Thomas Zefferer, Vesna Krnjic, Bernd Zwattendorfer: "Ein virtuelles Testframework für E-Government Komponenten", D-A-CH Security 2011, 2011, pp. 492-503

Ein virtuelles Testframework für E-Government Komponenten

Thomas Zefferer · Vesna Krnjic · Bernd Zwattendorfer

Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie (IAIK) – Technische Universität Graz
{thomas.zefferer, vesna.krnjic, bernd.zwattendorfer}@iaik.tugraz.at

Zusammenfassung

Um ein adäquates Level an Qualität und Sicherheit zu gewährleisten sind im Rahmen der Entwicklung von E-Government Komponenten umfangreiche Tests von besonderer Wichtigkeit. E-Government Komponenten dürfen im Rahmen von Tests jedoch nicht nur isoliert betrachtet, sondern müssen auch im Zusammenspiel mit der Umgebung, in der sie betrieben werden, überprüft werden. Für clientseitige Applikationen ergibt sich dadurch ein beträchtlicher Aufwand, da aufgrund der Pluralität an verfügbaren Betriebssystemen, Browsern oder auch Kartenlesegeräten eine Vielzahl an möglichen Konfigurationen der Umgebung berücksichtigt werden müssen.
Gängige Testtools und Teststrategien sind für E-Government Komponenten oft nur bedingt anwendbar. Dieses Paper skizziert einen alternativen Ansatz basierend auf virtuellen Maschinen und stellt eine konkrete Umsetzung dieses Ansatzes vor. Wir diskutieren Vor- und Nachteile des entwickelten virtuellen Testframeworks und zeigen, dass dieses in der Lage ist, das Testen von E-Government Komponenten signifikant zu erleichtern und damit zu deren Qualität und Sicherheit beizutragen.

1 Einleitung

Im Zuge der professionellen Entwicklung von Software sind speziell für sicherheitskritische Komponenten geeignete Teststrategien zur Gewährleistung vordefinierter Qualitätsstandards unumgänglich. Im Besonderen gilt dies auch für Softwarekomponenten aus dem Bereich des E-Governments. Da von diesen Komponenten sicherheitskritische Funktionen wie beispielsweise elektronische Signatur oder sichere Benutzerauthentifizierung zur Verfügung gestellt werden, ist deren Sicherheit und Integrität von besonderer Relevanz. Daher sind auch ausführliche und umfassende Tests einzelner E-Government Kernkomponenten für den Betrieb einer sicheren und vertrauenswürdigen E-Government Infrastruktur notwendig.

In Österreich stellen beispielsweise die Softwaremodule MOCCA (Modular Open Citizen Card Architecture) [CeOB10] für den Java Applet basierten Zugriff auf Chipkarten (Bürgerkarten), PDF-AS (PDF Amtssignatur) [LePR09] zur Erstellung von bürgerkartenbasierten PDF-Signaturen, oder MOA-ID [LeHP02] für die bürgerkartenbasierte Benutzerauthentifizierung derartige sicherheitskritische Kernkomponenten dar. Aufgrund derer Bedeutung für das österreichische E-Government werden Releases dieser Komponenten in einigen Fällen zusätzlichen externen Sicherheitsüberprüfungen unterzogen um so einen hohen Level an Qualität und damit einen höchstmöglichen Grad an Sicherheit zu gewährleisten.

Während Qualitätsansprüche zentraler und serverseitiger Softwaremodule wie Web-Services relativ effizient erfüllt werden können, stellt sich die Situation für clientseitige Softwarekom-

ponenten meist ungleich komplexer dar. Hier zeigte die Erfahrung, dass unabhängig von der Softwarekomponente selbst auch die Umgebung, in der diese Komponente ausgeführt wird, einen negativen Einfluss auf deren Funktionalität haben kann. Beispielsweise verursachte vor wenigen Monaten ein Update des Java Runtime Environments (JRE) Probleme im Graphical User Interface (GUI) des MOCCA Applets und führte dazu, dass Labels in der GUI unter gewissen Umständen nicht mehr korrekt dargestellt werden konnten. Benutzer waren dadurch nicht mehr in der Lage im Rahmen der Erstellung elektronischer Signaturen die zu signierenden Daten einzusehen. Dadurch stand eine wichtige Funktion im Rahmen der Signaturerstellung nicht mehr zur Verfügung, was negative Auswirkungen auf die Sicherheit des Signaturerstellungsprozesses hatte. Weitere konkrete Beispiele, in denen die Umgebung, in der sicherheitskritische E-Government Komponenten betrieben werden, negative Auswirkungen auf die Sicherheit der eigentlichen Komponente haben kann, werden im Verlauf dieses Papers noch ausführlicher diskutiert.

Die Problematik negativer, unvorhersehbarer Auswirkungen der Umgebung auf sicherheitskritische Softwarekomponenten ist vor allem für clientseitige Software gegeben. Während zentrale, serverseitige Komponenten in einer wohldefinierten und getesteten Umgebung betrieben werden können, sind Annahmen über Clientumgebungen aufgrund der Vielzahl an Betriebssystemen, JREs und Browsern nur schwer zu treffen. Die Pluralität an möglichen Clientsystemen und Konfigurationen wird durch diverse Verbreitungsstatistiken für Betriebssysteme und Web Browser verdeutlicht [NMS11] [BS11]. Verschärft wird diese Problematik zusätzlich durch den Umstand, dass die einzelnen Komponenten der Umgebung (Betriebssystem, Browser, etc.) in der Regel relativ kurzen Updatezyklen unterworfen sind, wodurch sich die Sammlung möglicher Umgebungen und Konfigurationen ständig ändert und erweitert. Darüber hinaus bedingen Änderungen in den Umgebungen unter Umständen Änderungen an den E-Government Komponenten selbst, wodurch sich auch für diese verkürzte Updatezyklen – und somit häufigere Testdurchläufe – ergeben.

