|  |  |
| --- | --- |
|  |  |
| Titel: | **Konzept für die Testautomatisierung**  **des Service Center Systems** |
| Verantwortlicher Autor: | Marc Siegmund (COQA)  marc.siegmund@six-group.com |

|  |  |
| --- | --- |
| Dokumentart: | Konzept |
| Dokument erstellt am: | 08.05.2009 |
| Letzte Veränderung: |  |
| Version: | 1.5 |
| Status: | REVIEW |

**Inhaltsverzeichnis**

1 Überblick 3

1.1 Zielbestimmung und Abgrenzung 3

1.2 Referenzen 3

1.3 Dokumenthistorie 4

1.4 Liste offener Punkte 4

2 Organisation und Planung 5

2.1 Vorgehen 5

2.2 Phasen 5

2.3 Voraussetzung für das automatisierte Testen 6

2.4 Vor- und Nachteile der Testautomatisierung 6

3 Aufgaben im Testprozesse 7

3.1 Erstellung eines Testfalls 7

3.1.1 Testfälle im QC erstellen 7

3.1.2 Testfälle mit QTP erstellen 7

3.2 Testdatenerstellung 7

3.3 Testskripterstellung 7

3.4 Testkonfiguration 8

3.5 Testausführung 8

3.5.1 Ausführen eines automatischen Tests aus QTP 8

3.5.2 Ausführen eines automatischen Tests aus dem QC 8

3.6 Testauswertung 8

3.7 Testdokumentation 8

3.8 Testadministration 9

4 Typische Anwendungsfälle 10

5 Testobjekte 12

5.1 SCS – Web-Anwendung (GUI) 12

5.2 Analyzer – Web-Anwendung (GUI) 12

5.3 Datenbank 12

5.4 Protokolle 13

5.5 Klassen, Module, Komponenten 13

6 Automatisierbare Testarten 14

6.1 Modul-/ Unit-/ Komponententests 14

6.2 Integrationstests 14

6.3 Smoketests 14

6.4 Funktionstests 14

6.5 Systemtests 14

6.6 Regressionstests 14

6.7 Performance- / Lasttests 14

7 Werkzeuge für das Testen 15

7.1 Quality Center 15

7.2 Werkzeuge für die Automatisierung von Tests 15

7.2.1 Quicktest Professional 15

7.2.2 Selenium 16

7.3 Simulatoren / Generatoren 16

8 Komponenten des Testsystems 17

8.1 SCS – Test-Workstation 17

8.1.1 Hardware 18

8.1.2 Software auf dem Test-Host 18

8.2 Infrastruktur 18

8.3 Netzwerkzugriffe 18

9 Abschliessende Bemerkung 19

10 Anhang 20

Software für Test-Workstation 20

# Überblick

Dieses Dokument ist ein Konzept für die Automatisierung von Tests für das **Service Center System** bzw. **SCS** der **S I X Card Solutions**. Das SCS ist eine firmeninterne Software-Anwendung zum Verwalten von Stammdaten der Kunden, Terminals, Standorten, Kontakten und Logins mit entsprechenden Berechtigungen. Die Komplexität des SCS nimmt kontinuierlich zu und der Testumfang wächst mit der Entwicklung der Software.

Mit Hilfe der Testautomatisierung lassen sich metrische Daten ermitteln, die es ermöglichen soll, jeden Software-Release qualitativ zu bewerten.

Bislang werden Tests manuell ausgeführt. Der Testumfang wächst mit jedem Release und Regressionstests sind bislang kaum oder nur unzureichend möglich.

Die Einführung von automatisierten Testfällen soll einerseits das Testen erleichtern und andererseits den Testumfang ausweiten und somit Kosten für das Testen senken. Es lässt sich ein deutlich grösserer Funktionsumfang prüfen und beliebig oft wiederholen. Bei konstanter Genauigkeit und minimaler Fehlerrate bei der Testausführung bei einer hohen Testabdeckung können automatisierte Tests die Qualität der Software präzise und detailliert analysieren.

## Zielbestimmung und Abgrenzung

Dieses Dokument ist ein Konzept für die Automatisierung.

In Anlehnung an das Testkonzept für das SCS und der Berücksichtigung der vorgegebenen Prozesse sollen Aufgaben beim Testen des SCS automatisiert werden.

Ziel ist vor allem die Automatisieren der Testausführungen von Funktions-, Regressions- und Systemtests. Weitere Aufgaben, u.a. die Erstellung von Testdaten, die sich ebenfalls automatisieren lassen, sind im Konzept berücksichtigt.

