungen zu zeigen.

7 *Keywordgesteuerte Tests*
Die Module in den Testfällen sind so zu strukturieren und aufzuteilen, dass diese in der Test- und Produktivumgebung und eventuell auch in anderen Webanwendungen zum Einsatz kommen können. Die Realisierung dieser Tests soll durch Keywordgesteuerte Tests erfolgen.

8 *Datengetriebene Tests*
Daten zum Login, für Vergleiche und definierte Werten sollen über externe Dateien in die Tests eingebunden werden.

9 *Validierung*
Für den Vergleich und die Prüfung von Ansichten, Elementen und Objekten sollen Validierungen verwendet werden.

10 *Ranorex Spy und Objekt-Repository*
Die Pfade von aufgezeichneten Objekten über den Recoder, sind im Ranorex Spy gegebenenfalls anzupassen.
Das Repository ist für die Übersicht und Wiederverwendbarkeit zu strukturieren. Den Objekten müssen eindeutige Namen zugewiesen werden.

11 *Report und Testausführung*
Für die Ausführung von individuellen Testfällen und Wiederholungen ist die Testausführung in Ranorex anzupassen. Der Report ist zu analysieren und gegebenenfalls zu ändern, um Fehler bei der Ausführung zu ermitteln, spezifischere Meldungen zu erhalten und den Funktionsumfang mit den Spezifikationen abzugleichen.

12 *Befehlszentrale für Web- und Mobile Tests, sowie Remote*
Die Testausführung und Überwachung sollen über Jenkins erfolgen. Deshalb wird auf die Befehlszentrale für Web- und Mobile Tests, sowie Remote-Tests verzichtet und diese im Rahmen dieser Arbeit nicht weiter betrachtet.

4 Umsetzung des Testkonzeptes

In diesem Kapitel werden die Umsetzung und Auswahlmöglichkeiten eines automatisierten Tests mit den Tools Ranorex und Jenkins beschrieben. Für die Realisierung und Durchführung der Tests ist es notwendig, die Struktur auf dem Testrechner zu definieren und Einstellungen vorzunehmen, um den Zugriff von Jenkins auf Ranorex zu gewährleisten. Es ist zu beachten, dass der Test auf Basis von Beispielfunktionen im Basis-Projekt realisiert wird und somit als Vorlage und Beispiel für eine mögliche Umsetzung in den aktiven Webanwendungen dient.

4.1 Konfiguration und Einstellungen des Testrechners

Der Testrechner ist ein Standard Windows-Client, der für Testzwecke genutzt wird. Für den Zugriff von Ranorex auf die Browser wurden die Richtlinien für die Browser Chrome, Firefox und Edge angepasst. Diese ermöglichen es in den Browsern das Add-on Ranorex Studio dauerhaft zu speichern. Für den Browser Firefox wurde zusätzlich die Einstellungen Proxy und vorherige Sitzung wiederherstellen deaktiviert. Auf den Browser Safari muss leider verzichtet werden, da es diesen nicht in der aktuellen Version für Windows 10 gibt. Eine Installation in einer älteren Version würde die Sicherheit beeinträchtigen und nicht dem derzeitigen Stand auf den Produktiven Systemen entsprechen. Aus diesen Gründen wird von der Anforderung diesen zu testen abgewichen.
Der Test-User hat Admin Rechte erhalten, um Git, TortoiseGit und die Add-ons über den Instrumentierungsassistent von Ranorex zu installieren.
Für die Versionsverwaltung werden Git und TortoiseGit installiert. Das Repository, in dem der automatisierte Test liegt, ist ein Git-Repository mit einer Gitignore. Diese sorgt dafür, dass Daten, die nur für die Konfiguration der Projektumgebung benötigt werden, nicht mit verwaltet werden.
Ranorex ist zur Zeit der Testerstellung in der Version 9.5.1 installiert, diese ermöglicht laut Releases endlich den Einsatz von dem Browser Edge in Ranorex.

Der Pfad für den Ranorex Test ist festgelegt, um den Zugriff aus Jenkins zu ermöglichen. Dieser lautet wie folgt: `C:\Jenkins\Workspace\Webanwendungen` . In diesem Pfad liegen ebenfalls Tests anderer Anwendungen, deshalb wird ein extra Ordner für Webanwendungen erstellt, dieser beinhaltet wiederum einen Ordner für die Test und Logindaten. Die Testdaten befinden sich bei den Tests. Dieser Ordner wird als Git-Repository verwaltet. Wird dieses Repository als Grundlage für neue Tests verwendet, sind die notwendigen Testdaten dabei. Die Logindaten sind getrennt, da es sich um sensible Daten handelt und diese in einem Git-Repository grundsätzlich nicht verwaltet werden dürfen.

4.2 Umsetzung eines automatisieren Tests in Ranorex

In mehren Abschnitten wird die Umsetzung beschrieben und auf die Anforderung eingegangen. Zusätzlich erfolgt eine Erklärung von den möglichen Grundstrukturen von Tests in Ranorex und welche Vor- und Nachteile diese für die Webanwendungen haben.

- Erstellung des automatisierten Ranorex Testes im Git Repository
- Aufbau der Solution
- Steuerung der Tests mit Keywords
- Strukturierung einer Testsuite
- Ranorex Spy und das Objekt-Repository
- Einbindung von Testdaten und Zuordnung zu Variablen
- Umsetzung der Cross-Browser Tests
- Konfiguration und Ausführung der Tests
- Analyse und Aufbau des Reports

Im Anhang A.1 sind Handlungsanweisungen zu finden die folgende Inhalte umfassen:

- Fokussierung
- Vermeidung von Meldungen
- Referenzierung auf DLL-Dateien
- Globale Parameter & Validierungen
- Einbindung von Testdaten

Die Abbildung 4.1 zeigt die Arbeitsumgebung von Ranorex mit der Solution der ‘NDR Yii_basic‘ Webanwendung. Es erfolgt eine Erklärung der Bereiche der Abbildung, da auf diese Bezug genommen wird. Für die Erkennung der Bereiche, sind Teile mit Rechtecken farblich markiert worden. Links im Bild in grün ist die Projektansicht einer Solution zu

sehen. Diese umfasst die Daten und Referenzen die mit einem Projekt verknüpft ist. Das Projekt für die Testumgebung ist aufgeklappt und wurde um die Ordner AnAbmelden, Browser und Objekt erweitert, um die Module zu strukturieren. Das Projekt für die Produktivumgebung ist gleich strukturiert und enthält nur das Modul 'Testsystem' nicht. In Türkis sind die Referenzen des Projektes für die Erläuterung der Keword gesteuerten Tests hervorgehoben. Links unten, in Rosa ist der 'Modul Browser' zu sehen. Dieser beinhaltet alle Module nach Projekten unterteilt und die integrierten DLL-Module, welche blau gekennzeichnet sind. Unten in orange ist der Bereich vom Output und der Error-Meldungen zu sehen. In rot hervorgehoben ist die 'File View', diese ist eine Aktionstabelle und umfasst in diesem Fall drei Ansichten. Die Testsuite der Testumgebung ist ausgewählt und zu sehen. Die blaue Umrandung zeigt die Run Konfigurationen. In gelb ist die erste Zeile des Testfalls 'Objekt_Funktionen' für die Erklärung der Cross-Browser Tests hervorgehoben. In braun ist das DLL-Modul 'ObjektID' mit der Datenbindung hervorgehoben. Rechts in weiß sind Eigenschaften 'Properties'.