Durch die Dynamik und Pluralität möglicher Zielumgebungen gestaltet sich die Durchführung vollständiger Tests clientseitiger Softwarekomponenten komplex und aufwändig. Ein systematischer Ansatz zur Durchführung von Tests in unterschiedlichen Zielumgebungen ist daher unumgänglich. Die Erfahrung zeigte, dass herkömmliche Testframeworks in diesem Zusammenhang nur bedingt einsetzbar sind. Vor allem die für E-Government Komponenten typische Integration der Chipkartenkommunikation stellt in der Praxis oft eine ernstzunehmende Hürde dar und erschwert die Durchführung systematischer Testdurchläufe.

Dieses Paper beschreibt, wie durch den Einsatz eines virtuellen Testframeworks den typischen Problemen, die beim Testen clientseitiger E-Government Softwarekomponenten auftreten können, begegnet werden kann. Das in diesem Paper beschriebene Framework ermöglicht flexible, effiziente und nachvollziehbare Tests von Softwarekomponenten mit unterschiedlichsten virtuellen Systemen und Umgebungen und trägt somit entscheidend zur Qualitätssicherung und damit auch zur Gewährleistung der Sicherheit von E-Government Kernkomponenten bei.

Dieses Paper ist wie folgt aufgebaut. Abschnitt 2 diskutiert die Wichtigkeit von umfassenden Tests von E-Government Komponenten in verschiedenen Umgebungen und definiert Anforderungen für ein entsprechendes Testframework. Abschnitt 3 gibt einen Überblick über existierende Testtools und Strategien und zeigt, dass diese die in Abschnitt 2 definierten Anforderungen meist nur bedingt erfüllen können. In Abschnitt 4 wird schließlich ein alternativer Ansatz vorgestellt, dessen konkrete Umsetzung in Form eines virtuellen Testframeworks gezeigt

und gesammelte Erfahrungen diskutiert. Abschnitt 5 fasst schließlich die wichtigsten Punkte dieses Papers zusammen.

2 Anforderungen von E-Government Komponenten

Elektronische Behördenwege spielen im Zeitalter des Internets eine zunehmend wichtige Rolle. Einerseits erlauben sie Bürgern oder Unternehmen eine raschere Abwicklung von Anträgen bzw. einen „Rund um die Uhr"-Zugang zu Behörden, andererseits erleichtert E-Government auch den Behörden selbst die vereinfachte Durchführung gewisser Aktivitäten wie z.B. die Bearbeitung von elektronischen Akten [BKA01] im Back-Office Bereich.

Häufig werden bei behördlichen Verfahren sensible und schützenswerte Daten von Bürgern oder Unternehmen verarbeitet. Im Rahmen von E-Government Anwendungen muss Bürgern und Unternehmen daher die gleiche Sicherheit, Vertrauenswürdigkeit oder Transparenz wie bei traditionellen behördlichen Prozessen geboten werden. Um diesen Anforderungen gerecht zu werden, müssen eingesetzte E-Government Komponenten nicht nur sorgfältig entworfen und entwickelt, sondern nach deren Fertigstellung auch ausgiebig auf ihre korrekte Funktionalität und ihr Verhalten getestet werden. Während solche Tests für serverseitige Komponenten zumeist relativ unkompliziert und automatisiert bewerkstelligbar sind, stellt sich die Situation bei clientseitigen Komponenten ungleich komplexer dar. Zu heterogen ist die Betriebssystem- bzw. Browserlandschaft, die auf den lokalen Rechnern von Benutzern installiert ist, was auch durch diverse Statistiken belegt wird [NMS11] [BS11].

Wesentliche Forderungen im E-Government sind oftmals Technologieunabhängigkeit bzw. die Möglichkeit der Nutzung für alle Bürger [DigÖ08]. In Österreich sind diese Forderungen sogar gesetzlich verankert [EGovG04]. Für E-Government Komponenten soll dabei die Unabhängigkeit von speziellen Hardware- und Softwarekomponenten als Ziel verfolgt werden, um sich auch in Zukunft in keine Abhängigkeit von einzelnen externen Komponenten begeben zu müssen und Technologieneutralität gewährleisten zu können.

Für die Entwicklung von E-Government Komponenten bedeutet dies, dass nicht speziell nur auf eine Hardware oder bestimmte Umgebung gesetzt werden kann. Die Forderung nach Plattformunabhängigkeit und Technologieneutralität schlägt sich auch auf Test-Prozeduren von E-Government Komponenten nieder. So darf für Tests nicht nur die E-Government Software isoliert betrachtet werden, sondern ebenfalls ihr Verhalten in unterschiedlichen Umgebungen, da diese unter Umständen die Funktionalität der Software beeinflussen können.

Die Erfahrung zeigte, dass die Umgebung, in der eine E-Government Komponente betrieben wird, tatsächlich schwerwiegende negative Auswirkungen auf Funktionalität und Sicherheit der eigentlichen E-Government Komponente haben kann. Im Folgenden wird anhand einiger Fallbeispiele gezeigt, inwiefern unterschiedliche Umgebungen die Funktionalität und Sicherheit von E-Government Komponenten negativ beeinflussen können.