## Referenzen

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Bezeichnung** | **Version / Datum** |
| [R1] | [„Standard glossary of terms used in Software Testing”](file:///\\\\base.dom\\Daten\\zuerich\\share\\QUALITY_ASSURANCE\\96_COQA\\TEST-Automatisierung\\SCS\\istqb%20glossart.pdf)  International Software Testing Qualifications Board (ISTQB)  Erik van Veenendaal (The Netherlands) | 2.0  December, 2nd 2007 |
| [R2] | [„SCA – Qualitätssicherungskonzept“](../Richtlinien/QS-Konzept/SCA-Qualitätssicherungskonzept_germanV1.4.doc)  SIX Card Solutions  Roland Wehrli | 04.11.2008 |
| [R3] | [Projektplan für Testautomatisierung](file:///\\base.dom\Daten\zuerich\share\QUALITY_ASSURANCE\96_COQA\TEST-Automatisierung\Projektplan%20für%20Testautomatisierung.mpp)  SIX Card Solutions  Marc Siegmund | Jeweils aktuell |
| [R4] | [Qualitätssicherungskonzept](file:///S:\QUALITY_ASSURANCE\96_COQA\TEST-Automatisierung\Richtlinien\QS-Handbuch\QS%20Handbuch_v0.3.dot)  SIX Card Solutions  TKOLT | W0.2 |
| [R5] | Allgemeines Testkonzept für das Service Center System  SIX Card Solutions  Jörg Blaser | ? |
| [R6] | [Test-Konzept](Test-Konzept-SCS.docx)  SIX Card Solutions  Marc Siegmund |  |
|  |  |  |
|  |  |  |

## Dokumenthistorie

|  |  |  |
| --- | --- | --- |
| **Datum** | **Was** | **Wer** |
| 30.06.2009 | Review | Marc Siegmund |
| 03.07.2009 | Review  Aufteilung in separate Dokumente, Anpassung der Struktur und Inhalte | Steffen Page |
| 03.07.2009 | Überarbeitung (Status: DESIGN) | Marc Siegmund |
| 06.07.2009 | Freigabe zum Review (Status: Review) | Marc Siegmund |
| 08.07.2009 | Überarbeitung und Besprechung im Team-Meeting | Team |
| 09.07.2009 | Einfügen von Verweisen zu den Refrenz-Dokumenten | Marc Siegmund |
|  |  |  |
|  |  |  |
| ??.??.2009 | Abnahme des Dokuments | Daniel Von Allmen |

## Liste offener Punkte

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Was** | **Wer** | **Datum** |
| [OP1] | Zertifikat anstatt SecurId | SYR |  |
| [OP2] | Ziele und Umfang vom Einsatz eines Roboters für Terminaltests sind Unklar | SYR, VOD |  |

# Organisation und Planung

Dieser Abschnitt beschreibt die organisatorischen Aspekte für die Test-Automatisierung des SCS.

Die (Projekt-)Planung für die Testautomatisierung seitens SCS erfolgt gemeinsam mit der Planung für die Testautomatisierung im Bereich „Saferpay“ in einer separaten [MS-Project-Datei](../Projektplan%20für%20Testautomatisierung.mpp).[R3]

Aktuelle Pendenzen sind in den Terminen und Aufgaben von Outlook aufgeführt.

## Vorgehen

Die Automatisierung wird zunächst in Phasen aufgeteilt (siehe Abschnitt 2.2).

Regelmässige Team-Sitzungen überprüfen den Verlauf der Automatisierung und dienen zum:

* Besprechen aktueller Punkte (Probleme, Aufgaben, organisatorische Dinge)
* Aufzeigen des Fortschritts in der Automatisierung
* Zielsetzung der Automatisierung
* Informationsaustausch innerhalb der Abteilung
* Sammeln neuer Ideen

## Phasen

Das Vorgehen wird in folgende Phasen unterteilt. Der detaillierte und aktuelle Plan ist in der [MS-Project-Datei enthalten](file:///\\\\base.dom\\Daten\\zuerich\\share\\QUALITY_ASSURANCE\\96_COQA\\TEST-Automatisierung\\Projektplan%20für%20Testautomatisierung.mpp).