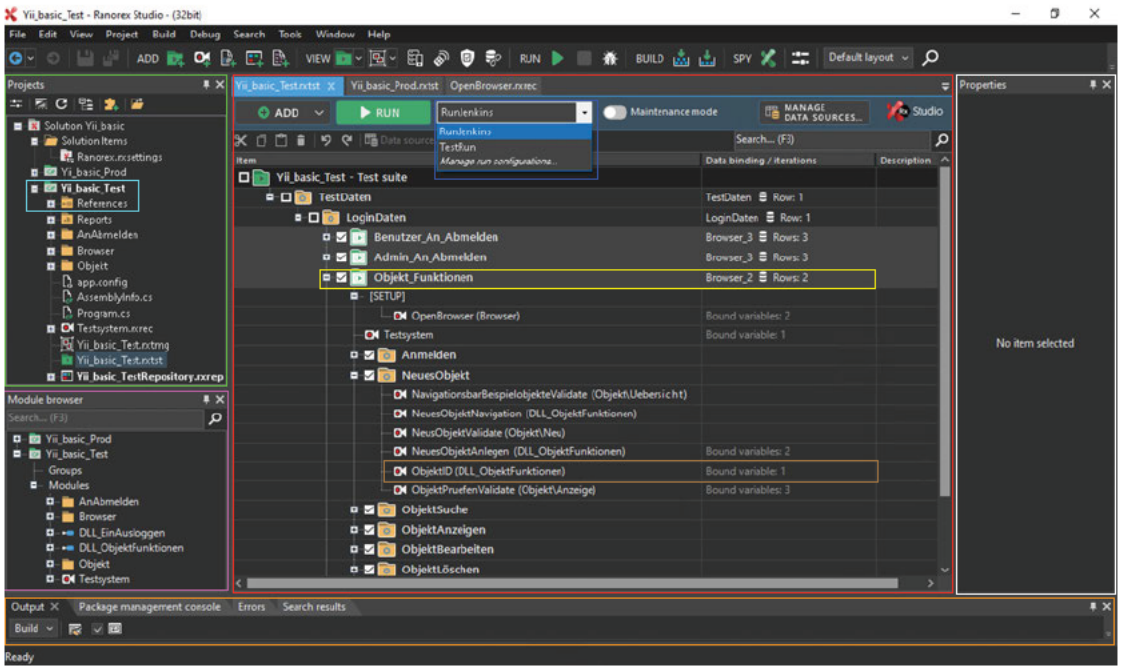


Abbildung 4.1: Arbeitsumgebung von Ranorex mit Testsuite der Testumgebung

4.2.1 Erstellung des automatisierten Ranorex Testes im Git Repository

Für die Erstellung eines neuen automatisierten Tests für eine Webanwendung wird in Ranorex Studio eine Solution ausgewählt. Nun wird der Speicherort, der Projektname und die Programmiersprache festgelegt. Als Projektname wird der Name der Webanwendung mit Erweiterung Test oder Prod, um anzuzeigen für welche Umgebung der Test ist, eingegeben. Als Speicherort ist ein Pfad festgelegt, siehe Abschnitt Abschnitt 4.1. Die Versionsverwaltung ist auszuwählen. Das Anlegen eines neuen Directories wird nicht ausgewählt. Die Solution wird dadurch im Repository erzeugt in dem sich der Git-Ordner befindet. Tritt nachfolgend eine Meldung für die Versionsverwaltung auf, so ist Git auszuwählen. Im nächsten Schritt wird das Konzept Web ausgewählt. Im Anschluss werden die URL und der Browser bestimmt. Auf dessen Basis der erste Testfall in diesem Projekt erzeugt wird. Die Whitelist wird ausgewählt, um den Fokus auf den Browser Chrome zu legen. Im Anschluss erzeugt Ranorex eine Testumgebung mit einem Testsuite-Projekt. Dieses Projekt enthält einen einfachen Testfall mit drei Aufzeichnungsmodulen. Dem ‘StartAUT‘, welches den Browser startet, ein leereres ‘Recording1‘ für die Aufzeichnung von Testaktionen und ‘CloseAUT‘ für das Schließen des Browsers.

4.2.2 Aufbau der Solution

Die automatisierten Tests der Webanwendung Yii_basic erfolgen in einer Solution mit zwei ‘Testsuite-Projekt‘. In der Abbildung 4.1 ist die Unterteilung der Solution zu sehen. Im Rahmen der Arbeit sollen die Tests Keyword-gesteuert sein. Aus diesem Grund wird eine weitere Solution erzeugt. Diese enthält DLL-Module, die wiederum in einzelne Projekte aufgeteilt sind. Siehe Abschnitt 4.2.3 für die Erklärung der Tests mit Keywords.

4.2.3 Steuerung der Tests mit Keywords

Die Testfälle werden für die Umsetzung in einzelnen Module gegliedert. Diese sind teilweise Keyword-gesteuert. Das bedeutet, dass diese Module in der Test- und Produktivumgebung ausgeführt werden können. Für diese Realisierung ist entscheidend, dass nur grundlegende Funktionen ausgeführt werden. Die Objekte sind in einem eigenen Repository gespeichert und haben keine URL Bindung. Validierungen und spezifische Funktionen sind nicht Teil dieser Module.

Die Umsetzung kann in Code erfolgen, durch definierte Methoden, die nach Schlagwörtern benannt sind und in einer Aktionstabelle bzw. eigenen Bibliothek verwaltet werden. Der zweite Ansatz sieht die Aufzeichnung von Modulen vor und wird im Rahmen der Arbeit umgesetzt. Die Module werden in einem Modulbibliotheks-Projekt als DLL-Datei erstellt. Auf das Projekt wird aus den Projekten der automatisierten Tests referenziert. Siehe im Anhang Abschnitt A.1.3. Der Vorteil ist, dass die Module nicht auf ein Projekt begrenzt sind. Änderungen der Module sind nur im Projekt möglich in dem diese erstellt werden. Es erfolgt eine Trennung der Schlüsselwort-Module vom Testentwurf und der Verwendung. Die Module selbst enthalten für einige Funktionalitäten Variablen. Die Bindung der Variablen an Daten und Parameter erfolgt im Testfall durch das Einsetzen der Module.

In Abbildung 4.1 ist zu sehen, dass zwei DLL-Dateien im ‘Modul browser‘ integriert wurden. Dazu wurde auf diese referenziert. Erfolgen Änderungen in den Modulbibliotheks-Projekten, müssen die Referenzen in der Projektansicht aktualisiert werden.

Die Abbildung 4.2 zeigt die einzelnen Modulbibliotheks-Projekte der Solution für die DLL-Module. Das Projekt ‘DLLModulLibary‘ ist ein Testsuite-Projekt und wurde nicht implementiert. Die beiden anderen Projekte sind aufgeteilt in das einfache Ein-, Ausloggen und in die Objektfunktionen. Durch diese Aufteilung kann das Ein- und Ausloggen für alle Webanwendungen verwendet werden. Die Objektfunktionen sind spezifisch und dienen als Vorlage für weitere Projekte.

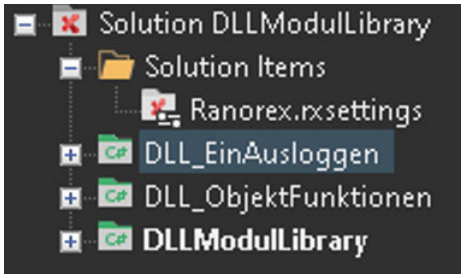


Abbildung 4.2: Solution DDL-Modul Bibliothek

4.2.4 Strukturierung einer Testsuite

Die Testsuite wurde in drei verschiedene Testszenarien unterteilt, um den Testumfang Abmelden, Anmelden und eine Basisfunktionen aufzuzeigen. Die Grundlegenden Szenarien sind das An- und Abmelden von einem Benutzer und dem Admin. Als Basisfunktion wird ein Objekt angelegt, angezeigt, gesucht, bearbeitet und anschließend gelöscht, um die Datenbasis in der Webanwendung nicht zu verändern.
Die Abbildung 4.1 zeigt die grundlegende Struktur der Testsuite für die Testumgebung.

Für die Umsetzung sind in der Testsuite drei Testfälle erzeugt und benannt worden. Diese erhalten jeweils eine Setup- oder Teardown-Region mit Modulen zum Öffnen und Schießen der Browser. Zwischen diesen Regionen werden Module für die Funktionalität eingefügt. Die zusätzlichen intelligenten Ordner in Ranorex ermöglichen es die Testdaten einzubinden, diese an Variablen zu binden oder einen Testfall zu strukturieren. Globale Parameter können für die Testsuite, einem intelligenten Ordner oder Testfall definiert werden. Diese gelten für die darunter liegenden Module, wenn eine Zuordnung erfolgt.

4.2.5 Ranorex Spy und das Objekt-Repository

Die Abbildung 4.3 zeigt das Repository des Modulbibliotheks-Projekt 'DLL_EinAusloggen'. Dieses Repository enthält eine leere DOM für den Pfad der Applikation, da das Projekt mit den Modulen und Objekten für verschiedene Webanwendungen einsetzbar ist. Die einzelnen Objekte sind je nach Modul und Funktion in einzelne Ordner gegliedert und benannt, die den Funktionen der Webanwendung entsprechen. Für die Übersicht und Zuordnung sollte für jedes Objekt eine Aufnahme existieren, siehe 'ButtonTag'.
Die Webanwendungen haben durch die Datenbank in der die Daten gespeichert werden dynamische IDs für Objekte und Elemente. Um Probleme zu vermeiden, können diese durch das Einrichten von Prioritätsregeln im Spy oder eine Analyse der Webapplikation vermieden und unterbunden werden.

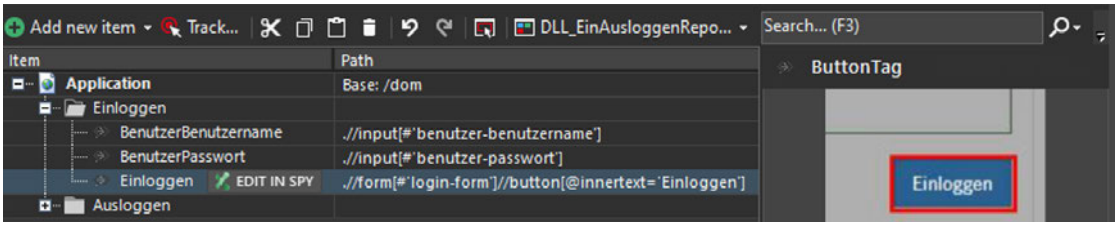


Abbildung 4.3: Objekt-Repository von DLL_EinAusloggen

4.2.6 Einbindung von Testdaten und Zuordnung zu Variablen

Testdaten können in einem intelligenten Ordner oder Testfall verwaltet werden. Allerdings mit der Einschränkung, dass jeweils nur eine Datei eingebunden werden kann. Werden nun für ein Modul individuelle Daten benötigt, ist es sinnvoll, diese in einen eigenen intelligenten Ordner zu speichern. Für Daten, die in allen Testfällen benötigt

werden, ist es ebenfalls sinnvoll, diese in Ordnern über den Testfällen anzuordnen.
Die Testdaten selbst können als Excel-, CSV- Datei oder in einer Datenbank abgelegt werden. Als Speichermedium sind CSV-Dateien zu bevorzugen, da binäre Dateien wie in Excel verwendet, sich nicht als reinen Text ohne eine zusätzliche Informationsebene darstellen lassen.
Die Daten sind tabellarisch abgelegt. In der ersten Zeile steht für jede Spalte der Name der Variablen dem diese zugeordnet wird, darunter die Werte. Hat eine Variable mehr als einen Wert muss diese in einer weiteren Tabelle gespeichert werden, da ein Testfall für jeden Wert abermals ausgeführt wird. Dieses wird als Daten-Iteration bezeichnet. Sensible Eingaben und Werte aus einer Datei wie Passwörter und User-Daten können für den Report verborgen werden. Beim Einbinden der Daten-Ressourcen müssen diese explizit maskiert werden.

