Vesna Krnjic, Philip Weber, Thomas Zefferer, Bernd Zwattendorfer - "Effizientes Testen von E-Government Komponenten in der Cloud" - D-A-CH Security 2013, pp. 225-236

Effizientes Testen von E-Government Komponenten in der Cloud

Vesna Krnjic · Philip Weber · Thomas Zefferer · Bernd Zwattendorfer

Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie – Technische Universität Graz

{vesna.krnjic, philip.weber, thomas.zefferer, bernd.zwattendorfer}@iaik.tugraz.at

Zusammenfassung

Für softwarebasierte Anwendungen in sicherheitskritischen Bereichen wie E-Government stellt die Durchführung umfassender Tests ein wichtiges Mittel zur Sicherstellung definierter Qualitäts- und Sicherheitsmerkmale dar. Aufgrund der ständig zunehmenden Komplexität von Softwarelösungen auch in sicherheitskritischen Anwendungen, werden für die Durchführung von Testroutinen im verstärkten Maße entsprechende Frameworks und Tools eingesetzt. Vor allem im Bereich des E-Government ergeben sich jedoch mitunter besondere Anforderungen, die von bisher verfügbaren Frameworks und Tools nicht vollständig erfüllt werden können. Zur Lösung dieses Problems stellen wir in diesem Artikel ein neuartiges Testframework vor. Dieses kombiniert die Vorteile etablierter virtueller Testframeworks mit Konzepten des Cloud Computing. Damit bietet das vorgestellte Testframework Entwicklerinnen und Entwicklern von E-Government-Anwendungen eine umfassende Lösung zum systematischen Testen sicherheitskritischer Komponenten im Bereich des E-Government und trägt zur Sicherung definierter Qualitäts- und Sicherheitsmerkmale bei.

1 Einleitung

Für die professionelle Entwicklung von Software ist die Gewährleistung ihrer Qualität von zentraler Bedeutung. Nach ISO/IEC 25010:2011 [ISO11] werden für die Messung der Qualität von Software u.a. die Faktoren Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Portabilität herangezogen. Formale Nachweise für die Einhaltung dieser Faktoren sind aufgrund der zunehmenden Komplexität von Software in der Praxis schwierig bis unmöglich zu erbringen. Aus diesem Grund stellen umfassende und ausgeklügelte Testmethoden derzeit den wichtigsten Ansatz zur Überprüfung der Einhaltung definierter Qualitätsmerkmale dar.

Die Einhaltung dieser Qualitätsmerkmale durch umfassende Tests ist besonders für sicherheitskritische Applikationen von zentraler Bedeutung, da hier auftretende Mängel unmittelbar zu Schäden unterschiedlicher Natur führen können. Prominente Beispiele für sicherheitskritische Anwendungen sind beispielsweise Softwarelösungen in den Bereichen E-Government oder E-Banking. Fehlerhafte bzw. mangelhafte Software kann in diesen Bereichen zu erheblichen finanziellen Schäden (E-Banking), oder auch zur Kompromittierung privater und schützenswerter Daten (E-Government) führen. Problematisch ist dies auch vor allem dann, wenn Software zur Erstellung qualifizierter – und damit rechtsgültiger – elektronischer Signaturen gemäß EU Signaturrichtlinie [EP99] fehlerhaft ist. Speziell für die Bereiche E-Banking und

E-Government sind daher geeignete Testmethoden zur Sicherstellung der erforderlichen Qualität im Zuge der Entwicklung von Softwarelösungen unumgänglich.

Vor allem im Bereich E-Government ergeben sich zumeist komplexe und verteilte Softwarelösungen bestehend aus sich ergänzenden Server- und Client-Komponenten. So werden E-Government-Dienste häufig über zentrale Web- und Applikationsserver angeboten. Für die Authentifizierung an diesen Diensten kommen jedoch auch clientseitige Komponenten wie zum Beispiel Chipkarten und Middleware-Komponenten [CeOB10] für den lokalen Zugriff auf diese Chipkarten zum Einsatz.

Da Server-Komponenten in der Regel in kontrollierten Umgebungen zum Einsatz kommen, können diese im Zuge der Softwareentwicklung relativ einfach und effizient getestet werden, da deren Zielplattform und Infrastruktur bekannt oder beeinflussbar ist. Für Client-Komponenten stellt sich die Situation als schwieriger dar. Da diese direkt am System der Endbenutzerin bzw. des Endbenutzers installiert und betrieben werden, müssen hier eine Vielzahl an möglichen Systemkonfigurationen berücksichtigt werden, die sich potentiell unterschiedlich auf die Funktionalität und damit auf die Qualität der jeweiligen Softwarelösung auswirken. Relevante Unterschiede können sich hier beispielsweise in Bezug auf das verwendete Betriebssystem oder den verwendeten Web-Browser ergeben.

Im Zuge der Softwareentwicklung ergibt sich bei der Erstellung clientseitiger E-Government-Lösungen damit die Herausforderung, qualitativ hochwertige Lösungen für eine möglichst große Anzahl unterschiedlicher Systemkonfigurationen zu implementieren. Dafür bedarf es geeigneter Testmethoden, über die unterschiedliche Konfigurationen potentieller Ziel-Systeme effektiv und effizient abgedeckt werden können.