- **Integration von Chipkarten (Kartenleser):** Chipkarten spielen im Bereich des E-Governments in vielen Ländern eine zentrale Rolle. Einerseits werden sie häufig zur eindeutigen elektronischen Identifikation und Authentifizierung verwendet, anderseits finden sie Einsatz als *Secure Signature Creation Device (SSCD)* zur Erstellung von elektronischen Signaturen gemäß der Signaturrichtlinie [EP99]. Identifikation oder elektronische Signaturen bilden wesentliche Bestandteile eines sicheren und zuverlässigen E-Governments. Sollte diese Funktionalität nach einem Software-Update in der Umgebung des Benutzers nicht mehr gegeben sein, so erfolgt eine gravierende Einschränkung in der

Nutzung von E-Government Anwendungen. Im Rahmen von Tests österreichischer E-Government Komponenten konnte festgestellt werden, dass Treiber-Updates im Betriebssystem dazu führen können, dass einzelne Kartenleser nicht mehr ordnungsgemäß funktionieren und so die Funktionalität der E-Government Anwendung nicht mehr gegeben ist. Diese Problematik erstreckt sich nicht nur auf ein Betriebssystem, sondern konnte auf mehreren Systemen (darunter MS Windows, Linux und Apple Mac OS) beobachtet werden:

- **Microsoft Windows:** Hier zeigte sich unter anderem das Verhalten, dass die mit MS Windows ausgelieferten Default-Treiber für Kartenlesegeräte nicht immer korrekt funktionierten. Außerdem konnte ein unterschiedliches Verhalten bei verschiedenen Versionen der von den Herstellern bereitgestellten Treiber festgestellt werden.
- **Linux:** Die Unterstützung für Kartenleser-Treiber ist unter Linux oft mangelhaft. In einigen Distributionen fehlen entsprechend fertige Treiber-Pakate, sodass Benutzer angehalten werden, selbst Treiber-Binaries zu erstellen und zu installieren. Des Weiteren ist die Funktionalität von Pinpad-Kartenleser unter Linux nicht immer verfügbar.
- **Apple Mac OS:** Auch unter Apple Mac OS konnte der Effekt festgestellt werden, dass Pinpad-Kartenleser nicht funktionieren, da der in MAC OS enthaltene PCSC Daemon die Pinpad Funktionalität nach dem PCSCv2 Standard nicht unterstützt. Probleme traten nach Betriebssystemupdates unter anderem auch mit der PCSC-Schnittstelle, welche für die Chipkartenkommunikation zuständig ist, auf.

- **Abhängigkeit von Java:** Im Rahmen des österreichischen E-Governments wird vermehrt auf Java gesetzt, um die gewünschte Plattformunabhängigkeit zu erreichen. So ist beispielsweise die österreichische Bürgerkartenumgebung MOCCA, welche eine Client-Middleware für die Kommunikation mit der österreichischen Bürgerkarte darstellt, als Java Applet implementiert. Aufgrund dessen ist die korrekte Funktionalität von Java eine wichtige Vorrausetzung für ein korrektes Verhalten der Bürgerkartenumgebung. Vergangene Tests haben gezeigt, dass z.B. ein Bug in einem neuen Java-Update unter bestimmten Umständen (wenn das Applet aus dem Browser-Cache geladen wurde) das Nichtanzeigen von Labels in Java-Applets zur Folge hatte. Die Auswirkung auf die Bürgerkartenumgebung war, dass der zu signierende Text für den Bürger nicht mehr dargestellt wurde und somit der Bürger die zu signierenden Daten nicht mehr einsehen konnte. Dies ist ein sicherheitskritisches Beispiel dafür, wie ein Update einer Umgebungskomponente eine E-Government-Komponente negativ beeinflussen kann. Da Java selbst regelmäßigen Update-Zyklen ausgesetzt ist, ist es daher notwendig, sämtliche auf Java basierende kritische E-Government Komponenten nach jedem Java-Update auf deren korrekte Funktionalität hin zu überprüfen.
- **Web-basierte Services (Browser):** Web-basierte Services bzw. Web Browser spielen eine wesentliche Rolle bei der Verwendung von E-Government Anwendungen. Obwohl einheitliche Spezifikation für z.B. HTML oder das HTTP-Protokoll existieren, so unterscheiden sich trotzdem meist die Implementierungen dieser Spezifikationen. Diese (oft ungewollte) unterschiedliche Auffassung der Spezifikationen durch Entwickler kann dazu führen, dass sich bei Verwendung von E-Government Komponenten in verschiedenen Umgebungen ebenfalls ein unterschiedliches Verhalten zeigt. Ein konkretes Beispiel ist die Implementierung der Same-Origin-Policy (SOP), die in vielen Browsern leicht unterschiedlich interpretiert und umgesetzt wird und bereits öfters zu Problemen bei der Verwendung diverser E-Government Komponenten geführt hat. Aus diesem Grund ist es un-

erlässlich, die Funktionalität von E-Government Komponenten in unterschiedlichen Umgebungen auf Korrektheit zu überprüfen.

- **Änderungen in Betriebssystemen:** Gravierenden Änderungen an Betriebssystemen können negative Auswirkungen auf die Funktionalität einzelner Softwarekomponenten haben. Dies gilt natürlich auch für Komponenten, die im Rahmen von E-Government Anwendungen eingesetzt werden. Als konkretes Beispiel kann hier die Einführung virtueller Verzeichnisse in Microsoft Windows Versionen ab Vista genannt werden. Hier konnten einige Probleme beobachtet werden, die zwar nicht unmittelbar die Sicherheit der E-Government Anwendung, sehr wohl jedoch deren Funktionalität in Mitleidenschaft zogen. Es ist jedoch grundsätzlich nicht auszuschließen, dass sich Änderungen an Betriebssystemen unter Umständen auch negativ auf das Sicherheitsniveau von E-Government Komponenten auswirken können.