In der Abbildung 4.1 ist zu sehen, dass über den Testfällen zwei intelligente Ordner angeordnet sind. Der erste ist für allgemeine ‘TestDaten‘, die in vielen Modulen verwendet werden. Der zweite für ‘LoginDaten‘, sensible Daten die maskiert sind. Um beide Datenquellen in allen Testfällen verwenden zu können liegen diese in überordneten intelligenten Ordnern. In der Spalte ‘Data binding/iterations‘ ist zu sehen, dass zu einigen Modulen eine Datenzuordnung erfolgt, diese wird durch ‘bound variable: #‘ dargestellt.

4.2.7 Umsetzung der Cross-Browser Tests

Im Modul zum Öffnen der Browser werden Variablen benötigt. Eine für das Setzen der URL und die zweite für den Aufruf des Browsers. Diese müssen in getrennten CSV-Dateien gespeichert werden, da die URL gleich bleibt und der Name des Browsers verschieden ist. Wie bereits bei der Einbindung von Testdaten erwähnt, erfolgt für jeden Browser eine Daten-Iteration für die Testfälle. In Abbildung 4.1 ist zu sehen, dass für die Testfälle jeweils eine Datei eingebunden wurde und mehrere ROWs angegeben sind. Die ROWs sind die Daten-Iterationen. Für jede Iteration wird der Testfall mit einem anderen Browser ausgeführt.

4.2.8 Konfiguration und Ausführung der Tests

Für die Realisierung der Testfälle ist es wichtig einzelne Funktionen separat auszuführen um diese zu prüfen. Eine Umsetzung ist möglich, da sich jeder Testfall, intelligente

Ordner, Modul und Aktionen individuell de- und aktivieren lassen. Um in einem Testszenario nur einen bestimmten Testfall in Jenkins auszuführen, werden individuelle Run-Konfigurationen erstellt. Diese können in Jenkins bei der Erstellung eines Jobs angegeben werden. In der Abbildung 4.1 ist zu sehen, dass eine Run-Konfiguration (in blau) für die Testentwicklung und eine für die Ausführung in Jenkins erstellt wurde. Auf mehrere Iterationen oder die Wiederholung bis zum Erfolg wird verzichtet, um die Komplexität der Berichte nicht zu erhöhen und in Jenkins die Übersicht der Testausführung zu gewährleisten. Die Report Level werden in den Standardeinstellungen verwendet und ein festgelegtes Fehlverhalten wird im Rahmen der Arbeit nicht konfiguriert.

4.2.9 Analyse und Aufbau des Reports

Wird ein Test ausgeführt erzeugt Ranorex automatisch einen Report, bzw. Bericht. Dieser hat einen festgelegten Aufbau und Inhalt. Der Berichtskopf beinhaltet den Namen des Testsuite (Aufzeichnungsmoduls), die System- und Testdaten, einen Fehler- oder Warnungszähler und das Testergebnis. Die Berichtsdetails strukturieren sich nach dem Testsuite. Diese beinhalten jedes Strukturelement und Element der Testaktion. Die Informationen teilen sich in die Spalten Zeit, Ebene, Kategorie und Meldung auf. Sind die Tests Datengesteuert, wird ein intelligenter Ordner für die Datenbindung durchlaufen und die Variablen werden mit den zugehörigen Datenwerten angezeigt. Bei geheimen Daten erfolgt eine automatische Maskierung. Sind einer Variablen mehrere Datenwerte zugeordnet, einspricht jeder Datenwert einer Test-Iteration. Für jede Iteration erfolgt eine separate Testausführung und -anzeige.
Im Bericht gibt es grundlegende und spezielle Aktionen. Spezielle Aktionen sind Benutzerdefinierte Protokollmeldungen (Log Meldungen), Validierungen und Separatoren. Separatoren strukturieren die Aktionen in der Aktionstabelle und im Bericht. Alle anderen Aktionen, wie ein Klick, sind grundlegende.
Der Ranorex Bericht hat Funktionen für die Filterung und Analyse. Die Filter unterteilen sich in Erfolg, Fehlschlag und Blockierung. Von jeder Meldung ist es möglich zum Element oder zur Aktion im Testsuite zu springen. Wird der Report außerhalb von Ranorex geöffnet, kann zu einer Meldung das Berichtselement in Ranorex Spy geöffnet werden.

4.3 Umsetzung der Überwachung und Steuerung mit Jenkins

Es erfolgt nun eine Erklärung der Umsetzung für die Überwachung und Steuerung der automatisierten Ranorex-Tests. Für die Integration einer Ranorex-Testsuite in Jenkins wird zuerst die Steuerung der Weboberfläche in Abbildung 4.4 beschrieben. [9] Im Anschluss wird die Erstellung eines Jenkins Jobs erläutert indem ein Ranorex-Testsuite eingebunden wird. Abschließend wird die Konsolenausgabe von einem erfolgreichen Job-Build detailliert beschrieben.

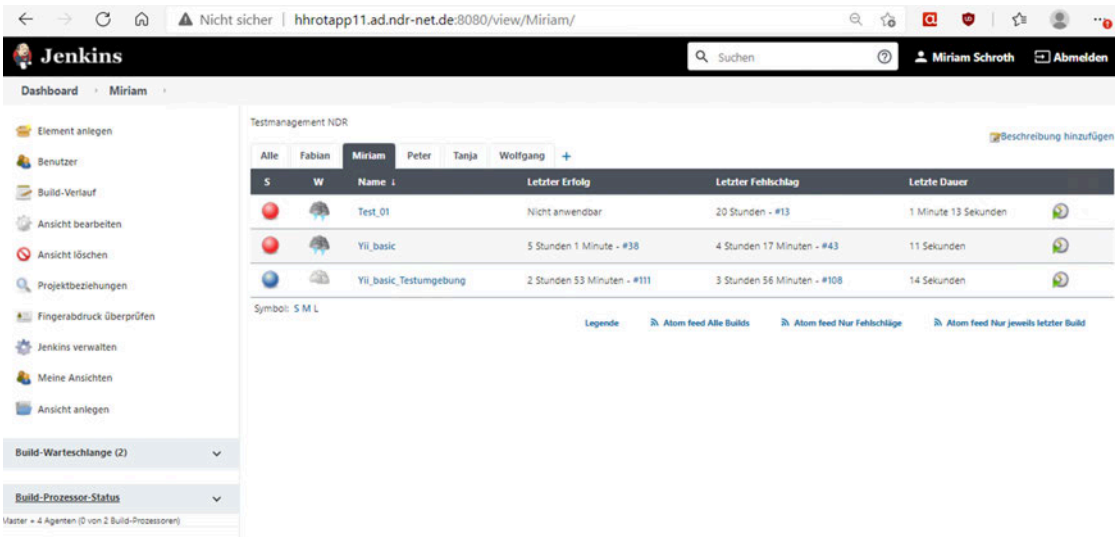


Abbildung 4.4: Jenkins: Webseite

Die Abbildung zeigt das Dashboard eines Users mit einer Übersicht aller Jobs dieses Users. Zu jedem Job werden Informationen in Spalten angezeigt. Die Spaten gliedern sich in den Status vom letzten Build, Bericht über die letzten Builds, den Job-Namen, letzten Erfolg und Fehlschlag, sowie die Dauer des letzten erfolgreichen Builds.
Auf der linken Seite ist die Navigation mit allen Optionen die auswählt werden können zu sehen. Über Element Anlegen wird ein neuer Job in Jenkins konfiguriert und angelegt. Für die Anzeige eines Jobs wird der Name von diesem ausgewählt. Im Anschluss wird der Job mit Arbeitsbereich, der Build History und Optionen angezeigt. Es ist möglich die Konfiguration und den Namen des Jobs zu ändern. In Build History kann ein Build auswählt werden, um die Konsolenausgabe von diesem zu analysieren.

Konfiguration eines Jenkins Jobs Die Abbildung 4.5 zeigt die Ansicht, die bei der Auswahl vom Element Anlegen im Dashboard erscheint. Der Job-Name wird eingegeben und der Job-Typ auswählt. Für die Einbindung von Ranorex-Testsuite wird das ‘Free Style‘ Softwareprojekt bauen ausgewählt.

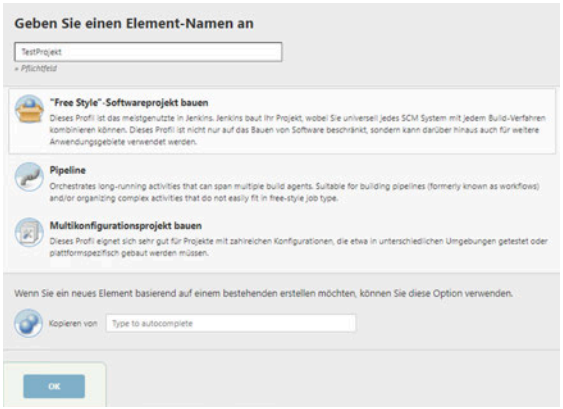


Abbildung 4.5: Jenkins: Neuer Job

Es öffnet sich die Konfiguration des Jobs, die in mehrere Bereiche unterteilt ist. Die Abbildung 4.6 zeigt den Bereich General mit der Beschreibung des Jobs und einigen Optionen.