In der Vergangenheit zeigte sich, dass virtuelle Testframeworks [ZeKZ11] im Unterschied zu herkömmlichen Frameworks [HP11][IBM11] ein geeignetes Mittel darstellen, um der großen Anzahl unterschiedlicher zu testender Systemkonfigurationen Herr zu werden. Bei diesem Ansatz werden verschiedene Systemkonfigurationen durch virtuelle Maschinen repräsentiert, die durch geeignete Software effizient verwaltet und verwendet werden können. Obwohl dieser Ansatz rasch zu den erwünschten Ergebnissen führt, manifestieren sich im praktischen Einsatz nach einiger Zeit einige zentrale Nachteile virtueller Testframeworks. Beispielsweise wächst durch die rasch steigende Anzahl an unterschiedlichen Betriebssystem- und Browserversionen auch die Anzahl möglicher und zu berücksichtigender Systemkonfigurationen und damit die Anzahl der für die Durchführung umfangreicher und möglichst vollständiger Tests benötigten virtuellen Maschinen. Damit stoßen virtuelle Testframeworks mit begrenzten lokalen Ressourcen relativ rasch an ihre Grenzen und sind nicht mehr in der Lage, alle zu testenden Systemkonfigurationen abzudecken.

Um dieser Problematik zu begegnen, stellen wir in diesem Artikel eine Weiterentwicklung des Konzepts der virtuellen Testframeworks vor. Dabei bedienen wir uns des Konzepts des Cloud-Computing, welches in den letzten Jahren enorm an Bedeutung gewonnen hat. Wir zeigen, dass durch die Kombination des Konzepts virtueller Testframeworks mit jenem des Cloud-Computing das bestehende Problem der mangelnden Skalierbarkeit elegant gelöst werden kann. Die praktische Umsetzbarkeit dieses neuen Ansatzes wird anhand einer konkreten Implementierung gezeigt. Diese Implementierung ist bereits im produktiven Einsatz und wird seit einiger Zeit erfolgreich für systematische Tests österreichischer E-Government-Komponenten eingesetzt. Damit trägt dieses System zur Gewährleistung der Qualität und damit auch der Sicherheit österreichischer E-Government-Lösungen bei.

2 Anforderungen an Testframeworks

Die stetig wachsende Verbreitung von Informations- und Kommunikationstechnologien (IKT) bewirkte in den letzten Jahren Veränderungen in vielen Bereichen des täglichen Lebens. Diese Entwicklung machte auch vor der öffentlichen Verwaltung nicht halt. Unter dem Begriff E-Government entstanden in den letzten Jahren zahlreiche Lösungen, die unter Verwendung von IKT Bürgerinnen und Bürgern die Möglichkeit bieten, Amtsgeschäfte und Behördenwege effizient auf elektronischem Wege abzuwickeln. Durch die strikten Sicherheitsanforderungen, die sich vor allem für transaktionale E-Government-Dienste ergeben, weisen E-Government-Lösungen mitunter einen beachtlichen Grad an Komplexität auf und machen die Integration zusätzlicher sicherheitssteigernder Konzepte wie beispielsweise Chipkarten notwendig [CeOB10].

Als zusätzlich erschwerender Faktor kommt hinzu, dass Technologieunabhängigkeit in der Regel eine zentrale Anforderung an E-Government-Dienste darstellt. Die Bereitstellung technologieunabhängiger Lösungen ist wichtig, um eine möglichst breite Masse an potentiellen Benutzerinnen und Benutzern zu erreichen und niemanden von der Verwendung bereitgestellter Dienste auszuschließen. In einigen Ländern wie Österreich ist die Notwendigkeit technologieunabhängiger Lösungen sogar gesetzlich verankert [EGov04].

Für die Entwicklung von E-Government-Diensten und Applikationen stellt die Forderung nach Technologieunabhängigkeit eine ernstzunehmende Herausforderung dar. Vor allem im Bereich clientseitiger Komponenten ergibt sich durch die Forderung nach Technologieunabhängigkeit eine Vielzahl möglicher Endbenutzersystemkonfigurationen, die im Zuge von Funktionstests entsprechend berücksichtigt werden müssen. In diesem Zusammenhang zeigt die langjährige Erfahrung in Entwicklung und Betrieb von E-Government-Lösungen, dass vor allem die Integration von Chipkarten, gegebene Abhängigkeiten von Java-Laufzeitumgebungen, die stetig steigende Anzahl an im Umlauf befindlichen Web-Browser-Varianten und Version, sowie unerwartete Änderung von im Umlauf befindlichen Betriebssystemen die Funktionalität von E-Government-Lösungen beeinträchtigen können [ZeKZ11].

Bisherige Testframeworks, die Entwicklerinnen und Entwickler dabei unterstützten, der stetig wachsenden Anzahl an unterschiedlichen Systemkonfigurationen Herr zu werden, versuchen hauptsächlich, folgende Kriterien zu erfüllen [ZeKZ11]:

- **Dynamische Adaptierbarkeit:** Updatezyklen von Betriebssystemen und Software sind oft unregelmäßig und anlassbezogen. Ein bestehendes Testframework muss daher sehr rasch und einfach in Hinblick auf neue Versionen und damit auf neue Systemkonfigurationen adaptierbar sein.
- **Effiziente Verwaltung:** Durch ständig neue Versionen von Betriebssystemen und anderen Softwarekomponenten steigt die Anzahl potentieller Systemkonfigurationen stets an. Effiziente Mechanismen zur einfachen und übersichtlichen Verwaltung der einzelnen im Testframework abgebildeten Systemkonfigurationen ist daher ein weiteres wichtiges Kriterium.
- **Zentrale Verfügbarkeit:** In der Regel arbeiten mehrere Entwicklerinnen und Entwickler bzw. verschiedene Teams an der Erstellung von E-Government-Diensten und Applikationen. Um Ressourcen zu sparen, sollte das Testframework zentral zugänglich sein und allen berechtigten Entwicklerinnen und Entwicklern zur Verfügung stehen.