Abbildung 1 Phasen für die Automatisierung

* Phase 1 (ab 2009)
  1. Quartal Aufbau von Prozesskenntnissen und fachlichen Kenntnisse über das SCS,

Team in das Quality Center einführen und schulen

* 1. Quartal Aufbau der Testumgebung mit Installation der Test-Workstation

Quality Center optimiert einsetzen

* 1. Quartal QTP-Einführung, Beginn der Testautomatisierung – sie ist die Explorationsphase zum Aufzeigen von Möglichkeiten der Automatisierung
  2. Quartal Realisierung von funktionalen Tests
* Phase 2 (ab 2010)
  1. Quartal Erstellung von Regressionstests auf Basis der funktionalen Tests
  2. Quartal Beginn Schnittstellen-Tests
* Phase 3

Ab 2010/11 Terminal-Tests mit Roboter

## Voraussetzung für das automatisierte Testen

Eine vollständige und in sich schlüssige Spezifikation ist für die Testautomatisierung essentiell. Anforderungen müssen präzise definiert werden, um ein anforderungsbasiertes Testen zu ermöglichen.

Die grundlegende Voraussetzung für das automatisierte Testen ist ein bestehendes Test-Konzept für das SCS. Die Prozesse müssen für die Qualitätssicherung müssen klar mit Angabe der verantwortlichen Personen definiert sein.

Das Priorisieren von Testfällen ergibt sich zum einen aus der Priorität der Anforderung oder zum Anderen (bei Regressionstests) aus dem Bedarf von Tests innerhalb der QA. Anhand dieser Vorgaben werden Testfälle automatisiert.

Das Vorhandensein einer stabilen Infrastruktur ist für die Automatisierung notwendig, da sonst die Auswertung von Testfällen, den Nutzen zunichte machen. Die Erweiterung der Testumgebung durch die Test-Workstation wird im Kapitel 8 beschrieben und wird für die Automatisierung vorausgesetzt.

Teile der Automatisierung erfordern ein Zertifikat, um automatische Zugriffe (speziell das Login) zu ermöglichen. Alternativ muss der Tester sich manuell mit der SecurId autorisieren.

## Vor- und Nachteile der Testautomatisierung

Testen an sich erfordert einen erhöhten Aufwand, der sich in der frühzeitigen Erkennung Fehler auszahlt.

Ähnlich verhält sich die Automatisierung der Tests. Zunächst ist ein grösserer Aufwand notwendig, der sich im Laufe der Zeit auszahlt, wenn die Tests häufig (wieder)verwendet werden.

Abbildung Vor- und Nachteile der Automatisierung

# Aufgaben im Testprozesse

Der Aspekt der Automatisierung beschränkt sich nicht nur auf die Ausführung von Tests. Die Testdatenerstellung, das Auswerten von Daten (aus z.B. Log-Files) oder das setzen von Konfigurationen bieten gute Möglichkeiten zur Automatisierung. Ziel ist die Automatisierung möglichst vieler Aufgaben in der Qualitätssicherung.

Die folgenden Unterabschnitte beschreiben wichtige Aufgaben im Testprozess. Es wird beschrieben, ob und wie sich die jeweilige Aufgabe automatisieren lässt bzw. der Nutzen im Verhältnis zum Aufwand gegeben ist.

## Erstellung eines Testfalls

Ein Testfall beschreibt den Ablauf eines Tests durch eine Sequenz von Aktionen mit genauen Erwartungen an das Resultat.

Alle Testfälle sollen im QC zentral abgelegt sein.

Aus bestehenden Dokumenten (Excel- oder Word-Dokumente) können Testfälle automatisiert ins QC importiert werden.

Eine weitere Möglichkeit ist die Erstellung anhand bestehender Protokolldaten, um z.B. das exakte Verhalten aus dem Live-Betrieb in der Testumgebung „nachzuspielen“.

Denkbar wäre eine automatische Generierung von Testfällen. Durch ein zu entwickelndes Tool könnten beispielsweise individuelle Regressionstests durch Angabe von Kriterien erstellt werden.

### Testfälle im QC erstellen

Testfälle lassen sich im QC über die GUI erstellen. Die Automatisierung bietet an dieser Stelle wenig Nutzen, da in erster Linie manuelle Tests erstellt werden.

Viele Testfälle sind bisher u.a. in Word-Dokumenten beschrieben. Der Import von Testfällen aus diesen bestehenden Dokumenten liesse sich automatisieren und komplett ins QC portieren. Die Automatisierung des Imports ist ein interessanter Punkt, um den Umstieg von einer grossen Anzahl an herkömmlichen Dokumenten zur zentralen Plattform QC zu erleichtern.

### Testfälle mit QTP erstellen

Per „Capture & Replay“ lassen sich beispielsweise mit QTP aufzeichnen. Diese sind dann entweder lokal im Dateisystem oder zentral im Quality Center abgelegt.