Wie die beschriebenen Beispiele bzw. Probleme zeigen, ist es im Rahmen der Qualitätssicherung von E-Government Komponenten wichtig, dass diese Komponenten nicht nur isoliert, sondern auch im Kontext mit unterschiedlichen Umgebungen auf korrekte Funktionalität überprüft werden. Um der Heterogenität der Systeme der einzelnen Benutzer-Clients gerecht zu werden, ist es notwendig, die entwickelte Software auf unterschiedlichen Systemkonfigurationen zu testen, um auf etwaige Einflüsse der Benutzerumgebung rechtzeitig und entsprechend reagieren zu können. Im Wesentlichen können die Anforderungen, welche ein für diese Zwecke geeignetes Testframework erfüllen muss, folgendermaßen zusammengefasst werden:

- **Zentrale Verfügbarkeit:** Entwicklern bzw. Testingenieuren soll mittels eines geeigneten Testsystems die Arbeit zur Qualitätssicherung abgenommen bzw. erleichtert werden. Das Testsystem mit unterschiedlichen Umgebungen (z.B. Betriebssystemen oder Browserversionen) sollte deshalb zentral verwaltet werden, um so auch mehreren Personen gleichzeitig ein einheitliches System zur Verfügung stellen zu können.
- **Dynamische Adaptierbarkeit:** Die Update-Zyklen von Softwareprodukten sind üblicherweise sehr dynamisch und finden nicht immer zu fixen Zeitpunkten statt. Um daher auf Änderungen wie z.B. ein neues Browser-Update rasch reagieren zu können, bedarf es eines Testsystems, welches rasch angepasst oder erweitert werden kann.
- **Effiziente Verwaltung:** Der Mix aus unterschiedlichen Betriebssystemen, Browsern oder Java-Versionen kann schnell zu einer unübersichtlichen Test-Matrix führen. Aus diesem Grund ist eine effiziente Verwaltung des Testframeworks eine weitere zentrale Anforderung.
- **Nachvollziehbarkeit:** Das Testen von Komponenten in einer bestimmten Umgebung ist kein einmaliger Vorgang. Sollte es beispielsweise trotz positiver Testergebnisse zu Problemen kommen, so sollte die verwendete Testumgebung leicht wiederherstellbar sein, um die durchgeführten Tests nochmals nachvollziehen zu können.

Die Erfüllung dieser Anforderungen ist für die Durchführung von effizienten und umfangreichen Tests von E-Government Komponenten unerlässlich. Im folgenden Abschnitt werden bestehende Teststrategien und Frameworks diskutiert und deren Tauglichkeit zur Erfüllung der oben genannten Anforderungen evaluiert.

3 Teststrategien und Frameworks

Aufgrund der Wichtigkeit von systematischen Tests zur Qualitätssicherung in der Softwareentwicklung existieren bereits eine Vielzahl an bewährten Strategien und Werkzeugen. In die-

sem Abschnitt werden einige dieser Ansätze diskutiert und auf deren Tauglichkeit für E-Government Komponenten hin überprüft.

Sicherheitskritische Softwarekomponenten werden häufig gemäß dem *Common Criteria for Information Technology Security Evaluation (Common Criteria, CC)* Standard [CCMB09] formell evaluiert und zertifiziert. Dieser international abgestimmte Standard zur Informationssicherheit enthält allgemeine Kriterien für die Prüfung und Bewertung der Sicherheit von Informationstechnologien. Vor einer Common Criteria Evaluierung wird das sogenannte *Target of Evaluation (TOE),* das meistens das zu evaluierende System darstellt, genau definiert und von anderen Komponenten, die der Umgebung zugeordnet werden, abgegrenzt. Die Kriterien werden nur auf dieses TOE angewendet, während die Umgebung nicht näher betrachtet wird.

Während sich dieser Ansatz zur Überprüfung und Evaluierung einzelner isolierter Komponenten bewährt hat, reicht die Common Criteria Evaluierung einzelner Komponenten für komplexe E-Government Lösungen oft nicht aus. Wie die Erfahrung zeigte, kann eine an sich sichere Komponente durch diverse Gegebenheiten der Umgebung in Funktion und Sicherheit signifikant beeinträchtigt werden. Komplexe E-Government Softwarelösungen können es sich daher nicht leisten, nur einen bestimmten und definierten Teilbereich einer intensiven Qualitäts- und Sicherheitsüberprüfung zu unterziehen. Vielmehr muss die Anwendungen als Einheit mit ihrer Umgebung, mit der diese in Wechselwirkung steht, gesehen werden. Dementsprechend ist auch die Teststrategie auszurichten.

Durch die Miteinbeziehung der Umgebung in die Teststrategie gestalten sich die Anforderungen an ein Testframework für clientseitige oder webbasierte E-Government Softwarekomponenten umfangreich und komplex, da eine Vielzahl an Faktoren bei der Durchführung von Softwaretests berücksichtigt werden müssen. Eine besondere Rolle spielt hier die lokale Umgebung des Benutzers. Durch die unterschiedlichen Betriebssysteme und die Fülle an gängigen Browsern, die diversen Java Runtime Environments (JRE) Versionen, und der Vielzahl an unterschiedlichen Kartenlesern und Chipkarten ergibt sich eine komplexe Testmatrix mit einer erheblichen Anzahl an abzudeckenden Testszenarien.