- **Nachvollziehbarkeit:** Durchgeführte Tests sollten jederzeit wieder nachvollziehbar sein, um im Falle von Problemen, die direkt bei Endbenutzerinnen oder Endbenutzern auftreten, entsprechende Systemkonfigurationen einfach wiederherstellen zu können.

Obwohl diese Kriterien durchaus ihre Berechtigung haben und ohne Zweifel die Grundlage entsprechender Testframeworks bilden müssen, zeigt die Erfahrung im praktischen Umgang mit entsprechenden Umsetzungen, dass diese Kriterien in dieser Form unvollständig sind und einige wichtige Aspekte außer Acht lassen. Dementsprechend schlagen wir eine Erweiterung der o.g. Liste an Kriterien und Anforderungen an Testframeworks um folgende Punkte vor:

- **Kompatibilität mit Chipkartentechnologie:** Obwohl laufend neue technologische Ansätze zur Umsetzung qualifizierter Signaturlösungen entwickelt und erprobt werden, stellen Chipkarten nach wie vor eine zentrale und häufig genutzte Technologie im Rahmen von E-Government-Lösungen dar. Für systematische Tests – und hier vor allem für automatische Tests – stellt dies eine bedeutende Herausforderung dar. Testframeworks, die für die Evaluierung von E-Government-Lösungen verwendet werden sollen, müssen daher in der Lage sein, mit Chipkarten entsprechend umgehen zu können.
- **Kompatibilität zu Java:** Vor allem im Zusammenhang mit der Integration von Chipkartentechnologie stellt Java nach wie vor eine wichtige Technologie dar. Zudem erlaubt Java die Erstellung plattformunabhängiger Lösungen, was einen Einsatz dieser Technologie vor allem im Bereich des E-Government als sinnvoll erscheinen lässt. Testframeworks, die in diesem Bereich eingesetzt werden sollen, müssen daher in der Lage sein, mit Java entsprechend umgehen zu können, um Probleme, die sich im Umgang mit dieser Technologie ergeben können, geeignet abbilden und simulieren zu können.
- **Skalierbarkeit:** Durch die relativ kurzen Updatezyklen gängiger Betriebssysteme und Anwendungen ergibt sich in kurzer Zeit eine relativ große Anzahl an potentiellen Systemkonfigurationen, die durch das Testframework abgedeckt werden müssen. Die Erfahrung zeigte, dass bestehende Testframeworks hier oft sehr rasch an ihre Grenzen stoßen. Nachhaltige Testframeworks müssen daher einen entsprechenden Grad an Skalierbarkeit aufweisen, um mit einer ständig wachsenden Anzahl an potentiellen Systemkonfigurationen umgehen zu können.
- **Einfache Wartung der Testumgebungen:** Softwarekomponenten werden in unregelmäßigen Abständen mit Patches versorgt, die beispielsweise kritische Sicherheitslücken in Web-Browsern schließen. Derartige Patches müssen in alle Systemkonfigurationen, die von einem Testframework unterstützt werden, eingespielt werden, um diese auf einem aktuellen Stand zu halten. Mit einer steigenden Anzahl an unterstützten Konfigurationen kann dies zu einem erheblichen Wartungsaufwand führen. Testframeworks sollten daher Möglichkeiten vorsehen, diesen Wartungsaufwand so gering wie möglich zu halten.
- **Kosteneffizienz:** Betrieb und Wartung eines Testframeworks kann mit zunehmender Komplexität (d.h. steigender Anzahl an unterstützten Systemkonfigurationen) signifikante Kosten verursachen, da mitunter umfangreiche Hardware-Ressourcen benötigt werden. Testframeworks sollten dementsprechend so ausgelegt werden, dass Ressourcen möglichst effizient und sparsam genutzt und Kosten gespart werden.

Im Laufe der letzten Jahre wurden diverse Testframeworks entwickelt, die zum Ziel haben, Entwicklerinnen und Entwickler bei der Erstellung komplexer Softwarelösungen zu unterstüt-

zen. Ob und inwieweit diese Frameworks in der Lage sind, die hier definierten Anforderungen zu erfüllen, wird im folgenden Kapitel näher untersucht.

3 Bestehende Lösungen

Das systematische Testen von Softwarekomponenten zur Qualitätssicherung und Qualitätsverbesserung ist keine neue Thematik und betrifft nicht nur den speziellen Bereich des E-Government. Aus diesem Grund existieren bereits eine Vielzahl an Strategien und Werkzeugen, die sich mit dem Thema des qualitativen und automatisierten Testens beschäftigen. Im Rahmen dieses Kapitels werden daher überblicksmäßig existierende Lösungen vorgestellt und deren Tauglichkeit für einen Einsatz zum Testen von E-Government-Komponenten analysiert.