Der vorrangige Weg QTP einzusetzen ist eine abstrahierte Programmierung der Testfälle (Key-Word-Driven Testing).

Die Testfälle lassen sich lokal im Dateisystem ablegen oder direkt zum Testplan vom QC übertragen

## Testdatenerstellung

Testdaten können anhand von Regeln automatisiert generiert werden. Dafür sind Generatoren (siehe Abschnitt 7.3) notwendig, die Test-Daten erstellen.

Mit QTP enthält bereits einen Generator, mit dem sich durch Angabe regulärer Ausdrücke Testdaten generieren lassen.

## Testskripterstellung

Ein Testskript ist der Programmcode für einen automatischen Ablauf eines Testfalls.

Die generische Erstellung von Testskripts ist möglich jedoch nicht geplant, da der Nutzen nicht im Verhältnis zum Aufwand steht ist.

Für Modultests bieten SDKs das Generieren von Testskripte, die anschliessend vom Programmierer (manuell) implementiert werden müssen.

QTP erzeugt beim Capture & Replay das Script, während der Benutzer den Test durchführt. Testobjekte werden automatisch gelernt, doch dabei sind generell manuelle Nachbearbeitungen notwendig, um diese wiederverwendbar aufzubereiten.

## Testkonfiguration

Testumgebungen könnten per Skript automatisch für die notwendige Testausführung angepasst werden.

Für das SCS ist beispielsweise die Konfiguration der Netzadressen für die unterschiedlichen Testumgebung ein wesentlicher Punkt. Desweiteren könnten gewisse Sätze an Testdaten für unterschiedliche Testzwecke vorkonfiguriert werden, die somit die Vorbedingung für das Testen schaffen..

## Testausführung

Die Ausführung ist der Schwerpunkt der Automatisierung. Software-Tests sind genau und lassen sich beliebig oft wiederholen.

QTP ist das zentrale Werkzeug zur automatischen Ausführung von Tests.

### Ausführen eines automatischen Tests aus QTP

Der Test kann über das QC oder vom Dateisystem geladen werden.

Der Test lässt sich auf dem lokalen System ausführen. Im QTP wird während des Testlaufs die aktuelle Position im Test-Skript markiert.

### Ausführen eines automatischen Tests aus dem QC

Automatische Tests, die im QC abgelegt sind, lassen sich von dort aus starten. Der Test selbst kann lokal oder entfernt ausgeführt werden. QTP muss auf dem jeweiligen System, auf dem der Test läuft, installiert sein.

## Testauswertung

Je nach Test, kann eine abschliessende Auswertung aus einem oder mehreren Testläufen folgen. Für Last und Performance-Tests kann die Aufbereitung der Testergebnisse erleichtert werden.

Wichtige Kennzahlen, die sich aus ermittelten Daten ergeben, lassen sich automatisch berechnen oder als Diagramm darstellen. Der Tester erhält somit aufbereitete Daten mit prägnanten Metriken über das SCS.

Das QC ermöglicht Auswertungen aus Sicht des Qualitätsprozesses und das Testmanagement. Individuelle Auswertungen beziehen sich auf den Test und die Verarbeitung der ermittelten Daten während des Laufs (z.B. Fehlerrate, Antwortzeiten).

* Die Möglichkeiten zur Auswertung sind im QC sehr umfangreich,  
  einige Beispiele sind:
  + Statistiken lassen sich generieren
    - Anzahl Testläufe
    - Anzahl gefundener Fehler
    - Dauer für einzelne oder alle Testläufe
    - Prozentuale Abdeckung der Anforderungen durch Tests
    - Risikobewertung
  + Reports lassen sich generieren
    - Auflistung von Testläufen mit Status und Findings
    - Auflistung von gefundenen Findings zu einer Anforderung
  + Export in Dokumente (Word, Excel, XML, …)
* Automatisierung von Auswertungen
  + Auswertung von Logfiles
  + Auswertungen in der DB

## Testdokumentation

QTP Tests übermitteln (automatisch) die Testergebnisse mit den Informationen aus den einzelnen Schritten einer Test-Instanz an das QC zum entsprechenden Test-Set.

Alternativ kann der Testreport lokal auf dem Dateisystem erstellt werden. Dafür muss der Test aus dem QTP (siehe Abschnitt 3.5.1) gestartet werden.

Bei Verwendung eigener Test-Treiber ist der Zugriff auf das QC eine API entfernt möglich. Die Funktionalität steht der über der QC-GUI nichts nach.