Das Testen webbasierter Anwendungen auf unterschiedlichen Clientsystemen ist keine auf den E-Government Bereich beschränkte Herausforderung. Dementsprechend existieren bereits eine Vielzahl an Tools und Frameworks, die systematische und teilweise automatisierte Tests webbasierter Anwendungen erlauben.

- Selenium [HoKe06] ist ein auf HTML und JavaScript basierendes Testframework für Web Anwendungen. Durch die Aufzeichnung der Benutzerinteraktion mit der Web Anwendung können Tests automatisch wiederholt werden, wodurch Testdurchläufe schneller und zuverlässiger durchgeführt werden können. Selenium wird in den vier Ausführungen Selenium Core, Selenium IDE , Selenium Remote Control (RC) und Selenium Grid angeboten. Das Framework ist betriebssystemunabhängig und kann mit allen Browsern verwendet werden. Selenium unterstützt zwar die wiederholte Durchführung von Testdurchläufen, bietet jedoch keine Möglichkeiten um Tests auf verschiedenen Systemkonfigurationen systematisch durchzuführen.
- Die HP Functional Testing Software [HP11] und IBM Rational Functional Tester [IBM11] sind gängige Testwerkzeuge, welche Aufnahmen der Benutzerinteraktion mit einer Softwarebenutzeroberfläche erlauben und damit ein automatisiertes Testen ermöglichen. Diese Tools unterstützen die Erzeugung von Funktions- und Regressionstests. Die Testaufnahmen können einfach über einen Rekorder erstellt werden. Alternativ kann für die Erzeugung der automatischen Testfälle auch eine Skriptsprache verwendet werden.

Dadurch können Aufnahmen verändert und parametrisiert werden. Ähnlich wie Selenium benötigen auch diese Test-Frameworks eine Testumgebung auf der sie eingesetzt werden können. Da diese Tools nicht plattformunabhängig sind, können sie zum Testen von E-Government Anwendungen jedoch nur bedingt eingesetzt werden.

- Im Gegensatz zu den bereits genannten Tools verfolgen AdobeBrowserLab [Adobe11] und CrossBrowserTesting [CBT11] einen anderen Ansatz. Diese Online-Tools ermöglichen die Evaluierung einer Webseite mit unterschiedlichen Betriebssystemen und Browsern. Durch Angabe einer URL wird die unter dieser URL erreichbare Seite automatisch überprüft. Über diese Tools sind prinzipiell Tests auf verschiedenen Plattformen möglich. Die für E-Government Anwendungen relevanten Aspekte wie Kompatibilität zu Kartenlesegeräten oder verschiedenen JREs können mit diesen Tools jedoch ebenfalls nicht abgedeckt werden.

Obwohl bereits eine Vielzahl an Ansätzen und unterstützenden Tools für automatisiertes Testen existiert, sind diese für komplexe E-Government Anwendungen nur bedingt einsetzbar. Vor allem die für E-Government Anwendungen spezifischen Aspekte wie Chipkartenkommunikation oder die Abhängigkeit von Java Laufzeitumgebungen machen einen effizienten Einsatz bestehender Tools und Frameworks oft unmöglich.

Aus diesem Grund wurde für Elemente des österreichischen E-Governments ein alternativer Ansatz entwickelt und in Form eines virtuellen Testframeworks, welches im nächsten Abschnitt näher beschrieben wird, umgesetzt.

4 Virtuelles Testsystem

Durch die besonderen Anforderungen, die umfangreiche Tests von E-Government Komponenten mit sich bringen, bietet die Verwendung herkömmlicher Testframeworks oft keine zufriedenstellende Lösung. Um der Dynamik sich ständig verändernder Umgebungen begegnen zu können, wurde von den Autoren ein alternativer Ansatz verfolgt. Ziel war die Entwicklung eines zentralen virtuellen Testframeworks, welches flexibel auf sich ändernde Anforderungen angepasst und einfach für effiziente Tests verschiedener E-Government Komponenten verwendet werden kann.

In diesem Abschnitt werden Aufbau und Funktionen des entwickelten Frameworks näher vorgestellt. Anhand praktischer Erfahrungen, die mit diesem Framework bisher gesammelt werden konnten, werden Vorteile dieses Ansatzes diskutiert und weitere Verbesserungsmöglichkeiten skizziert.

4.1 Aufbau

Es wurde ein Ansatz virtualisierter Systeme verfolgt. Das beschriebene Framework verwendet Komponenten der Firma VMware, die zur besseren Veranschaulichung für den Leser im Folgenden auch mit Produktnamen genannt sind. Das grundlegende Konzept ist aber auch auf andere Virtualisierungslösungen übertragbar.

Das erstellte Testframework bietet Benutzern ein umfangreiches Repository an unterschiedlichen Testumgebungen und Systemkonfigurationen in Form von virtuellen Maschinen. Zur Verwaltung der virtuellen Maschinen kommen im Speziellen die Komponenten VMware ESX Server [VmS11], VMware vCenter Server [VmC11] und VMware LabManager Server [VmL11] zum Einsatz. Abb. 1 zeigt den prinzipiellen Aufbau des Testframeworks.