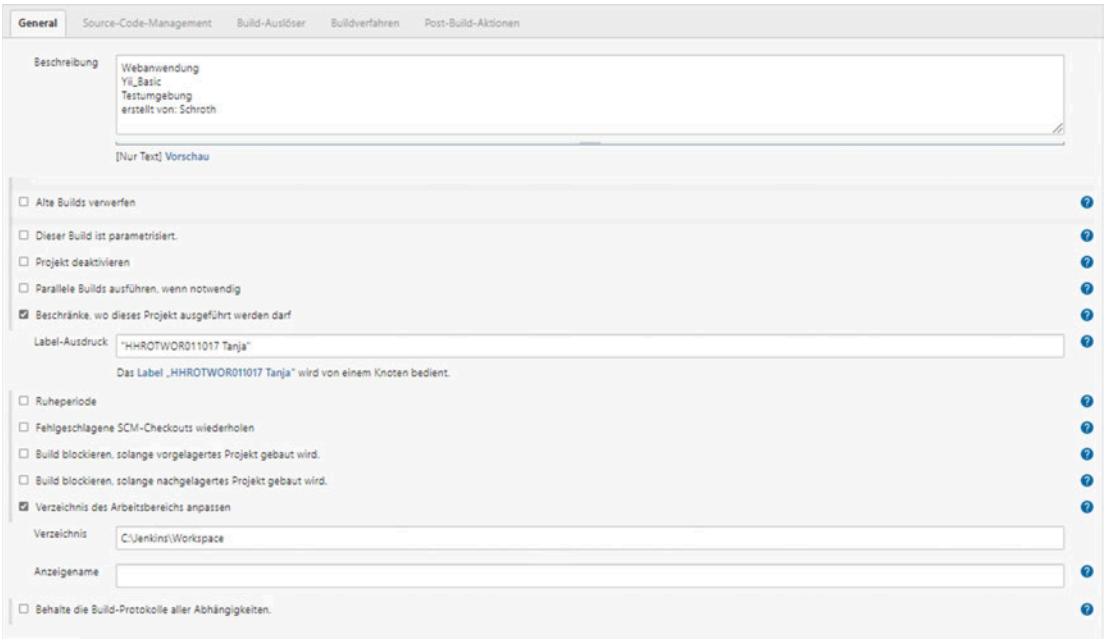


Abbildung 4.6: Jenkins Job: General

Über ‘Beschränke, wo dieses Projekt ausgeführt werden darf ‘ erfolgt die Zuordnung eines bestimmten Rechners (Knoten) auf den Jenkins über einen Agenten zugreift. Der Pfad für das Arbeitsverzeichnis auf dem Rechner wird festgelegt, um Jenkins den Zugriff zu ermöglichen.

Im Bereich Build-Auslöser in Abbildung 4.7 ist es möglich den Zeitpunkt und einen Auslöser, z.B. über ein Skript für die Ausführung des Jobs festzulegen.

Abbildung 4.7: Jenkins Job: Build-Auslöser

In den Buildverfahren werden Build-Schritte hinzugefügt. Für die Einbindung und Ausführung eines Ranorex-Testsuite wird der Schritt ‘Run a Ranorex Test Suite‘ auswählt. Die Abbildungen 4.8 zeigt mögliche Konfigurationen.

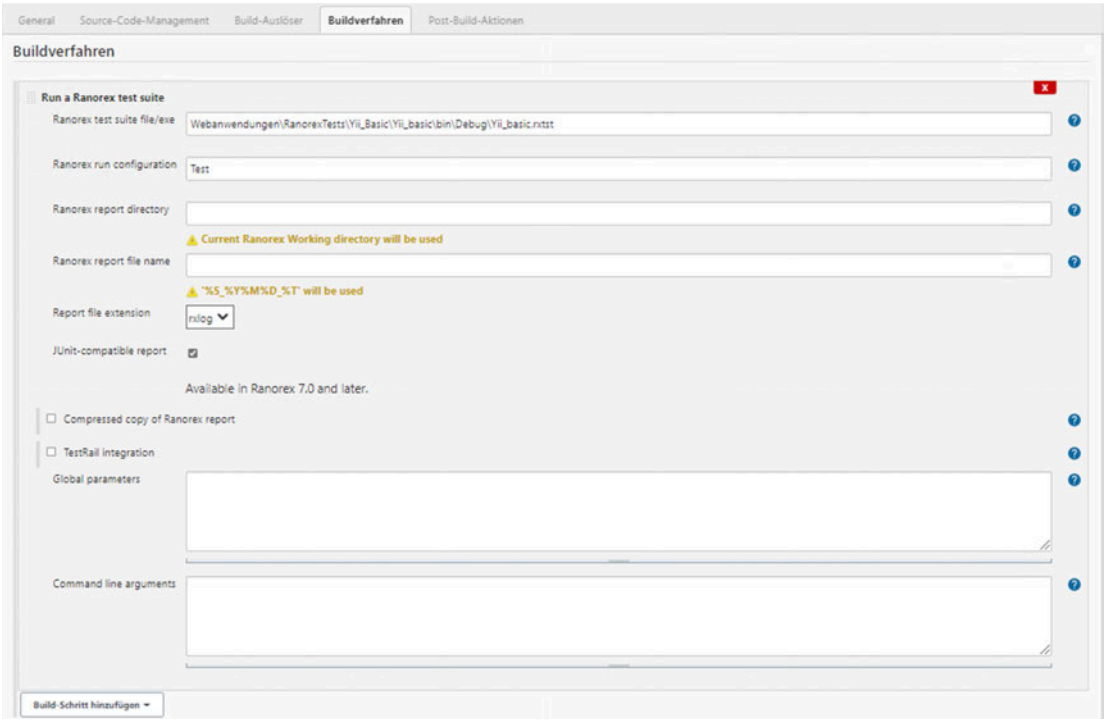


Abbildung 4.8: Jenkins Job: Buildverfahren

Zuerst wird der Pfad für die Testsuite-Datei (*.rxtst) oder (*.exe) eingegeben. Dabei ist drauf zu achten, dass sich diese im Ausgabeordner befindet, womit der Unterordner des Testsuites `bin\Debug` gemeint ist. Unter der Run-Konfiguration wird im Standard die derzeit ausgewählte Konfiguration in Ranorex ausgeführt. Es ist möglich eine Run Konfiguration zu setzen und diese für den Job explizit in Ranorex zu konfigurieren. Erfolgt keine Angabe eines Ranorex Report Verzeichnises wird der Pfad verwendet in dem

sich die auszuführende Datei (Testsuite) befindet. Im Standard hat der Report den Dateinamen des Testsuites und das Format in html oder rxlog. Über die Option JUnit wird zu dem Ranorex-Report ein kompatibler JUnit-Report erstellt. Über die Auswahl Komprimierung Report ist es möglich den Report mit allen Dateien in ein separates Archiv zu speichern. Die Ergebnisse können an die API TestRail gemeldet werden. Globalen Parametern im Testsuite von Ranorex können Werte zugewiesen oder überschreiben werden. Außerdem können zusätzlich Befehlszeilenargumente für den Testsuite eingegeben werden. Eine Übersicht über alle Befehlszeilenargumente [14] sind im User Guide im Unterkapitel Ranorex Studio expert zu finden.

Über Post-Build-Aktionen ist es möglich Benachrichtigungen über Mails zu generieren, siehe Abbildung 4.9.

Nach Abschluss der Konfiguration wird der Job entweder automatisch über einen Auslöser, nach einer bestimmten Zeit oder manuell ausgeführt.

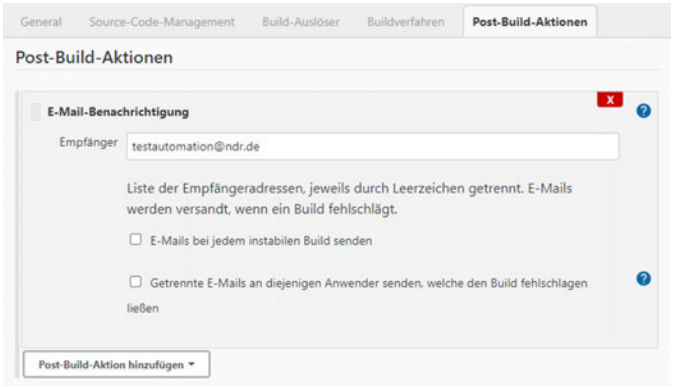


Abbildung 4.9: Jenkins Job: Post-Build-Aktionen

Beispiel einer Jenkins Job-Build Konsolenausgabe

Im Build Jobs kann die Konsolenausgabe angezeigt werden. In Abbildung 4.10 ist ein Teil einer Konsolenausgabe zu sehen. Für die Erklärung wurde die Abbildung farblich gekennzeichnet. Im rosa gekennzeichneten Abschnitt ist das Run-System auf den der Job ausgeführt wird zu sehen. Im hellgrünen sind die Jobkonfigurationen. Ab dem roten Abschnitt wird der Ranorex-Testsuite ausgeführt, welcher mit dem starten der Logging-Konsole beginnt. Der blaue Abschnitt zeigt den Report vom Open-Browser. Im letzten braunen Abschnitt ist ein Teil des Logins zu sehen. In den letzten beiden Abschnitten sind nach dem Start des Testmoduls die definierten Variablen mit den konfigurierten Werten des Moduls zu sehen. Zur Sicherheit sind sensible Daten, wie Passwörter und Benutzernamen unkenntlich.