## Testadministration

# Typische Anwendungsfälle

Abbildung Anwendungsfälle

In Abbildung 3 sind die typischen Anwendungsfälle für das SCS aufgegliedert und dient als Übersicht für die Funktionalität. Diese sollen ein grundlegender Bestandteil für die Testautomatisierung sein.

Die Funktionalität ist deutlich umfangreicher, denn die Anwendungsfälle für z.B. die Administration, das Ticketing sind lediglich berücksichtigt. Je nach Priorität, Risiko und Komplexität werden die Anwendungsfälle als Test realisiert und automatisiert.

# Testobjekte

Dieses Kapitel befasst sich mit den Testobjekten für das SCS in Hinblick auf die Automatisierung. In den Unterabschnitten sind die Testobjekte aufgeführt.

## SCS – Web-Anwendung (GUI)

Die GUI setzt sich aus einer Menge an **Webseiten** zusammenund bildet das Frontend für den Benutzer. Jede GUI bzw. Webseite wird durch ein Testobjekt repräsentiert. Die wichtigsten Testobjekte sind:

* Anmelde-Seite
* Menu-Seite (ist im SCS und im Zebra als Frame in die Seite integriert)
* Such-Seite
* Anmelde-Seite
* Kundendetail-Seite
* Standortdetail-Seite
* Terminaldetail-Seite
* Benutzerdetail-Seite

Diese Testobjekte bieten den grössten Nutzen, da sie sich hervorragend für **Smoke-, Funktions-, System- und Regressionstests** eignen.

Der Benutzer des SCS wird simuliert. Deshalb liegt verstärkt der Fokus in der Modellierung genau dieser Testobjekte. QTP unterstützt GUI-Tests sehr professionell und effizient durch Capture & Replay, durch Inspektion der GUI-Elemente sowie die automatische Erstellung des Test-Skripts.

Die Anmelde-Sicht spielt dabei eine besondere Rolle, da ein Proxy (der WES-Server) den Zugriff kontrolliert.

## Analyzer – Web-Anwendung (GUI)

„Analyzer“ ist das Frontend zum Überwachen und zum Protokollieren der meisten Aktivitäten des SCS.

Analyzer ist interessant für die Validierung von Tests. Somit nimmt Analyzer einerseits die Rolle als Testobjekt und andererseits als Werkzeug zur Validierung von Ein- und Ausgaben beim Testen ein.

Analyzer bietet sich ebenso für die Überwachung der Testumgebung an, da Dienste, Fehlerzustände und weitere Daten über die Systemumgebung gesammelt werden. Bei fehlgeschlagenen Testläufen lässt sich die Ursache bereits mit Hilfe von Analyzer automatisch eingrenzen. Sollte ein für den Test erforderlicher Dienst inaktiv sein, könnte das Testergebnis automatisch annulliert werden.

Jede Ansicht stellt ein Testobjekt dar. Relevante Testobjekte sind:

* Event Log-Seite
* Alerts-Seite
* Statistics-Seite
* Queue Status-Seite
* Watchdog-Seite

Der Zugriff ist mit Zertifikat möglich und lässt sich dadurch automatisieren.

## Datenbank

Die Datenbank ist zum Validieren der Ein- und Ausgaben relevant. In der DB lässt sich überprüfen, ob das SCS Daten korrekt verarbeitet und ablegt.

Der Zugriff auf die DB-Instanzen ist intern und extern nur mittels VPN-Verbindung und SecurId möglich. Ob Zertifikate als Alternative funktionieren werden ist derzeit unklar und vorerst nicht vorgesehen. Die Prüfung auf Datenbank-Ebene ist von extern nicht automatisiert möglich.

Der Bedarf für eine Validierung auf DB-Ebene ist (vorerst) nicht Kernpunkt. Darum werden DB-Testobjekte zunächst nicht realisiert und nicht beschrieben.

Analyzer (siehe Abschnitt 4.2) kann stattdessen zur Überprüfung verwendet werden kann. [BE3]

## Protokolle

Die Realisierung von Testobjekten und Testfällen für Schnittstellen und Protokolle wird nach der Etablierung des Testsystems angestrebt. [B5]

Folgende Protokolle sind für das automatische Testen von Bedeutung:

* EP2:SI-CONFIG
* EP2:SI-INIT
* SCS-FO
* COI (Fax, Email, File)

Die der Protokolle verlangt das Parsen und Validieren von Messages. Dafür muss das bestehende Konzept ergänzt werden und die Voraussetzungen (Simulatoren) erfüllt sein.