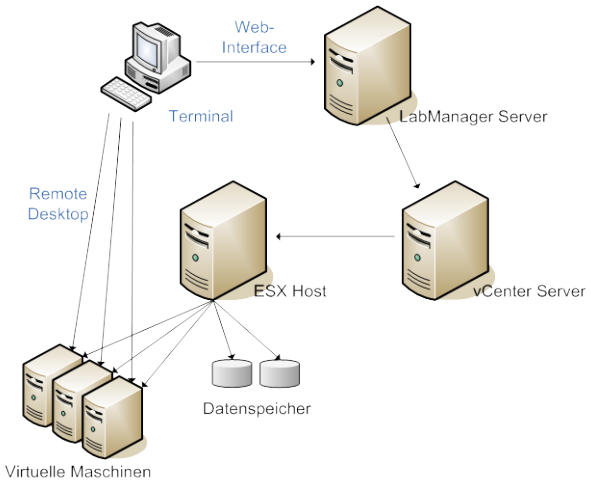

Abb. 1: Allgemeiner Aufbau des virtuellen Testframeworks

Ein zentraler VMware ESX Server, welcher über ausreichend Rechenleistung, sowie Arbeits- und Datenspeicher verfügt, dient als zentrale Komponente und stellt eine Reihe definierbarer virtueller Maschinen für Testzwecke zur Verfügung. Die Verwaltung dieser virtuellen Maschinen kann über ein Web-Interface vorgenommen werden. Dieses wird vom zentralen VMware LabManager Server zur Verfügung gestellt, welcher seinerseits über einen VMware vCenter Server auf den zentralen ESX Host Zugriff hat. Aus Sicht der Benutzer sind sowohl ESX Server als auch vCenter Server völlig transparent.

Der Zugriff auf die einzelnen virtuellen Maschinen kann entweder über den VMware LabManager Server und eine VMware Konsole in Form eines Browser Plug-ins, oder auch direkt über Remote-Desktop Protokolle erfolgen.

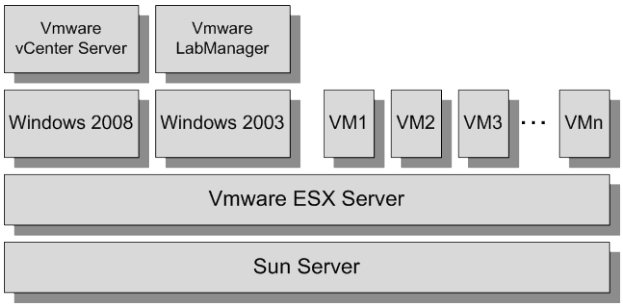

Abb. 2: Verwendete Infrastruktur

Der in Abb. 1 dargestellte prinzipielle Aufbau wurde gemäß dem in Abb. 2 dargestellten Schema umgesetzt. Als einzige physikalische zentrale Komponente kam dazu ein Sun Server zum Einsatz. Sämtliche anderen Komponenten wurden basierend auf dieser physikalischen Komponente rein virtuell ausgeführt. Der VMware ESX Server wurde direkt auf dem Sun Server installiert und dient als Basis für alle weiteren virtuellen Komponenten. Auf dem ESX Server wurden zwei virtuelle Microsoft Windows Rechner aufgesetzt, auf welchen die beiden VMware Produkte vCenter Server und LabManager Server installiert wurden[1]. Der ESX Server bildet des Weiteren die Grundlage für alle anderen virtuellen Maschinen des Testframeworks.

[1] Ein Betrieb dieser beiden VMware Komponenten auf einem einzigen System war aufgrund unterschiedlicher Anforderungen an das Betriebssystem nicht möglich.

4.2 Funktionen

Die verwendeten VMware Komponenten bieten Benutzern prinzipiell eine Fülle an Möglichkeiten und Funktionen. Für Tests von E-Government Kernkomponenten ist folgende Verwendung des Testframeworks vorgesehen.

1. Der Benutzer verbindet sich mit dem VMware Lab Manager Server und öffnet das webbasierte Interface zur Verwaltung von Konfigurationen[2] und virtuellen Maschinen.
2. Der Benutzer wählt aus einer vorhandenen Sammlung von Konfiguration die für den jeweiligen Test passenden virtuellen Maschinen aus und klont daraus eine eigene Arbeitskopie.
3. Der Benutzer startet die in der Arbeitskopie enthaltenen geklonten virtuellen Maschinen.
4. Der Benutzer verbindet sich zu den gestarteten virtuellen Maschinen über die webbasierte VMware Konsole oder direkt über Remote Desktop Protokolle.
5. Der Benutzer führt die vorgesehenen Tests aus.
6. Nach Beendigung der Tests fährt der Benutzer die verwende Arbeitskopie herunter.
7. Die Arbeitskopie wird ja nach Einstellung nach einer bestimmten Zeit gelöscht.

Neben dieser Grundfunktionalität bietet das Testframework zahlreiche zusätzliche Funktionen, sodass sämtliche in Abschnitt 2 definierte Anforderungen erfüllt werden können. Im Speziellen bieten sich durch den Einsatz dieses Systems für Benutzer unter anderem die im Folgenden näher erläuterten Möglichkeiten und Vorteile.

4.2.1 Zentrale Verfügbarkeit

Durch den Einsatz zentraler Komponenten und den Betrieb des Testframeworks auf einem allgemein zugänglichen Server kann das Framework von mehreren Personen (auch gleichzeitig) genutzt werden. Der zentrale Ansatz erleichtert zudem die Wartung der verschiedenen virtuellen Maschinen und ermöglicht eine effiziente Nutzung von Ressourcen.