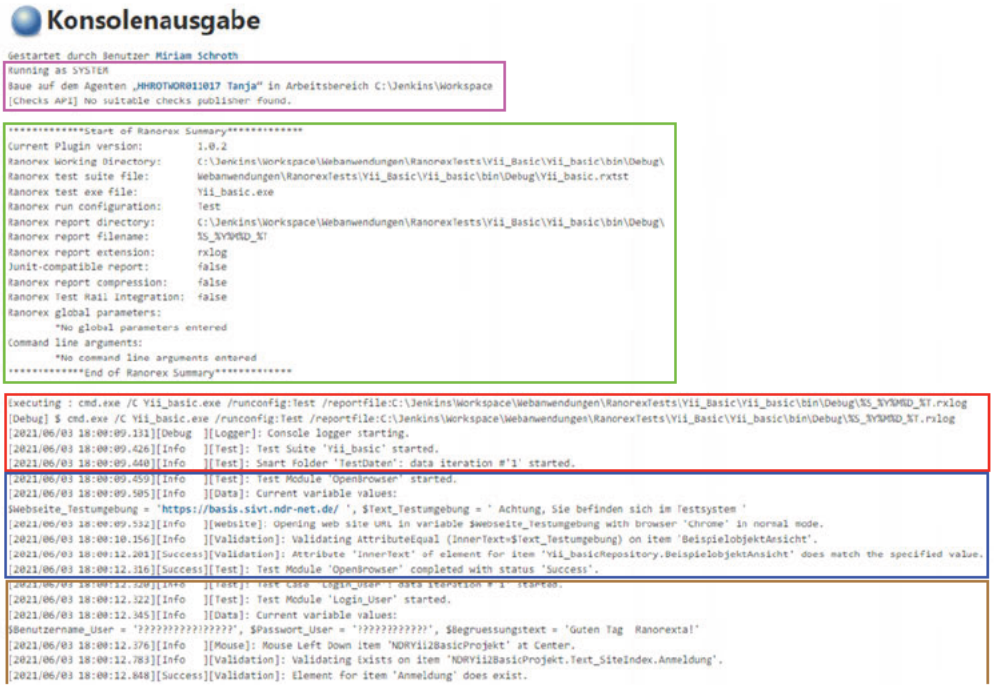


Abbildung 4.10: Jenkins: Konsolenausgabe eines Job-Build

5 Evaluation

In diesem Kapitel erfolgt die Evaluation der Arbeit. Im Ersten Teil werden die Anforderungen an das Testkonzept evaluiert. Im zweiten Teil werden technische Probleme und Entscheidungen, die bei der Umsetzung auftaten charakterisiert. Zum Schluss erfolgt ein Fazit der Arbeit.

5.1 Evaluation der Anforderungen

Die nachfolgende Übersicht zeigt, dass fast alle Anforderungen an das Testkonzept erfolgreich umgesetzt wurden. FA steht in der Tabelle für funktionale Anforderungen und NFA für nicht funktionale Anforderungen.

ID	**Anforderung**	**Kapitel**	**Erfüllt**
1 FA	*Testumfang*	4.2.4	✓
2 FA	*Teststruktur*	4.2.2, 4.2.4	✓
3 FA	*Datenbasis*	A.1.5, A.1.4	✓
4 FA	*Browser*	4.2.7	✗
5 NFA	*Erweiterbarkeit*	4.2	✓
6 NFA	*Wiederholbar*	4.3	✓
7 FA	*Automatisierte Tests*	4.2	✓
8 FA	*Testdaten*	4.2.6, A.1.5	✓
9 FA	*Steuerung und Überwachung*	4.3	✓
10 FA	*Feedback*	4.3	✓
11 FA	*Versionsverwaltung*	4.1	✓

Die Anforderung ‘4 FA *Browser*‘ wurde im Rahmen der Arbeit teilweise umgesetzt. Die Umsetzung der Tests mit dem Browser Edge und Safari waren nicht möglich. Auf den Browser Safari muss leider verzichtet werden, da es diesen nicht in der aktuellen Version

für Windows 10 gibt. Der Ranorex-Spy findet im Edge die Felder auf dem Bildschirm nicht, diese werden als ein ganzes Objekt erfasst.

5.2 Charakterisierung von Problemen und Entscheidungen

Bei der Umsetzung des Testkonzeptes in Ranorex und Jenkins wurden einige Probleme festgestellt und Entscheidungen getroffen. Diese Teilen sich in mehrere Punkte auf die nun einzeln erläutert werden.

- Struktur der Solutions
- Anpassungen bei Änderungen der Webanwendung
- Funktion Löschen in der Webanwendung

Struktur der Solutions
Ranorex schlägt für die Umsetzung von automatisierten Test drei grundlegende Teststrukturen vor. Es erfolgt eine Erläuterung warum im Testkonzept zwei Solutions verwendet wurden und wie das mit Keywordgesteuerten Tests zusammenhängt.
Grundsätzlich verfügen die Webanwendung für die Test- und Produktivumgebung jeweils eine eigene URL. Die automatisierten Tests können in Ranorex in einer Solution zwei getrennten oder einer Umgebung/en getestet werden. Ranorex erzeugt für den Projekttyp 'Testsuite-Projekt' automatisch eine Exe-Datei. In Jenkins wird für die Überwachung und Steuerung von automatischen Tests in einem Job die Testsuite aus einem Projekt in Ranorex integriert, siehe Kapitel 4.3. Bei der Ausführung des Jobs in Jenkins wird die Exe-Datei, die einer Testsuite zugeordnet ist, automatisch ausgeführt, siehe Abbildung 4.10 im Kapitel 4.3.

In der Solution-Struktur werden die Tests in verschiedenen Solutions erstellt. Bei Webanwendungen bekommen die Test- und Produktivumgebungen dadurch jeweils eine eigene Solution. Die Prod-Solution wird dabei auf Basis der Test-Solution erstellt. Im Repository der Prod-Solution ermöglicht eine Änderung der DOM (Document Object Model) es die Objekte einer anderen URL zuzuordnen. Durch die Trennung der Umgebungen in zwei getrennte Solutions existieren für die Ausführung in Jenkins zwei Exe-Dateien. Zu beachten ist, dass bei Änderungen oder Erweiterungen zu einem späteren Zeitpunkt die Testfälle in beiden Solutions angepasst werden müssen. Durch die Teilung in zwei verschiedene Solutions ist immer eine Anpassung der URL in beiden notwendig und es können Probleme durch die Anpassung der DOM in der Prod-Solution entstehen.

Bei der Projektbasierten- und Testsuiten-Struktur befinden sich die Tests für die Test- und Produktivumgebung in einer Solution. Der unterschied hierbei ist, dass sich bei der Testsuiten-Struktur in einem Projekt zwei Testsuiten befinden. Bei der Projektbasierten Struktur werden zwei Projekte mit jeweils einer Testsuite erstellt. In beiden Fällen ist es möglich Module aus der anderen Testsuite zu verwenden. Das Öffnen und Schließen der Browser muss durch die verschiedenen URLs der Webanwendungen mit unterschiedlichen Modulen erfolgen.
In der Testsuiten-Struktur existiert nur eine Exe-Datei. Jenkins kann bei der Ausführung der Tests die Testsuiten keiner Exe-Datei zuzuordnen und eine Fehlermeldung in der Jenkins Konsole erscheint. Diese gibt an, dass die Testsequenz mehrere aktivierte Testsuite enthält. Aus diesem Grund ist die Umsetzung der Tests in einem Projekt nicht möglich.
Bei der Projektbasierten-Struktur ist zu erwarten, dass in den Repositories für die verschiedenen Webseiten jeweils eine DOM von Ranorex-Spy erzeugt wird. Werden nun in der Testumgebung Tests erstellt, so können diese nicht in der Produktivumgebung verwendet werden. Es ist zu ermitteln, wie die DOM verändert werden kann um für beide Webseiten gültig zu sein. Ein Vorteil wäre, dass in den Modulen auf die gleichen Repository-Objekte verglichen wird und so die Module in beiden Umgebungen verwendet werden können.

Bei der Umsetzung wurden zwei Solutions verwendet. Diese gliedern sich in die Solution für das Testen der Webanwendung und eine für DLL-Module. Für beide Solution wurde die Projektbasierten-Struktur gewählt.
Die erste Solution beinhaltet die Tests für die Test- und Produktivumgebung in jeweils einem eigenständigen ‘Testsuite-Projekt‘. Diese Umsetzung wurde gewählt, da Module aus dem anderen Projekt einfach eingebunden werden können.
In der zweiten Solution werden die Projekte in unterschiedliche Funktionen aufgeteilt, welche die Keywordgesteuerten DLL-Module enthalten, siehe Kapitel 4.2.3. DLL-Module können in verschiedenen Solutions, Projekten oder Testsuiten zu verwenden werden, da sie keine URL Bindung haben und somit keiner DOM zugeordnet sind.

Anpassungen bei Änderungen der Webanwendung
Während der Erstellung der Tests in Ranorex wurde der Login von Benutzern in der Webanwendung geändert. Diese führte zu einer Anpassung der HTML-Seite, wodurch die HTML-Objekte zusätzliche Lehrzeichnen oder komplett andere Namen hatten. In den automatisierten Tests musste die Pfade der Objekte über dem Ranorex-Spy im Objekt-Repository angepasst werden. Durch Validierungen mit Variablen erfolgte eine zusätzliche Änderungen in den Testdaten.

Funktion Löschen in der Webanwendung
Bei der Umsetzung der automatisierten Tests in Ranorex wurde beim Löschen das angelegten Beispielobjektes nicht auf das vorgelagertes Pop-Up Fenster verglichen, das beim Aufruf von Löschen erscheint. Der Ranorex Spy kann diese Objekt nicht finden. Für die Umsetzung wurde daher Ok über die Eingabetaste auf der Tastatur bestätigt, welches zum Erfolg führte. Diese Umsetzung ist nicht schön, andererseits zweckmäßig. Im Rahmen einer Erweiterung sollte eine Anpassung erfolgen. Der User Guide von Ranorex beschreibt, wie mit Hilfe der 'PopupWatcherLibrary' und der Klasse 'PopupWatcher' auf Pop-Up Fenster reagiert und zugegriffen werden kann.