## Klassen, Module, Komponenten

Testobjekte sind einzelne Klassen, Module oder Komponenten. Sie werden aufgrund der Vollständigkeit erwähnt. Es ist nicht Ziel, diese Testobjekte seitens der QA zu prüfen, da Modultests Aufgabe der Software-Entwickler sein/bleiben, die mit Hilfe von Testtreibern ihre Module prüfen.

# Automatisierbare Testarten

Dieser Abschnitt stellt die Testarten vor, die für die Automatisierung relevant sind.

## Modul-/ Unit-/ Komponententests

Modultests lassen sich durch Unit-Tests automatisiert testen. Da es sich um White-Box Tests handelt, die Kenntnisse über das Innenleben der Software sowie der Programmiersprache erfordern, sollen diese Tests durch die Entwickler erstellt werden.

## Integrationstests

Das Zusammenspiel der Schnittstellen und Komponenten kann durch automatisierte Tests geprüft werden. Da die Komponenten durch die Entwicklung integriert wird, was Programmierkenntnisse sowie Kenntnisse über das Innenleben der Software voraussetzt, sollten diese Tests durch die Entwicklung analog zu den Modultests erfolgen.

## Smoketests

Auf minimalen Level prüfen Smoketests neue und bestehende Anforderungen.

Für die neuen Anforderungen sind automatisierte Tests wiederzuverwenden. Oft werden neue Anforderungen neue Testfälle erfordern und somit die Entwicklung neuer automatisierter Tests verlangen.

Bestehende (unveränderte) Anforderungen lassen sich durch einen enorm abgespeckten Regressionstest auf Funktionalität überprüfen.

## Funktionstests

Funktionale Tests stehen im Vordergrund der Automatisierung.

Die Funktionstests werden mittels QTP „End-to-End“ direkt an der SCS-Web-GUI durchgeführt. Das entspricht dem Verhalten eines Endbenutzers.

## Systemtests

Systemtests sind nicht trivial, da automatisierte Tests für alle im Test betroffenen Systeme vorhanden sein müssen, damit der Anwendungsfall komplett automatisiert durchgeführt werden kann.

Das SCS wird auf korrekte Interoperation mit den umgebenden Systemen wie: Front Office, Pass, Topas/PMS, Terminals verifiziert.

## Regressionstests

Regressionstest sind eine Verkettung von ausgewählten Funktionstests und Systemtests, die mit wenig Aufwand einen grossen Teil der Funktionalität prüfen, ohne jeden Fall zu berücksichtigen.

Eine Sammlung von automatisierten Testfällen zu einem Regressionstest gebündelt prüfen, ob die bisherige Funktionalität gewährleistet ist. Die Testautomatisierung muss für eine Auswahl an Funktionstests vorhanden sein, die sich parametrisierbar zu einer Sequenz von Testfällen als eine Szenario zusammensetzen lassen. Ein Regressionstest kann darüberhinaus auch nicht funktionale Testfälle beinhalten.

Die automatisierten Funktionstest müssen einen gewissen Reifegrad erreicht haben.

## Performance- / Lasttests

Besonders gut lassen sich Performance- bzw. Lasttests automatisieren, da eine hohe Anzahl an wiederkehrenden Abläufen durchgeführt wird.

In einer Testumgebung entsprechen die ermittelten Statistiken meist nicht die der Produktivumgebung, da aus Kostengründen nicht die gleiche Performance zur Verfügung steht. Deshalb sind die Ergebnisse kritisch zu hinterfragen. Dennoch sind Rückschlüsse auf das Verhalten der Software möglich. Es lässt sich beispielsweise erkennen, wie sehr eine neue oder geänderte Komponente das System beeinflusst, welche Komponente bei hoher Last instabil wird und wo ihre Grenze (Maximum, Minimum) liegt oder den „Flaschenhals“ im Softwaresystem lokalisieren.

# Werkzeuge für das Testen

Für das Testen kommen die in den folgenden Unterabschnitten aufgeführten Werkzeuge zum Einsatz. Diese bieten eine Verwaltung des Qualitäts-Prozess, des Testen oder unterstützen bei der Analyse der Software-Anwendung.

## Quality Center

Das Quality Center (QC) wird als zentrales **Testmanagement**-Tool eingesetzt. Die Software unterstützt die Verwaltung von:

* Releases
* Anforderungen (sogar mit Risiko-Analyse)
* Tests (im Testplan)
* Ausführbare Testinstanzen (in Test-Sets)
* Findings (/ Defects bzw. Abweichungen zur Spezifikation)

Anforderungen lassen sich spezifizieren anhand deren sich der Testplan mit Testfällen ableiten lässt.