3.1 Überblick

Im Folgenden wird ein kurzer Überblick über existierende Systeme und Frameworks gegeben, die das Testen Web-basierter Anwendungen auf unterschiedlichen Client-Systemen erlauben.

3.1.1 Selenium

Selenium [HoKe06] ist ein auf HTML und JavaScript basierendes Testframework, welches speziell für das Testen von Web-basierten Anwendungen optimiert ist. Wesentliche Idee dabei ist die Aufzeichnung und automatische Wiederholung von Benutzerinteraktionen. Nebenbei bietet Selenium noch die Möglichkeit, Testfälle in unterschiedlichen Programmiersprachen wie Java, PHP oder Perl zu schreiben, welche gegen die meisten modernen Web-Browser durchgeführt werden können. Aufgrund seines Web-basierten Kerns ist Selenium betriebssystemunabhängig und kann daher auch mit den meisten Browsern verwendet werden.

3.1.2 TestingBot

TestingBot[1] setzt ein automatisiertes Testframework mittels des zuvor vorgestellten Frameworks Selenium in der Cloud um. Dabei können Webseiten rund um die Uhr bezüglich Browser-Kompatibilität getestet werden. TestingBot bietet dabei 96 Browser-Betriebssystem-Kombinationen in der Cloud an und unterstützt unterschiedliche Plug-Ins wie z.B. für PHP oder Ruby. Testfälle können einfach über das Selenium IDE Add-On erstellt und in das Test-Lab von TestingBot importiert werden. Wesentlicher Vorteil dieser Lösung ist die Möglichkeit, unterschiedliche Browser-Betriebssystem-Kombinationen parallel in der Cloud zu testen.

3.1.3 testCloud

Im Gegensatz zu den zuvor erwähnten Lösungen verfolgt testCloud[2] einen kontroversen Ansatz. Dabei werden Tests über eine Crowd von menschlichen Testern letztendlich manuell durchgeführt. Im Gegensatz zu technologiebasierten Cloud-Lösungen werden in diesem Fall jedoch nicht nur IT-Ressourcen, sondern auch „echte“ Menschen und „echte“ Geräte zum Testen bei Bedarf bereitgestellt. TestCloud bietet explorative Tests sowie Testfälle für alle Browser und Betriebssysteme an. Von den „gemieteten“ Testern gefundene Fehler werden über spezielle Bug-Report-Tools wie z.B. Jira[3] oder Redmine[4] bereitgestellt und können von

[1] http://testingbot.com

[2] https://www.testcloud.de

[3] http://www.atlassian.com/software/jira/overview

Kundinnen und Kunden einfach exportiert werden, um diese an die entsprechenden Entwicklerinnen und Entwickler weiterzuleiten.

3.1.4 Virtuelles Testframework für E-Government-Komponenten

Das Virtuelle Testframework für E-Government-Komponenten (VT-EGOV) [ZeKZ11] stellt bereits einen fundamentalen Ansatz dar, um E-Government-Komponenten sicher und zuverlässig zu testen. Dieser Ansatz verwendet als Basis virtuelle Maschinen, auf denen unterschiedliche Betriebssysteme, Browser und E-Government-Komponenten installiert werden können. Zentrale Idee dabei ist, der Dynamik sich ständig verändernder Umgebungen von Client-Systemen gerecht zu werden und somit ein effizientes Testen von sicherheitskritischen Komponenten zu ermöglichen.

3.2 Analyse

Im Rahmen dieses Unterabschnitts werden die zuvor beschriebenen Test-Frameworks auf deren Tauglichkeit hinsichtlich der in Abschnitt 3 beschriebenen Anforderungen analysiert. Tabelle 1 gibt einen prägnanten Überblick, inwiefern die einzelnen Anforderungen von den vorgestellten Tools erfüllt werden können. Ein Pluszeichen „+“ bedeutet dabei, dass die Anforderung erfüllt werden kann, ein Minuszeichen „-“, dass sie nicht erfüllt werden kann.

Tabelle 1 - Analyse bestehender Test-Systeme

Anforderung	Selenium	TestingBot	testCloud	VT-EGOV
Dynamische Adaptierbarkeit	-	-	-	+
Effiziente Verwaltung	-	+	-	+
Zentrale Verfügbarkeit	-	+	+	+
Nachvollziehbarkeit	+	+	+	+
Kompatibilität mit Chipkartentechnologie	-	-	-	+
Kompatibilität zu Java	-	-	-	+
Skalierbarkeit	-	+	-	-
Einfache Wartung der Testumgebungen	-	+	-	-
Kosteneffizienz	-	+	+	-