5.3 Fazit

Es ist zu empfehlen automatisierte Tests für alle Webanwendungen im NDR einzuführen damit Fehler automatisiert erkannt werden. Das Testkonzept kann hierbei als Orientierung und Vorlage dienen.

Die Fachgruppe IT-Service im NDR muss entscheiden für welche Webanwendungen automatisierte Test notwendig sind. Eine Priorisierung erleichtert die Entscheidung in welchem Umfang und welcher Reihenfolge die Webanwendungen automatisierte Tests erhalten.

Zu Beginn sollten für alle Webanwendungen in der Test und Produktivumgebung nur das An- und Abmelden eines Benutzers umgesetzt werden. Mit diesem einfachen Test wird neben dem An- und Abmelden, die grundsätzliche Verfügbarkeit einer Webanwendung in beiden Umgebungen geprüft.
Um die DLL-Module für das An- und Abmelden aus dem Modulbibliotheks-Projekt in allen Webanwendungen verwenden zu können müssen der Code und die Einlogdaten in allen Webanwendungen identisch sein.

Für die Erstellung von Testfällen ist das Vorgehen aus dem Design angebracht, da dieses im Testkonzept erfolgreich umgesetzt wurde. Eine Umsetzung mit Keywordgesteuerten Tests ist zu empfehlen, da diese in beiden Umgebungen und gegebenenfalls auch in anderen Webanwendungen verwendet werden können.

Bei Testerweiterungen können Testfälle in der gleichen Testsuite in einem Projekt erstellt werden. Über die Run Konfigurationen in Ranorex können die Testfälle in unterschiedlichen Jobs in Jenkins eingebunden werden.

Zu beachten ist, dass automatisierte Tests gepflegt werden müssen sobald eine Änderung oder Erweiterung erfolgt. Der Zeitaufwand ist nicht zu unterschätzen, da sich die HTML-Seite mit den Objekten ändert und die bestehenden Tests angepasst werden müssen.

Abschließend lässt sich sagen, dass sich Ranorex für die Umsetzung von automatisierten Tests für Webanwendungen eignet. Die Steuerung und Überwachung der Tests aus Jenkins ist sinnvoll und erfolgreich.

6 Zusammenfassung

Ziel der Arbeit war es, ein Testkonzept für automatisierte Tests von internen Webanwendungen im NDR zu entwickeln. Diese sollten mit Ranorex erstellt werden und die Überwachung und Steuerung sollte über Jenkins erfolgen.
Dabei wurden zum Verständnis die Grundlagen und die Situation im Norddeutschen Rundfunk beschrieben. Für das Konzept wurde die jetzige Konfiguration, Installation und Zugriffsmöglichkeiten der Tools Ranorex und Jenkins im NDR, sowie deren grundlegenden Eigenschaften und Funktionen für die Umsetzung der Tests mit Webanwendungen beschrieben. Aus diesen Informationen wurden die Anforderungen an das Testkonzept abgeleitet und umgesetzt. Im Anschluss erfolgte eine Evaluierung der umgesetzten Anforderungen, sowie eine Auflistung von Probleme und Entscheidungen die bei der Umsetzung getroffen wurden.

Im Rahmen der Arbeit wurde gezeigt, dass sich automatisierte Tests für die Webanwendung ‘NDR Yii_basic‘ mit Ranorex erstellen und durch Jenkins steuern und überwachen lassen.

6.1 Ausblick

Eine Erweiterung des Testkonzeptes ist möglich. Im folgenden wird beschrieben welche Erweiterungen für automatisierte Tests in Ranorex sinnvoll sind. Es ist sinnvoll das Testkonzept um einen Continuous Integration (CI) und Continuous Deployment (CD) Prozess zu erweitern.

Erweiterung der automatisierten Tests in Ranorex

Im Rahmen einer Erweiterung sollte die Umsetzung im Edge Browser erfolgen, nachdem das bestehende Problem mit der Objekterfassung mit diesem behoben wurde. Der Safari

Browser lässt sich wohl testen, wenn in Ranorex das Konzept Mobile Anwendungen zusätzlich umgesetzt wird. Die Realisierung des Löschens von angelegten Beispielobjekten sollte langfristig angepasst werden. So können Pop-up Fenster, die oft in den Webanwendungen im NDR auftreten, vernünftig getestet werden.
Bei Erweiterungen der Tests ist es sinnvoll die Keywordgesteuerten Test beizubehalten, da diese in verschieden Ranorex Tests verwendet werden können.

Umsetzung von CI und CD

Um alle Webanwendungen automatisiert testen zu können ist es sinnvoll einen Prozess für CI und CD einzuführen. Das Testkonzept sieht eine Überwachung und Steuerung von automatisierten Tests vor, allerdings wird nicht definiert in welchem Umfang und wie oft die Tests ausgeführt werden sollen. Um einen CI-Prozess in Jenkins umsetzen zu können muss ein Arbeitsablauf erstellt werden. Für diesen müssen zuvor die technischen Randbedingungen ermittelt werden. Nachfolgend werden Detailfragen zu den Bereichen des Testumfanges, Testausführung, Randbedingungen und den Arbeitsablauf beschreiben. Weiter ist es sinnvoll einen CD-Prozess einzuführen, der den CI-Prozess erweitert. Die derzeitige Development-Chain sollte dazu in Jenkins integriert werden, Möglichkeiten für eine Umsetzung wenden aufgezeigt.

Testumfang
In welchem Umfang sollen die automatisierten Tests erfolgen?
Smoke-, Sanity- oder Funktions-Tests?
Wie viele Tests beinhaltet ein Job in Jenkins?

Testausführung
Ist eine parallele Ausführung der Tests in Jenkins und Ranorex möglich?
Können parallel mehrere Jobs in Jenkins über einen oder mehrere Agenten auf dem gleichen Knoten ausgeführt werden?
Welche Zeit benötigt ein Job in Jenkins für die Ausführung und wovon ist diese abhängig?

Randbedingungen
Ist eine Umsetzung mit der Distributed Build Architektur von Jenkins möglich oder muss eine Erweiterung durch mehr Agenten erfolgen?
Können auf dem Rechner auf dem die Tests erstellt werden auch die Jenkins Jobs ausgeführt werden oder ist eine Trennung notwendig?

Können die Tests weiterhin auf einem Standard Windows-PC umgesetzt werden?
Verfügt der NDR über genug Lizenzen für Ranorex für eine Umsetzung?
Welche Kosten entstehen für weitere Lizenzen?
Welche Kosten entstehen bei der Automatisierung der Tests für eine oder mehrere Webanwendungen?

Arbeitsablauf

Ist es möglich den Bereitstellungsprozesses und das Testen der Webanwendungen in der Entwicklung vollständig zu automatisieren?

Development-Chain

In der jetzigen Development-Chain werden die Webanwendungen über Skripten auf die Test- und Produktivumgebung deployed. Es ist sinnvoll diesen Prozess in Jenkins zu integrieren indem ein vollautomatischer Arbeitsablauf erstellt wird. Nach einem Merge in Bitbucket in den Test- oder Prod-Branch erfolgt ein automatisches Deploy auf die jeweilige Umgebung mit anschließenden automatisierten Tests und einem Feedback.
Für diese Umsetzung muss Apache als Server installiert sein und eine Verbindung über SSH zum Jenkins CI-Server bestehen. Jenkins holt den Code aus Bitbucket, build diesen und stellt ihn anschließend im Verzeichnis des Apache-Servers bereit. Über die SSH Verbindung wird der Bereitstellungsprozess für den der Apache-Server ausgelöst. [3]
Ob diese Art für das Deployment von unterschiedlichen vhost geeignet ist, muss individuell getestet werden.
Für die Integration und Kommunikation von Jenkins und Bitbucket stehen einige Plugins zur Verfügung die mit einer Beschreibung in der Tabelle 6.1 aufgelistet sind.
Im aktuell verwendeten Jenkins ist es im Moment nicht möglich ‘Multibranch-Pipeline‘ zu erstellen. Diese erlauben es mehrere Jenkins-Dateien für unterschiedliche Zweige eines Projektes zu implementieren. Eine Jenkins-Datei enthält eine Versionskontrolle, wie Git oder Bitbucket und wird mit einer einfachen Pipeline erstellt. Im ‘Multibranch-Pipeline‘ erkennt, verwaltet und führt Jenkins automatisch Pipelines für Zweige aus.

Plugin Name	**Beschreibung**
Bitbucket	Die Integration zwischen Bitbucket und Jenkins wird über einen einzelnen Endpunkt ermöglicht. Durch ein Push in einem Projekt empfängt Jenkins Daten und führt ein Job auf Basis der Änderungen aus.
Bitbucket Build Status Notifier	In der Bitbucket-Benutzeroberfläche wird angezeigt ob der Build erfolgreich ist.

Bitbucket Branch Source	Ermöglicht die Nutzung von Bitbucket als Multi-Branch-Projektquelle auf zwei Arten. Bei einem Team-/Projektordner werden für jedes sichtbare Repository in einem Team oder Projekt automatisch Multi-Branch-Projekten erstellt. Ist die Quelle ein einzelnes Repository wird für jeden Zweig und Pull Request in diesem Repository ein Job in Jenkins erstellt. Für diese Realisierung sind in Jenkins die 'Multibranch Pipeline' erforderlich.
Bitbucket Filter Project Trait	Erweiterung des 'Bitbucket Branch Source' Plugins um zwei Filter für Team Projekte. Die Filterung erfolgt über einen Schlüssel oder den Namen mit einem regulären Ausdruck. Für alle Projekte die im Filter entahlten sind können in Jenkins Jobs erstellt werden.
Bitbucket Push and Pull Request	Builds in Jenkins werden nach Push- und Pull-Requests-Events ausgelöst. Bei gleichzeitiger Verwendung mit dem 'Bitbucket' Plugin sind zusätzliche Einstellungen erforderlich, da beide auf den gleichen Endpunkt zugreifen.
Bitbucket Pullrequest Builder	Es erweitert das 'Bitbucket Push and Pull Request' Plugin um eine Checkbox, die ein Pull-Request erstellt und das Ergebnis von diesem an Jenkins übermittelt.
Bitbucket Pull Requests filter	Filterung der Pull-Request nach Feldern mit bestimmten Ausdrücken. Jenkins berücksichtigt nur die Pull-Requests, mit dem regulären Ausdruck übereinstimmen oder defilierte Wörter enthalten.