Testsets bündeln je nach Fokus bzw. Thema für das Testen eine Menge an Testinstanzen. Testinstanzen lassen sich entweder manuell oder automatisch ausführen.

Die Protokollierung erfolgt automatisch. Findings lassen sich am besten während des Testen erstellen, dann ist die Entstehung eines Findings plausibel und nachvollziehbar. Somit sind Reports aktuell und zentral verfügbar. Eine Vielzahl an Informationen (z.B. Testresultate, Testabdeckung oder Testfortschritt) lassen sich auf diverse und anpassbare Arten (z.B. Diagramm, Report) darstellen.

## Werkzeuge für die Automatisierung von Tests

Für die Testautomatisierung sind Anwendungen verfügbar, die in den Unterabschnitten vorgestellten Tools wichtig. Dieser Abschnitt wird bei Bedarf um weitere Tools ergänzt.

Dieses Dokument gibt einen Überblick der eingesetzten Tools und verweist auf die entsprechende Dokumentation (intern oder extern – von Drittanbietern).

Die Ansprüche an Werkzeuge für die Automatisierung von Tests sind:

* Erstellen von automatisierten Tests
* Ausführen von automatisierten Tests

### Quicktest Professional

Quicktest Professional (QTP) unterstützt das **„Key-Word-Driven Testing“**.

Testobjekte lassen sich entweder während des Testen inspizieren und „lernen“ oder zu222nächst abstrakt wie eine Schnittstelle spezifizieren. Das ermöglicht die Entwicklung von Tests für noch nicht fertige Software. Wie bei der Objektorientierten Programmierung lassen sich Testobjekte über Methodenaufrufe steuern. Es werden Schlüsselbegriffe verwendet, um aus Sicht des Testens auf die Testobjekte zuzugreifen. Aus der technischen Aktion: Web.Button.Click() würde: Vertrag.Confirm().

Die Methoden sind die „Key-Words“ und vom Testentwickler zu implementierten. Die Methoden bzw. Aktionen lassen sich von Testern ohne Programmierkenntnisse (wieder).

Für das „Key-Word-Driven Testing“ ist ein zunächst grösserer Initialaufwand erforderlich, der sich durch die gründliche Planung auszahlt. Eine stabile Test-Struktur baut sich auf und für das Erstellen von automatisierten Testfällen müssen keine technischen order Programmier-Kenntnisse vorhanden sein.

Quicktest Professional unterstützt unter anderem „Capture and Replay” bei dem der Tester Aktionen aufzeichnet, die sich wiederholt “abspielen” lassen. Solche Testfälle sind benötigen relativ wenig Zeit zur Erstellung

Zu Bemerken ist:

* Lässt sich direkt ins QC einbinden.
* dass QTP nicht nur Web-Anwendungen automatisch ansteuern kann, sondern z.B. auch Windows-GUIs, SAP oder Oracle DBMS.
* QTP eignet sich ausschliesslich für Windows-Plattformen.

### Selenium

Ist eine „open-source“ Lösung, die speziell auf Web-Anwendungen zielt und fertige „Key-Word“-vordefinierte Aktionen bereit hält.

* Eingeschränkt auf Webanwendungen
* Keine Anbindung zum QC, Entwicklungsarbeit ist notwendig

## Simulatoren / Generatoren

**Simulatoren** verhalten sich wie echte System(sub)komponenten. Sie sind notwendig, um unabhängig und gezielt zu testen, da sich das Verhalten explizit steuern lässt. Ein Beispiel ist die Simulation eines physischen Terminals.

**Generatoren** dienen zur Erstellung von Daten, Nachrichten oder sonstigen Testdaten, die für das Testen notwendig sind, um beispielsweise Vorbedingungen von Tests zu setzen oder Systeme gezielt anzusteuern. Als Beispiel ist ein EP2-Nachrichten-Generator, mit dessen Hilfe sich Nachrichten generieren lassen.

Der Einsatz von Simulatoren ist vorerst noch nicht vorgesehen. Diese sind zu berücksichtigen bzw. explizit zu erwähnen, falls die Testautomatisierung von Simulatoren abhängig ist.

# Komponenten des Testsystems

Die Basis der Testsystem wird in den Unterpunkten beschrieben. Das Testsystem ist so ausgelegt, dass es sich leicht portieren lässt.

In Abbildung 4 ist der Aufbau und das Zusammenspiel schematisch dargestellt. Daraus geht hervor, dass die Kopplung der Test-Workstation sehr lose

Das detaillierte Schema aus Abbildung 5 zeigt interne Details, die für die Testentwicklung wichtige sind.