4.2.2 Dynamische Adaptierbarkeit

Das Testframework verfügt über ein Set an vordefinierten virtuellen Maschinen mit unterschiedlichen Konfigurationen. Diese reichen von Systemen mit neu installierten Betriebssystemen bis hin zu Systemen mit diversen vorinstallierten Softwarekomponenten wie JRE, Browser und Bürgerkartenumgebungen. Somit ist ein Großteil aller gängigen Systemkonfigurationen für die Durchführung von Tests sofort verfügbar.

Die vordefinierten Systeme können von Benutzern jederzeit geladen, geklont und für eigene Testzwecke beliebig verwendet werden. Nach Abschluss der Tests können diese Klone entweder gelöscht, oder aber dem Set an vordefinierten Systemen hinzugefügt und so für eine spätere Wiederverwendung (auch durch andere Benutzer) aufbewahrt werden. Der Zeitpunkt des Löschens, kann bei der Erstellung der Konfiguration bestimmt und jederzeit dem Testverlauf angepasst werden. Durch das Klonen werden die Basiskonfigurationen erhalten wodurch der Urzustand des jeweiligen Systems für den nächsten Testdurchlauf wieder verwendet werden kann. Sollte keine der im Testframework vordefinierten virtuellen Maschinen für den

[2] Unter einer Konfiguration versteht man eine Sammlung virtueller Maschinen.

durchzuführenden Test geeignet sein, können Benutzer Betriebssysteme, die sich unter den VM Vorlagen befinden, zu einem eigenen Set hinzufügen und verwenden.

4.2.3 Effiziente Verwaltung

Eine effiziente Verwaltung des Testframeworks wird prinzipiell durch den zentralen Ansatz begünstigt. Der verwendete VMware LabManager Server erlaubt neben dem raschen Hinzufügen, Klonen und Entfernen von virtuellen Maschinen auch eine flexible Gruppierung und Archivierung von virtuellen Maschinen. Durch ein flexibles Benutzerverwaltungs- und Rechtemanagementsystem können zudem Berechtigungen für Benutzer dynamisch angepasst und auf gegebene Anforderungen abgestimmt werden.

4.2.4 Nachvollziehbarkeit

Der VMware LabManager Server verfügt über eine umfangreiche Archivierungsfunktion. Diese kann dazu genutzt werden um bestimmte Systemkonfigurationen zu konservieren und für eine spätere erneute Verwendung verfügbar zu halten. Damit kann die Nachvollziehbarkeit von Testdurchläufen auf bestimmten Systemkonfigurationen jederzeit gewährleistet werden.

4.3 Evaluierung

Im praktischen Einsatz zeigt sich sehr rasch, dass durch die Verwendung des virtuellen Testframeworks die Effizienz und der Umfang von Testdurchläufen gesteigert werden konnte. Vor allem ermöglicht das Testframework eine erhöhte Frequenz von Tests sicherheitsrelevanter Komponenten, wodurch etwaige durch Änderungen der jeweiligen Umgebung hervorgerufene Probleme früh erkannt werden können. Daneben offenbarten sich während der Verwendung des Testframeworks auch einige Probleme, die jedoch im Zuge einer weiteren Optimierung des Frameworks lösbar sein sollten.

Ein klarer Vorteil des bei diesem Framework verfolgten Ansatzes ist dessen zentraler Ansatz. Dadurch stehen Entwicklern und Supportmitarbeitern jederzeit beliebige Testumgebungen zur Verfügung. Vor allem im Zuge der Bearbeitung von Supportanfragen erwies sich das Framework als hilfreich, da dieses ein rasches Rekonstruieren bestimmter Umgebungen und Durchführen entsprechender Tests erlaubt. Von der raschen Verfügbarkeit beliebiger Konfigurationen profitieren auch Entwickler von E-Government Komponenten, da diese bereits während des Entwicklungsprozesses einzelne Komponenten auf deren Kompatibilität zu verschiedenen Umgebungen hin überprüfen können. Schließlich konnte durch das virtuelle Testframework auch die Durchführung abschließender, systematischer Tests signifikant verbessert werden.

Der intensive Einsatz des virtuellen Testframeworks offenbarte auch diverse Verbesserungspotentiale. Im Zusammenhang mit E-Government Komponenten erwies sich vor allem die bei manchen Betriebssystemen fehlende Unterstützung für eine Weiterleitung der USB Schnittstelle an das virtuelle Testsystem als störend. Dies betrifft jedoch hauptsächlich Linux basierte Testumgebungen und sollte durch Verwendung alternativer Remote Desktop Programme lösbar sein. Als relativ aufwändig erwies sich auch die Wartung der unterschiedlichen virtuellen Maschinen des Testframeworks. Durch den Einsatz entsprechender Skripts (z.B. zur automatisierten Durchführung von Betriebssystemupdates) konnte jedoch auch dieser Herausforderung begegnet werden. Probleme ergaben sich auch aufgrund lizenzrechtlicher Rahmenbedingungen, die beispielsweise eine Virtualisierung von Mac OS Betriebssystemen verhindert.

Diese Limitierung lässt sich durch ein Ausweichen auf physikalische Systeme umgehen, führt jedoch zu erhöhtem Aufwand und geringeren Variationsmöglichkeiten.

Insgesamt führte der praktische Einsatz der virtuellen Testumgebung zu einer deutlichen Erhöhung der Effizienz in der Durchführung von Applikationstests und trug so zur Gewährleistung adäquater Qualitäts- und Sicherheitsstandards bei.