Tabelle 6.1: Bitbucket Plugins die mit Jenkins interagieren können

Literaturverzeichnis

[1] *Unterstützung namensbasierter virtueller Hosts*. URL: https://httpd.apache.org/docs/2.2/de/vhosts/name-based.html. – accessed: 2021-05-07

[2] *Apache-Dokumentation zu virtuellen Hosts*. URL: https://httpd.apache.org/docs/2.2/de/vhosts/name-based.html. – accessed: 2021-05-07

[3] *Apache2 als Deployment-Server und Jenkins als CI-Server*. URL: https://devops4solutions.com/ci-cd-jenkins-php/. – accessed: 2021-07-27

[4] Witter, Frank: *Basiswissen Softwaretest*. dpubkt.verlag, 2020. – ISBN 978-3-86490-583-4

[5] *Bitbucket 2020*. URL: https://bitbucket.org/product/de/. – accessed: 2020-12-12

[6] Buth, Bettina: *Folien zur Vorlesung Certified Tester*. HAW, WiSe 2020/21

[7] *Git 2020*. URL: https://www.atlassian.com/git. – Zugriffsdatum: 2020-12-12

[8] *Jenkins 2020*. URL: https://www.jenkins.io. – accessed: 2020-12-12

[9] *Integrate Automated Testing Into Jenkins*. URL: https://www.ranorex.com/blog/integrating-ranorex-automation-in-jenkins-continuous-integration-process/. – accessed: 2021-06-08

[10] *Jenkins, Distributed builds* . URL: https://wiki.jenkins.io/display/JENKINS/Distributed+builds. – accessed: 2021-07-23

[11] *Bitbucket 2020*. URL: https://www.atlassian.com/de/software/jira. – accessed: 2020-12-12

[12] *Step 4: Review code changes with a pull request.* URL: https://bitbucket.org/product/de/guides/basics/four-starting-steps#step-4-review-code-changes-with-pull-requests. – accessed: 2021-05-15

[13] *Ranorex.* URL: https://www.ranorex.com/de/. – accessed: 2020-12-12

[14] *Ranorex: Command line execution.* URL: https://www.ranorex.com/help/latest/ranorex-studio-expert/runtime-and-remote-execution/command-line-execution/. – accessed: 2021-06-08

[15] *Git Grundbegriffe: Working Tree (Sandbox).* URL: http://gitbu.ch/ch01.html. – accessed: 2021-05-29

[16] *XAMPP.* URL: https://www.apachefriends.org/de/index.html. – accessed: 2021-05-12

[17] *Yii-Framework.* URL: https://www.yiiframework.com. – accessed: 2021-05-29

A Anhang

A.1 Handlungsanweisungen für Ranorex

Dieser Teil umfasst Handlungsanweisungen die bei der Realisierung der automatisierten Test in Ranorex wichtig sind und dort in der Umsetzung nicht in allen Einzelheiten beschrieben werden. Diese umfasst die folgenden Inhalte:

- Fokussierung
- Vermeidung von Meldungen
- Referenzierung auf DLL-Dateien
- Globale Parameter & Validierungen
- Einbindung von Testdaten

A.1.1 Fokussierung

Für die Fokussierung wird in der Whitelist in Abbildung A.1 für die Aufnahmen über den Recoder der Prozess vom Chrome Browser integriert. Die anderen Prozesse sind bei gleichzeitigem öffnen der Browser zu entfernen um die Aufnahme von Aktionen aus diesen Prozessen zu vermeiden.

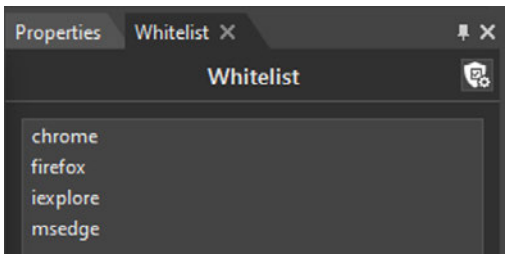


Abbildung A.1: Whitelist

A.1.2 Vermeidung von Meldungen

Um Meldungen bei der Ausführung der automatisierten Test zu vermeiden muss in allen Modulen in dem ein Browser geöffnet wird eine Eigenschaft angepasst werden. In der

Abbildung A.2 ist das Modul ‘OpenBrowser‘ mit der Aktion ‘Open browser‘ zu sehen. Für diese muss die Eingenschaft Instrument auf False gesetzt werden.

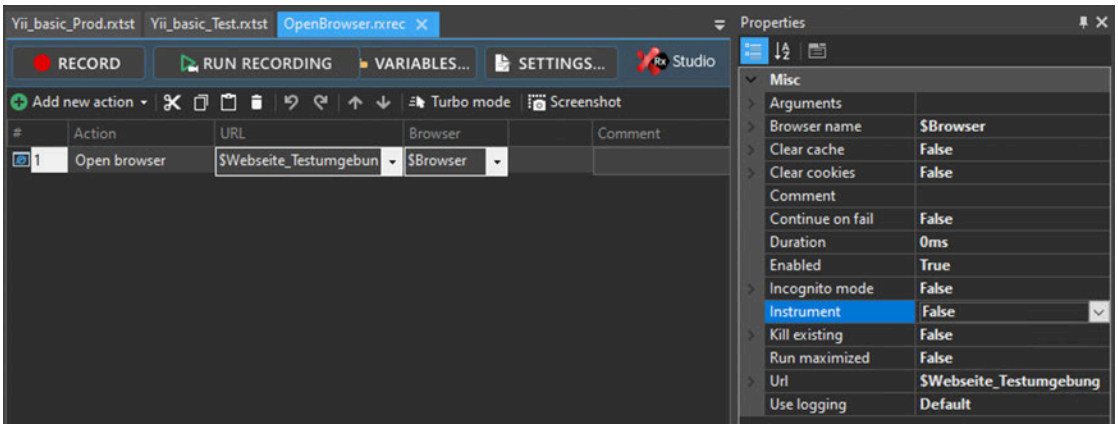


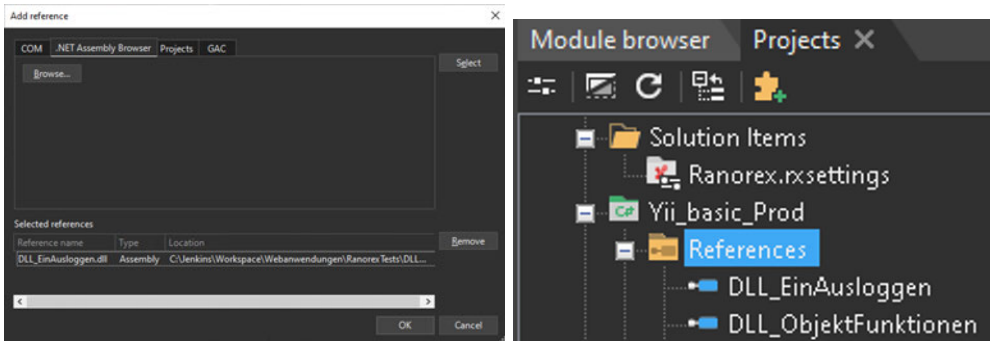
Abbildung A.2: Modul ‘OpenBrowser‘

A.1.3 Referenzierung auf DLL-Dateien

In Projekten der Test- und Produktivumgebung wird auf die DLL-Dateien Referenziert. Für die Umsetzung muss in der Projektansicht ‘add refernces‘ ausgewählt werden. Im Anschluss ist der Reiter ‘.Net Assembly Browser‘ im Dialogfenster auszuwählen. Die Abbildung A.3a zeigt das Dialogfenster mit einer hinzugefügten Referenz im Bereich ‘select refernces‘. Nach Bestätigung der Eingabe sind die Referenzen unter ‘Refernces‘ in der Projektansicht zu sehen, siehe Abbildung A.3b. Werden DLL-Dateien im Modulbibliotheks-Projekt verändert, so müssen diese neu gebaut (rebuild) und im referenzierten Projekt aktualisiert werden.

A.1.4 Globale Parameter & Validierungen

Hinter den Webanwendungen liegt eine Datenbank, die beim Erzeugen von neuen Objekten eine neue ID vergibt. Wird nun in einem Test ein neues Objekt erzeugt, ist es sinnvoll die ID zu speichern und alle weiteren Operationen auf diesem Objekt auszuführen. Für die Realisierung wird die dynamische ID in einer globalen Variable des Testfalles gespeichert. Bei einem Vergleich oder einer Suche wird immer die ID verwendet die diesem bei der Erstellung zugeordnet wird. In der Abbildung A.4 ist zu sehen, dass eine ID in eine Variable über ‘Get value‘ gespeichert wird. Das Wait darüber ist notwendig, um zu



(a) Einstellungen Referenzen (b) DLL-Referenzen in Prod-Projekt

Abbildung A.3: Referenzierung auf DLL-Module

garantieren, dass eine Seite neu geladen wird.