[4] http://www.redmine.org

Die Anforderung der dynamischen Adaptierbarkeit kann im Wesentlichen nur vom VT-EGOV-Framework erfüllt werden, da hier einfach und schnell neue Systemkonfigurationen erstellt werden können. Die anderen Tools sind hingegen zu allgemein für Browser-basierte Anwendungen gehalten, sodass die gewünschte Flexibilität nicht erreicht werden kann. TestingBot bietet zwar über 96 Browser-Betriebssystem-Kombinationen an, jedoch kann keine individuelle Kombination erstellt werden. Eine effiziente Verwaltung ist nur bei zwei der vorgestellten Lösungen gegeben (TestingBot und VT-EGOV), da nur diese mit der Möglichkeit ausgestattet sind, mit zahlreichen Browser-Betriebssystem-Kombinationen effizient umzugehen. Eine zentrale Verfügbarkeit des Testsystems ist mit allen Systemen bis auf Selenium möglich. Dieses Framework alleine, ohne weiteres Rahmenwerk wie beispielsweise bei TestingBot, ist speziell nur auf die Konfiguration jenes Rechners, auf dem Selenium installiert ist, zugeschnitten. Alle vorgestellten Systeme erfüllen das Kriterium der Nachvollziehbarkeit. Das heißt, dass durchgeführte Tests und Testkonfigurationen jederzeit reproduzierbar sind und Test einfach wiederholt werden können. Eine Kompatibilität zu Chipkartentechnologien ist hingegen nur beim VT-EGOV gegeben, da die anderen Systeme keine Möglichkeit anbieten, Zugriffe auf Kartenlesegeräte für Chipkarten in ihr Test-Framework einzubinden. Eine Kompatibilität zu Java ist ebenfalls nur beim VT-EGOV gegeben, da dieses System das einzige ist, welches erlaubt, Java in das Test-Framework einzubinden. Alle anderen Systeme sind im Wesentlichen für reine Web-Anwendungen optimiert.

Skalierbar ist im Wesentlichen nur die vorgestellte Cloud Lösung TestingBot, da hier das Cloud Computing-Paradigma vollständig ausgenutzt werden kann und mehrere Tests parallel abgearbeitet werden können. Benötigte IT-Ressourcen wie z.B. Arbeitsspeicher spielen dabei keine Rolle, wohingegen die anderen Lösungen in diesem Bereich an ihre Grenzen stoßen können. Auch das Kriterium der einfachen Wartung kann nur von der vorgestellten Cloud-Lösung erfüllt werden, da sich hier der Cloud-Anbieter um die unter dem Test-Framework liegende Infrastruktur kümmert. Bei allen anderen Lösungen muss die Wartung selbständig von den Entwicklerinnen und Entwicklern oder Administratorinnen und Administratoren, welche für das Testsystem zuständig sind, übernommen werden. Letztendlich ergibt sich die beste Kosteneffizienz bei der Cloud-Lösung TestingBot und bei der Crowd-Lösung testCloud. Bei beiden Lösungen ist es möglich, nur die wirklich benötigten Ressourcen anzufordern. Dadurch werden auch nur die wirklich konsumierten Leistungen verrechnet, was wiederum die Kosteneffizienz sicherstellt.

Insgesamt kann festgehalten werden, dass Lösungen basierend auf virtuellen Testumgebungen (VT-EGOV) bzw. Cloud-basierte Lösungen klare Vorteile gegenüber anderen Ansätzen aufweisen. Im folgenden Kapitel zeigen wir, wie durch eine Kombination dieser beiden Ansätze ein Testframework geschaffen werden kann, das allen in Kapitel 2 definierten Anforderungen genügt.

4 Cloud-basiertes Testframework

Testlösungen im Bereich des E-Government müssen besonderen und vor allem sich ständig verändernden Anforderungen genügen. In Österreich wurde zu diesem Zweck in den vergangenen Jahren ein virtuelles Testframework basierend auf der in Kapitel 3 vorgestellten Lösung VT-EGOV eingesetzt. Im praktischen Einsatz stellte sich jedoch heraus, dass dieses Testframework trotz des vielversprechenden Ansatzes einige Mängel aufweist, was sich vor allem in der mangelhaften Skalierbarkeit des Systems widerspiegelt. Um hier eine Verbesserung der Situation herbeizuführen und die vorhandene zentrale virtuelle Lösung an die flexib-

len sich ständig ändernden Anforderungen anzupassen, wurde die bestehende Lösung adaptiert und erweitert, um ein einfaches und effizientes Testen von E-Government-Komponenten zu ermöglichen. Die resultierende Lösung, welche den Ansatz virtueller Testframeworks weiterverfolgt und diesen um Aspekte des Cloud Computing erweitert, wird in diesem Kapitel näher vorgestellt.

Die Architektur der vorgeschlagenen Lösung wird anhand eines konkreten in Österreich im produktiven Einsatz befindlichen Testframeworks veranschaulicht. Dieses Framework verwendet Komponenten und Module des Unternehmens VMware[5]. Das grundlegende Konzept des vorgestellten Testframeworks ist jedoch nicht auf Komponenten dieses speziellen Herstellers beschränkt, sondern kann prinzipiell mit Hilfe beliebiger virtueller Cloud-Infrastrukturen umgesetzt werden.

Grundidee des hier vorgestellten Frameworks ist es, Benutzerinnen und Benutzern entsprechend dem VT-EGOV Konzept vorkonfigurierte Testumgebungen in Form virtueller Maschinen zur Verfügung zu stellen. Als Basis für die virtuelle Architektur dienen bei der hier vorgestellten Implementierung VMware ESXi Server [VMwS13], die ein gemeinsames Set von virtuellen Maschinen anbieten. Aufbauend darauf stellt VMware vCenter Server [VMwC13] eine zentrale Plattform für das Management der gesamten virtuellen Infrastruktur dar. Die Verwaltung der einzelnen für Testzwecke vorgesehenen virtuellen Maschinen wird über die Komponente vCloud Director [VMwD13], die Benutzerinnen und Benutzern einen rollenbasierten Zugang über eine Web-Konsole bietet, durchgeführt.