Zertifikat / SecurId

Nevisbox

WES-Server

Test-Workstation

Internet

(externer Netzzugriff)

Web-Browser



Quality Center

SCS

Service Center System

Abbildung Übersicht Testsystem



Abbildung Übersicht der Komponenten des Testsystems [R8]

## SCS – Test-Workstation

Die SCS – Test-Workstation bestehend aus dem Test-Host ermöglicht das manuelle sowie vor allem das automatisierte Testen.

Dieser wird anstelle eines Office-PCs/Notebooks eingesetzt und für sämtliche Testzwecke eingerichtet. Die Unterpunkte präzisieren die Aspekte der Workstation

### Hardware

Folgende Hardware wird benötigt:

* Test-Host / Desktop Computer mit Standard-Peripherie
* Netzwerk-Switch
* Maus/Monitor /Tastatur-Switch
* USB-Stick
* Terminal

### Software auf dem Test-Host

Folgende Software wird benötigt:

* Betriebssysteme (Host-OS sowie Guest-OS)
* Browser im jeweiligen Betriebssystem
* Virtuelle Maschine
* Remote-Access (z.B. Teamviewer, Remote-Desktop, VNC)
* Quicktest Professional
* Entwicklungstools (SDK)
* Zertifikat für den Zugriff auf Applikationen der SIX Card Solutions
* Tools

## Infrastruktur

Die Test-Workstation benötigt einen (firmenexternen) Zugang zum Internet.

Ziel ist es, die Test-Workstation allen zur Verfügung zu stellen, um Tests in einer mit den benötigten Software-Tools und einer leicht zu konfigurierbaren Umgebung zu erstellen oder auszuführen.

## Netzwerkzugriffe

Folgende Netzzugriffe müssen vom Test-Host sowie physisch als auch logisch (Berechtigung, Firewall) sichergestellt werden:

* Quality Center
* Service Center System
  + L-Test-System
  + K- Test-System
  + Produktives Test-System
  + Zebra Test-System

Damit sich Zugriffe automatisieren lassen, ist anstatt der Authentifizierung mittels SecurId ein gültiges Zertifikat vorzuziehen, das im Betriebssystem des Test-Hosts vorhanden sein muss.

# Abschliessende Bemerkung

Die Automatisierung im Qualitätsprozess entlastet Tester und bewirkt ein unermüdliches und exaktes Testen. Die Automatisierung orientiert sich an den Bedarf an Tests, die eine hohe Ausführungsrate haben und somit einen grossen Nutzen bieten. Es gibt Punkte, die sich prinzipiell automatisieren lassen, der Bedarf sowie der Nutzen noch zu überprüfen sind, um darüber eine Entscheidung zu treffen.

Als besonderer Ausblick sei erwähnt, dass zukünftig mit Hilfe eines pneumatischen Roboters im Labor-Umfeld automatisches Testen am physischen Terminal ermöglicht werden soll. Die Anforderungen müssen dafür noch spezifiziert werden.[OP2] Dieser Aspekt ist somit im Konzept vorerst berücksichtigt, erfordert jedoch eine separate Konzeption, da u.a. Echtzeitansprüche, variierendes Reaktionsverhalten und Transaktions-Kontexte berücksichtigt werden müssen, die nicht trivial sind. Prinzipiell sollte ein Testtreiber zur Verfügung stehen, der das Terminal steuert (Interaktionen des Benutzers simuliert) und Ausgaben (Display, Signaltöne) interpretiert.

# Anhang

Software für Test-Workstation

* Betriebssysteme (Host-OS sowie Guest-OS)
  + Windows XP (Lizenz notwendig)
  + Windows Vista (Lizenz notwendig)
  + Windows 7 (Lizenz notwendig)
  + Linux – Kubuntu
  + MAC? (Lizenz notwendig)
* Browser im jeweiligen Betriebssystem (soweit verfügbar)
  + Internet Explorer
  + Opera
  + Firefox
  + Safari
  + Chrome
* Software zur Virtualisierung (VMWare oder VBox)
* Remote-Access (z.B. Teamviewer, Remote-Desktop, VNC)
* Quicktest Professional (lokale Lizenz – z.B. „seat-licence“ oder Unterstützung )
  + Plug-In für das Quality Center
  + Patches
* Entwicklungstools
  + SQL Server Management Studio
  + Netbeans
  + JavaDK
  + Active Perl
  + Cygwin
* Zertifikat für den Zugriff auf Applikationen der SIX Card Solutions
* Tools
  + Process Explorer
  + KeePass
  + OpenOffice
  + Wireshark / Packtyzer
  + FastStone Capture