Mit Validierungen ist es möglich die gespeicherte ID mit der aktuellen Objekt-ID zu

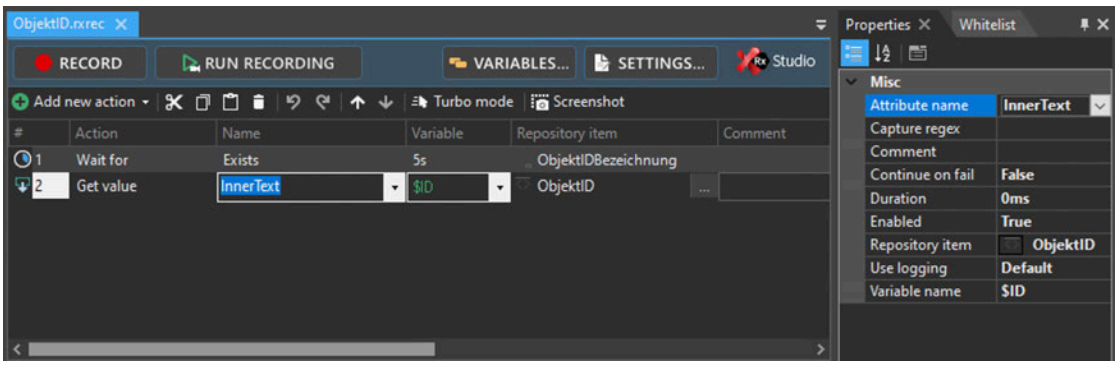


Abbildung A.4: DLL-Modul ‘ObjektID‘

vergleichen. In der Abbildung A.5 im rosa Rahmen ist diese Validierung umgesetzt. Der Vergleich erfolgt über den inneren Text eines Objekts mit einer Variablen. In den Eigenschaften muss für ‘Match name‘ das Validierungs-Attribut und für den ‘Match value‘ der Übereinstimmungswert auf den verglichen wird eingetragen sein (roter Rahmen). Der ‘Match value‘ ist eine Variable die mit dem globalen Parameter verknüpft ist. Der Vergleich erfolgt auf einem Objekt, dem ‘Repository item‘. In grün wird das Objekt im Objekt-Reposetory hervorgehoben. Im Pfad von diesem ist zu sehen, dass das Validierungs-Attribut mit der selben Variable verglichen werden muss.

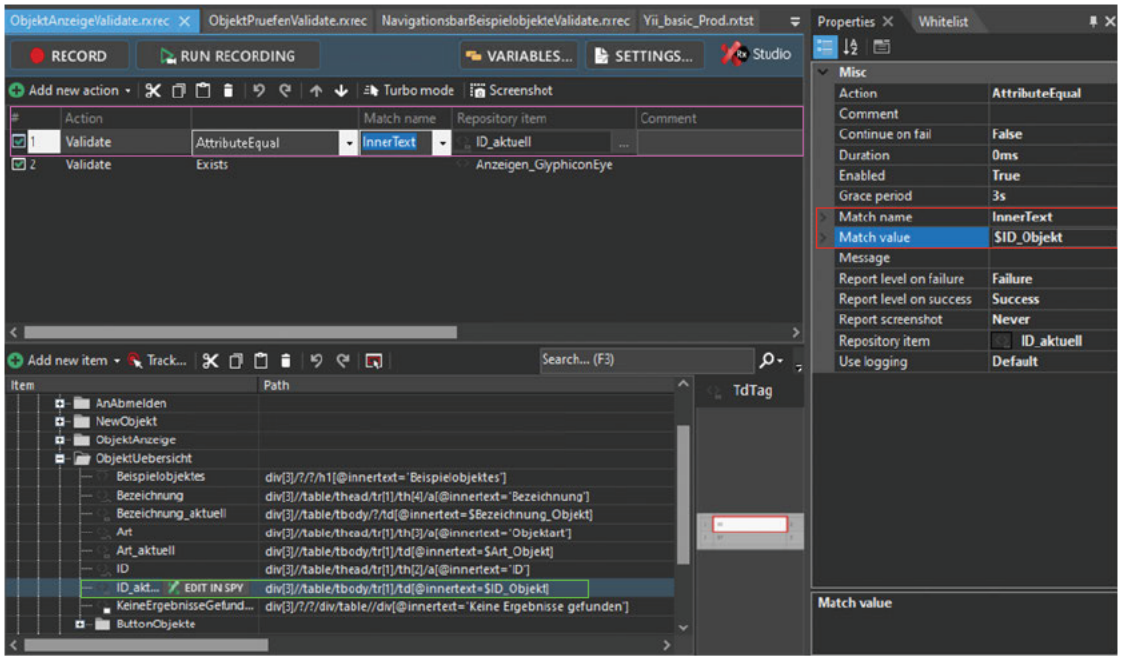


Abbildung A.5: Validierung

A.1.5 Einbindung von Testdaten

Beim Einbinden von Testdaten können bestimmte Einstellungen vorgenommen werden, diese werden nun weiter erläutert. In der Abbildung A.6 ist zu sehen, dass drei CSV-Dateien in einer Testsuite eingebunden wurden. Die Erklärung erfolgt an den 'LoginDaten'. Im Bereich 'Data source config' wird für die Daten ein Name gewählt und der Pfad für diese eingegeben. Die Datenquelle ins Ausgabeverzeichnis des Projektes zu kopieren ist sinnvoll, wenn dieses nicht sensibel oder geheim ist. Die Daten können dadurch einfach über Git verteilt werden, wenn kein Zugriff auf die Original Daten besteht. Mit den Optionen 'Use header' wird festgelegt, ob die CSV-datei einen Header hat. Beim 'Autoload data when test sute opens' wird ermittelt wie viele Zeilen die CSV-Datei beim laden der Testsuite hat. Diese werden als ROWs in der Testsuite angezeigt. Über die Option 'Mask' werden sensible oder geheime Daten im Report maskiert. Das Label wird bei der Umsetzung nicht gesetzt.

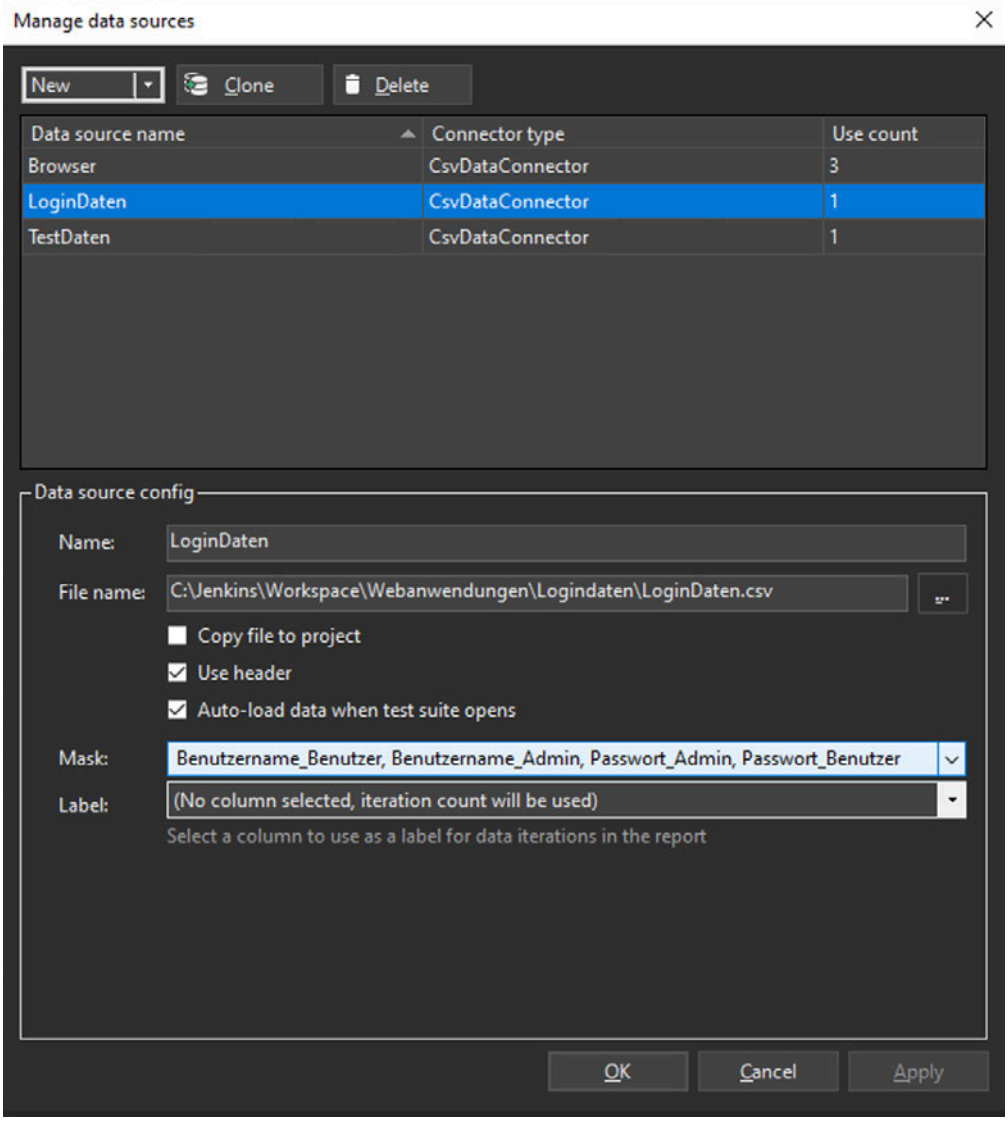


Abbildung A.6: Management der Testdaten

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Testkonzept für automatisierte Tests von internen Webanwendung mit Ranorex und Jenkins

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original