Die verwendeten VMware-Produkte bilden eine Private Cloud und ermöglichen so eine gemeinsame Nutzung der Infrastruktur. In Abb. 1 werden die wichtigsten Komponenten der virtuellen Cloud-Infrastruktur gezeigt. Sowohl ESXi Server als auch vCenter Server sind für Endbenutzerinnen und Endbenutzer transparent und nur für Administratorinnen und Administratoren des Testframeworks sichtbar. Zentraler Zugangspunkt für Benutzerinnen und Benutzer ist die vCloud Director Komponente.

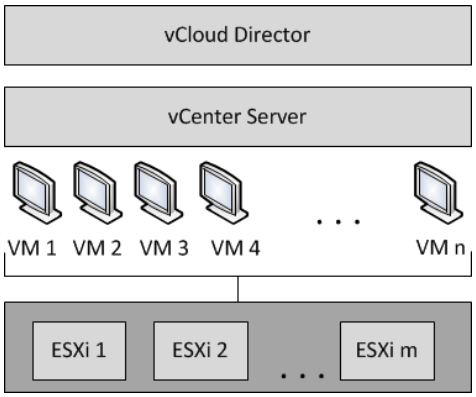

Abb. 1: Verwendete Architektur

Die grundlegende Architektur, die für die Bereitstellung des vCloud Directors notwendig ist, ist in Abb. 2 dargestellt. Benutzerinnen und Benutzer erreichen den vCloud Director über einen Web-Browser. Mehrere vCloud Director Server-Instanzen können bei Bedarf breitgestellt werden. Diese verwenden eine gemeinsame Datenbank und verbinden sich mit einem oder mehreren vCenter Servern.

[5] http://www.vmware.com

Zentrale Einheit bei der Verwaltung virtueller Maschinen über den vCloud Director sind sogenannte Organisationen. Eine Organisation ist eine Verwaltungseinheit, die Benutzerinnen und Benutzer, Gruppen und Rechenressourcen umfasst. Eine solche Einheit wird von Organisationsadministratorinnen und Organisationsadministratoren verwaltet.

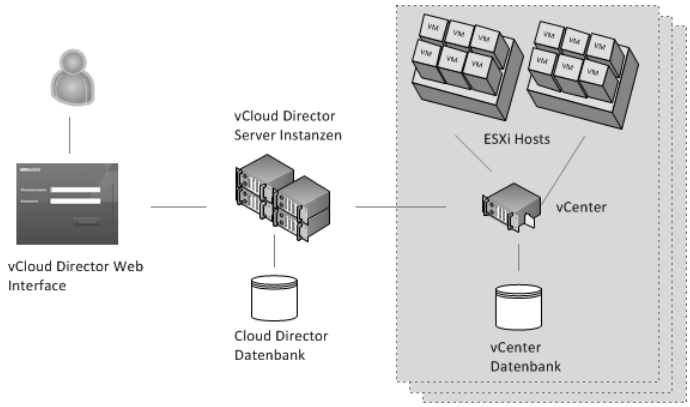

Abb. 2: Architektur des vCloud Directors

Administratorinnen und Administratoren können Organisationen Ressourcen wie Netzwerke, CPU-Kerne und Speicher zur Verfügung stellen. Dabei werden Ressourcen zu entsprechenden Klassen zusammengefasst. Abb. 3 illustriert dies für die vordefinierten Klassen Gold und Silber. Die Gold-Klasse beinhaltet CPUs mit erweiterten Befehlssätzen für kryptographische Operationen und Solid-State Disks für bessere Performance. Virtuelle Maschinen und vApps – Ansammlungen von virtuellen Maschinen – mit erhöhten Anforderungen können sehr einfach der entsprechenden Klasse zugewiesen werden. Intern werden die vorhandenen Ressourcen dynamisch und entsprechend der jeweiligen Klasse auf die zugewiesenen vApps aufgeteilt, wodurch eine effiziente Nutzung von Hardware-Ressourcen gewährleistet wird.

Eine weitere Effizienzsteigerung bei der Verwendung von Ressourcen kann durch das Konzept des Over-Provisioning erreicht werden. Dabei werden vorhandene Ressourcen mehrfach an Organisationen vergeben. Die Ressourcennutzung der Organisationen wird stetig überwacht, um bei Engpässen entsprechend rasch reagieren zu können. Dadurch ist es möglich, Ressourcen anhand des Nutzungsverhaltens der Organisationen entsprechend nur bei wirklichem Bedarf zu kaufen. Frühere Systeme wurden oft überdimensioniert und Ressourcen blieben dadurch ungenutzt.