5 Fazit

Umfangreiche Tests sind für die Qualitätssicherung in der Softwareentwicklung im Allgemeinen und für E-Government Komponenten im Speziellen von besonderer Bedeutung. Vor allem im E-Government Bereich, in dem häufig mit sensiblen und sicherheitskritischen Daten gearbeitet wird, ist eine korrekte Funktionalität der eingesetzten Software wichtig, um einen adäquaten Schutz dieser Daten zu gewährleisten. Die Erfahrung zeigte, dass übliche Test- und Evaluierungsansätze wie z.B. die Common Criteria Evaluierung oft zu kurz greifen, da diese zwar eine definierte Softwarekomponente detailliert betrachten, die Umgebung, in der diese Komponente eingesetzt wird, jedoch oft außer Acht lassen. Vor allem in clientseitigen E-Government Anwendungen spielt die Umgebung (Betriebssystem, Browser, JRE, etc.) jedoch oft eine entscheidende Rolle und kann Funktionalität und Sicherheit der eigentlichen E-Government Komponente kompromittieren. Aus diesem Grund ist eine Miteinbeziehung der Umgebung in die Durchführung von Tests unerlässlich.

Aufgrund der großen Anzahl an gängigen Betriebssystemen, Browsern und anderen Komponenten der Umgebung stellt die Durchführung vollständiger Tests eine ernstzunehmende Herausforderung dar. Erschwert wird die Situation zusätzlich durch häufige Updates der Komponenten der Umgebung (Browser-Updates, Betriebssystem-Updates, etc.), die stets neue Testdurchläufe erforderlich machen. Um dieser Komplexität und Dynamik Herr zu werden, sind systematische Ansätze notwendig. Eine Analyse bestehender Tools und Frameworks zeigte, dass diese vorhandenen Testwerkzeuge meist nicht in der Lage sind die Anforderungen für Tests von E-Government Komponenten zu erfüllen. Hier stellte sich besonders die Integration von Chipkarten als schwer zu automatisierendes Problem dar.

Aufgrund der Unzulänglichkeiten existierender Ansätze und Frameworks wurde ein alternativer Ansatz entwickelt und in diesem Paper vorgestellt. Basierend auf Produkten der Firma VMware Inc. wurde ein virtuelles Testframework erstellt. Diese besteht aus einem einfach zu verwaltenden Repository virtueller Maschinen, die unterschiedliche Umgebungen für Tests zur Verfügung stellen. Durch den Einsatz dieses virtuellen Testframeworks konnten der Umfang und die Effizienz der Tests österreichischer E-Government Komponenten signifikant verbessert werden. Die dadurch erreichbare Steigerung der Qualität führte auch zu einer Erhöhung der Sicherheit der getesteten Komponenten. Das in diesem Paper vorgestellte Testframework trägt somit wesentlich zur Qualität und Sicherheit der österreichischen E-Government Infrastruktur bei.

Literatur

[Adobe11] Adobe BrowserLab, https://browserlab.adobe.com/ (2011)

[BKA01] Bundeskanzleramt: ELAK-Konzept. http://www.digitales.oesterreich.gv.at/DocView.axd?CobId=19396 (2001)

[BS11] Browser Statistik, http://www.browser-statistik.de/ (2011)

[CBT11] Cross Browser Testing, http://crossbrowsertesting.com/ (2011)

[CeOB10] M. Centner, C. Orthacker, W. Bauer: Minimal-Footprint Middleware for the Creation of Qualified Signatures. In: Proceedings of the 6th International Conference on Web Information Systems and Technologies (2010) 64-69.

[CCMB09] Common Criteria for Information Technology Security Evaluation, CC v3.1. Release 3, http://www.commoncriteriaportal.org/cc/ (2009)

[DigÖ08] Digitales Österreich, Behörden im Netz - Das österreichische E-Government ABC, http://www.bka.gv.at/DocView.axd?CobId=27782 (2008)

[EGovG04] E-Government-Gesetz (E-GovG) (2004), Bundesgesetzblatt für die Republik Öster-reich BGBl.I Nr. 10/2004

[EP99] Richtlinie1999/93/EG des Europäischen Parlaments und des Rates vom 13. Dezember 1999 über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2000:013:0012:0020:DE:PDF (1999)

[HoKe06] A. Holmes, M. Kellogg: Automating Functional Tests Using Selenium, Proceedings of AGILE 2006 Conference (2006)

[HP11] Functional Testing Software, Simplify the automation of functional and regression testing https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24^1322_4000_100_(2011) zuletzt abgerufen am 28.04.2011

[IBM11] Rational Functional Tester, Performance test automation for quality driven software delivery http://www-01.ibm.com/software/awdtools/tester/performance/ (2011)

[LeHP02] H. Leitold, A. Hollosi, R. Posch: Security Architecture of the Austrian Citizen Card Concept. In: ACSAC '02, Proceedings of the 18th Annual Computer Security Applications Conference, IEEE Computer Society (2002) 391.

[LePR09] H. Leitold, R. Posch, and T. Rössler: Media-break resistant eSignatures in eGovernment: an Austrian experience. In: Javier Lopez Dimitris Gritzalis, Emerging Challenges for Security, Privacy, and Trust - 24th IFIP SEC, Springer (2009) 109-118.

[NMS11] Net Market Share, Operating System Market Share, http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10 (2011)

[SigG99] Signaturgesetz (SigG) (1999), Bundesgesetzblatt für die Republik Österreich BGBl.I Nr. 190/1999

[VmS11] VMware vSphere Server, http://www.vmware.com/de/products/vi/esx/ (2011)

[VmC11] VMware vCenter Server, http://www.vmware.com/de/products/vi/vc/ (2011)

[VmL11] VMware vCenter Lab Manager, http://www.vmware.com/de/products/labmanager/overview.html (2011)