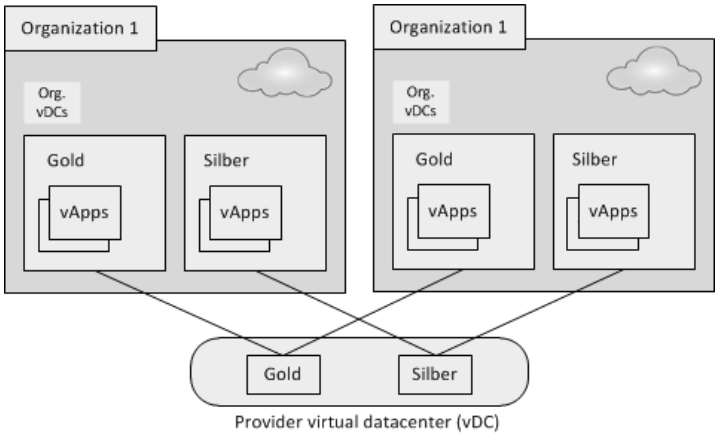

Abb. 3: Organisationen eines vCloud Directors

Benutzerinnen und Benutzer können über die Web-Konsole mit den Ressourcen der ihnen zugeteilten Organisation interagieren und neue virtuelle Maschinen und vApps erzeugen, verwenden und verwalten. Vordefinierte virtuelle Maschinen und vApps, die in typischen Testszenarien im E-Government-Bereich verwendet werden, sind in Katalogen gespeichert, die bei der Erstellung virtueller Maschinen als Vorlage dienen können. Je nach zugeteilter Rolle und Bedarf kann eine Benutzerin oder ein Benutzer neue Kataloge anlegen, vApps erstellen, vApps vom Katalog in die eigene Cloud hinzufügen, bestehende vApps der zugeteilten Gruppe starten, oder nur speziell für die Benutzerin oder den Benutzer vordefinierte vApps verwenden. Die virtuellen Maschinen werden in der Cloud gestartet. Der Zugriff auf die gestarteten Maschinen erfolgt über die Web-Browser-Konsole oder über Remote-Desktop-Protokolle. Bei der Bereitstellung der virtuellen Maschinen wird bereits festgelegt, wie lange vApps maximal ausgeführt werden können. Damit kann verhindert werden, dass nicht verwendete vApps unnötigerweise Ressourcen verbrauchen.

5 Evaluierung

Durch die Kombination des Ansatzes virtueller Testsysteme mit den Konzepten des Cloud Computing erreicht die hier vorgestellte Lösung im Vergleich zu anderen Ansätzen ein höheres Maß an Flexibilität und Funktionalität. Um die Tauglichkeit der hier vorgestellten Lösung im Kontext des E-Government zu evaluieren, wird diese im Folgenden hinsichtlich der im Kapitel 2 definierten Anforderungen analysiert.

- **Dynamische Adaptierbarkeit:** Durch den Einsatz von Cloud Computing und damit verbundener zentraler Lösungen, kann der Aufwand für die Zusammenstellung neuer Systemkonfigurationen gering gehalten werden. Benutzerinnen und Benutzer können aus einer großen Anzahl an virtuellen Maschinen eine eigene Testumgebung dynamisch erstellen. Dafür werden vordefinierte vApps, die eine oder mehrere virtuelle Maschinen beinhalten können, herangezogen. Typische E-Government-Testszenarien können durch diese vorkonfigurierten virtuellen Maschinen rasch nachgestellt werden. Die Zusammensetzung der vApps beginnt bei Systemen mit neuinstallierten Betriebssystemen, reicht über Systeme mit vorinstallierten Web-Browsern und geht bis hin zu Systemen mit speziellen vorinstallierten Softwarekomponenten aus dem E-Government-Bereich.
- **Effiziente Verwaltung:** Die VMware-Komponente vCloud Director bietet umfangreiche Mechanismen für die Verwaltung komplexer Testlandschaften. Virtuelle Maschinen können dabei sehr schnell benutzerspezifischen vApps hinzugefügt, verlinkt und wieder entfernt werden. Eine dynamische Erweiterung der Testumgebung durch neue VMs ist einfach möglich. Der praktische Einsatz dieser Lösung zeigt, dass die vordefinierten Systeme, die typische Testszenarien aus dem E-Government-Bereich repräsentieren, Benutzerinnen und Benutzern viel Zeit bei der Erstellung der gewünschten Testumgebung ersparen. Die integrierte Benutzerverwaltung und das umfangreiche Rechtemanagementsystem können dynamisch angepasst werden. Je nach Anforderung besteht so die Möglichkeit, Benutzerinnen und Benutzern entsprechende Rechte zuzuweisen.
- **Zentrale Verfügbarkeit:** Eines der Hauptmerkmale von Cloud Computing ist die zentrale Verfügbarkeit. Durch die Integration von Cloud Computing-Aspekten erfüllt auch das hier vorgestellte Testframework diese Anforderung vollständig. Der Einsatz dieser Technologie ermöglicht eine gemeinsame Nutzung von Ressourcen. Sowohl

das Testframework als Ganzes, als auch einzelne virtuelle Maschinen können von mehreren Benutzerinnen und Benutzern gleichzeitig verwendet werden.

- **Nachvollziehbarkeit**: Jede Benutzerin und jeder Benutzer besitzt seine eigene Test-Cloud mit eigenen vApps. Bestimmte Systemkonfigurationen, die für weitere Tests und eine spätere Nachvollziehbarkeit nützlich sind, können längerfristig gespeichert oder als Basiskatalog für alle Benutzerinnen und Benutzer zu Verfügung gestellt werden. Testsysteme können somit jederzeit einfach und rasch aus Basiskatalogen geklont werden, wodurch die spätere Nachvollziehbarkeit von Tests gewährleistet ist.
- **Kompatibilität mit Chipkartentechnologie:** Benutzerinnen und Benutzer können sich zu einer virtuellen Maschine über Remote-Desktop-Protokolle verbinden, welche eine Weiterleitung von USB-Geräten unterstützen. Somit ist gewährleistet, dass auch USB-Kartenlesegeräte am Testsystem verwendet werden können. Auf diese Weise können auch Tests mit Chipkarten sehr einfach durchgeführt werden.
- **Kompatibilität mit Java:** Java-Technologien können nach Belieben in das Testsystem eingebunden werden. Benutzerinnen und Benutzer können innerhalb einer virtuellen Maschine jede beliebige Technologie verwenden, die für die Durchführung der Tests notwendig ist.
- **Skalierbarkeit**: Durch die dahinterliegende Cloud-Infrastruktur ist das Testsystem in jeder Hinsicht beliebig erweiterbar. Sowohl Hardware- als auch Softwarekomponenten können dem bestehenden System jederzeit hinzugefügt werden.
- **Einfache Wartung der Testumgebung:** Sowohl der Wartungsaufwand für Administratorinnen und Administratoren des Systems über den zentralen vCenter Server als auch die Wartung der vApps jeder einzelnen Benutzerin bzw. jedes einzelnen Benutzers kann effizient und teilweise automatisch durchgeführt werden. Der zentrale Ansatz und die gemeinsam genutzte Infrastruktur begünstigt eine kostengünstige und einfache Wartung.
- **Kosteneffizienz**: Das Testsystem besteht aus einer virtuellen Private Cloud-Lösung, die geringe Infrastrukturkosten hat. Storage-Kosten sind die einzigen Hardwarekosten, die bei dieser Lösung anfallen. Die Linked Clone-Technologie [VMwD13] ermöglicht das Klonen von Basis-vApps in untergeordnete vApps, indem nur die Änderungen gespeichert werden, die von den untergeordneten vApps stammen, während die restlichen Daten aus den Basis-vApps verwendet werden. Durch den Einsatz dieser Technologie können Storage-Kosten reduziert werden. Durch die einfache Bereitstellung der virtuellen Maschinen und die effiziente Wartung der Testumgebung können ebenfalls Kosten gespart werden.

6 Fazit

Umfangreiche und systematische Tests stellen vor allem auch bei sicherheitskritischen IT-Lösungen eine wichtige Methode zur Gewährleistung der Einhaltung gegebener Qualitätsstandards dar. Vor allem im Bereich des E-Government zeigt die langjährige Erfahrung, dass die Durchführung systematischer Tests durch spezielle Anforderungen von E-Government-Lösungen oftmals schwierig und aufwändig ist. Bisher vorgestellte und etablierte Testframeworks können die zahlreichen Anforderungen oft nur teilweise und auf unbefriedigende Art und Weise erfüllen.

Um dieser Problematik geeignet zu begegnen, wurde in diesem Artikel ein neuartiges Konzept zur Durchführung effizienter und effektiver Tests im Bereich des E-Government vorge-

stellt. Dieses Konzept kombiniert Aspekte etablierter virtueller Testsysteme mit jenen des Cloud Computing und erlaubt so die Entwicklung eines Testframeworks, das in der Lage ist, allen Anforderungen an E-Government-Testsysteme zu genügen. Die praktische Umsetzbarkeit der hier vorgestellten Lösung wurde anhand einer konkreten Implementierung positiv evaluiert. Diese Implementierung basiert auf Virtualisierungskomponenten und Cloud-Lösungen der Firma VMware. Das vorgeschlagene und in diesem Artikel im Detail diskutierte Konzept ist jedoch prinzipiell auch über beliebige andere Komponenten mit entsprechender Funktionalität umsetzbar. Unabhängig von der gewählten Umsetzung bietet das vorgeschlagene Konzept Entwicklerinnen und Entwicklern von E-Government-Komponenten die Möglichkeit, bereitgestellte Lösungen effektiven und effizienten Tests zu unterziehen und dadurch geforderte Qualitäts- und Sicherheitsmerkmale einzuhalten.

Literatur

[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 64-69, (2010)

[EGov04] E-Government-Gesetz (E-GovG), Bundesgesetzblatt für die Republik Österreich BGBl.I Nr. 10/2004, (2004)

[EP99] Richtlinie1999/93/EG des Europäischen Parlaments und des Rates vom 13. Dezember 1999 über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen, http://eurlex.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 regresion testing https://h10078.www1.hp.com/cda/hpms/display/main/hpms_-content.jsp?zn=bto&cp=1-11-127-24^1322_4000_100_, (2011)

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

[ISO11] ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. http://www.iso.org/iso/home/store/catalogue_ics/catalogue_-detail_ics.htm?csnumber=35733, (2011)

[VMwC13] VMware vCenter Server, http://www.vmware.com/products/vcenterserver/-overview.html, (2013)

[VMwD13] VMware vCloud Director, Key Features and Functionality, http://www.vmware.com/products/vcloud-director/features.html, (2013)

[VMwS13] VMware vSphere ESX and ESXi Info Center, http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html, (2013)

[ZeKZ11] T. Zefferer, V. Krnjic, B. Zwattendorfer: Ein Virtuelles Testframework für E-Government Komponenten. In D-A-CH Security 2011, (2011)