profi.com®
we make IT work

CASE STUDY | SERVICE VIRTUALISIERUNG

Im Zeitalter des »IOT – Internet of Things« gewinnen in unserer Gesellschaft Begriffe wie EAI[1] , SOA[2] , BPM[3] und Microservices immer mehr an Bedeutung. Das Resultat ist eine immer vernetztere Welt.

Damit geht einher, dass die Entwicklung und der Betrieb von Anwendungen nicht ohne Schnittstellen – respektive Kommunikation mit der »Außenwelt« – auskommen wird. Außerdem werden performante Testumgebungen und dadurch gewährleistete Testdurchführungen zwingend notwendig, um eine garantierte Qualität der Software zu gewährleisten.

Dies gilt im Besonderen für End-To-End-Tests, da diese sehr viele vom Testfall genutzte heterogene Systeme tangieren. Da die Verfügbarkeit dieser Umsysteme, Services und Schnittstellen nicht immer im eigenen Hoheitsbereich liegt, stellt dies eine enorme Herausforderung für Entwicklungs-, Betriebs- und Testteams dar.

Eine Möglichkeit, die Bereitstellung zu gewährleisten, bietet das Konzept der Service Virtualisierung. Diese Lösung wird auf einfache und effiziente Weise den Bedarf der oben beschriebenen Anforderungen in den kommenden Jahren[4] decken.

Was ist Service Virtualisierung?

Service Virtualisierung beschäftigt sich mit der Bereitstellung von Services in einer heterogenen IT-Systemlandschaft, die gerade nicht verfügbar sind. Ein »Service« im IT-Umfeld umschreibt die Bereitstellung von Dienstleistungen, z.B. Daten, Informationen oder Berechnungen in verteilten Rechenarchitekturen. Die Kommunikation zwischen den Services erfolgt zumeist von Maschine zu Maschine über definierte Schnittstellen und Protokolle. Beispiele für Services sind Web-Services, WCF-Services, Java-Services.

Virtualisierung[5] bezeichnet die Nachbildung eines Objekts (in unserem Fall Soft- und Hardwarekomponenten) durch ein ähnliches Objekt vom selben Typ mit Hilfe einer Softwareschicht. Dadurch können Services »vorgetäuscht« werden, die gerade nicht vorhanden sind. Den Begriff Virtualisierung möchten wir abgrenzend von den Begrifflichkeiten Simulation[6] und Emulation[7] verwenden, welche Ähnlichkeiten aufweisen, allerdings ebenso Unterschiede.

Eine typisch heterogene Systemlandschaft ist in Abbildung 1 dargestellt. Die Beispielanwendung soll per Desktop-PC und mobilem Endgerät verfügbar sein.

Typischerweise entspricht eine solche Anwendung den gegenwärtigen Sicherheitsstandards über Firewall, Userbeschränkung und ähnliches. Die IT-Landschaft der Anwendung besitzt Web-Server, Applikationsserver, Datenbankserver und diverse Web-Services. Die Kommunikation soll sich dabei über verschiedenste Protokolle erstrecken, wie z.B. REST, SOAP, LDAP, MQ, JDBC.

Vorzufinden ist eine solche Architektur in großen Unternehmen mit gewachsener komplexer IT-Infrastruktur, wie z.B. in der Finanzbranche. Amazon und PayPal sind weitere Vertreter dieser IT-Landschaft.

Häufig werden nicht alle Services innerhalb der eigenen IT-Systemlandschaft gehostet. In unserem Beispiel ist der »Payment«-Service extern gehostet, d. h. dieser befindet sich nicht in der Obhut des Unternehmens und man ist abhängig von einem externen Anbieter.

Zum Entwicklungsprozess der Beispielanwendung gehören auch Testzyklen. Diese laufen idealerweise parallel zum Entwicklungsprozess ab. In unserer sehr heterogenen IT-Infrastruktur genügt es nicht mehr, allein die Anwendung autark und entkoppelt vom Rest der kompletten IT-Infrastruktur zu testen. Der Testprozess muss Szenarien mit Prüfungen der Kommunikation / Antworten aller genutzten Services, auch denen die extern gehostet sind, umfassen.

Ist allerdings der externe Service temporär nicht erreichbar, so lassen sich für diesen Zeitraum keine End-To-End Tests durchführen. Hier setzt die Service Virtualisierung an.

Diese generiert nun für den Service »Payment« anhand eines zugrundeliegenden Datenmodells vordefinierte Antworten, abhängig von der Anfrage. Damit ist man für den gesamten Testzeitraum unabhängig vom externen Anbieter und kann jederzeit End-To-End Tests durchführen.

Die Service Virtualisierung kann genutzt werden, um sehr viel früher in der Softwareentwicklung übergreifende Komponenten- und Integrationstests durchzuführen und die Testzyklen entscheidend zu verkürzen.

(1) EAI, Enterprise Application Integration, (2) SOA, Service-orientierte Architektur, (3) BPM, Business Process Management, (4) Gartner: »Market Guide for API Testing and Service Virtualization«. July 6, 2015
(5) https://de.wikipedia.org/wiki/Virtualisierung_(Informatik), (6) https://de.wikipedia.org/wiki/Simulation,
(7) https://de.wikipedia.org/wiki/Emulator

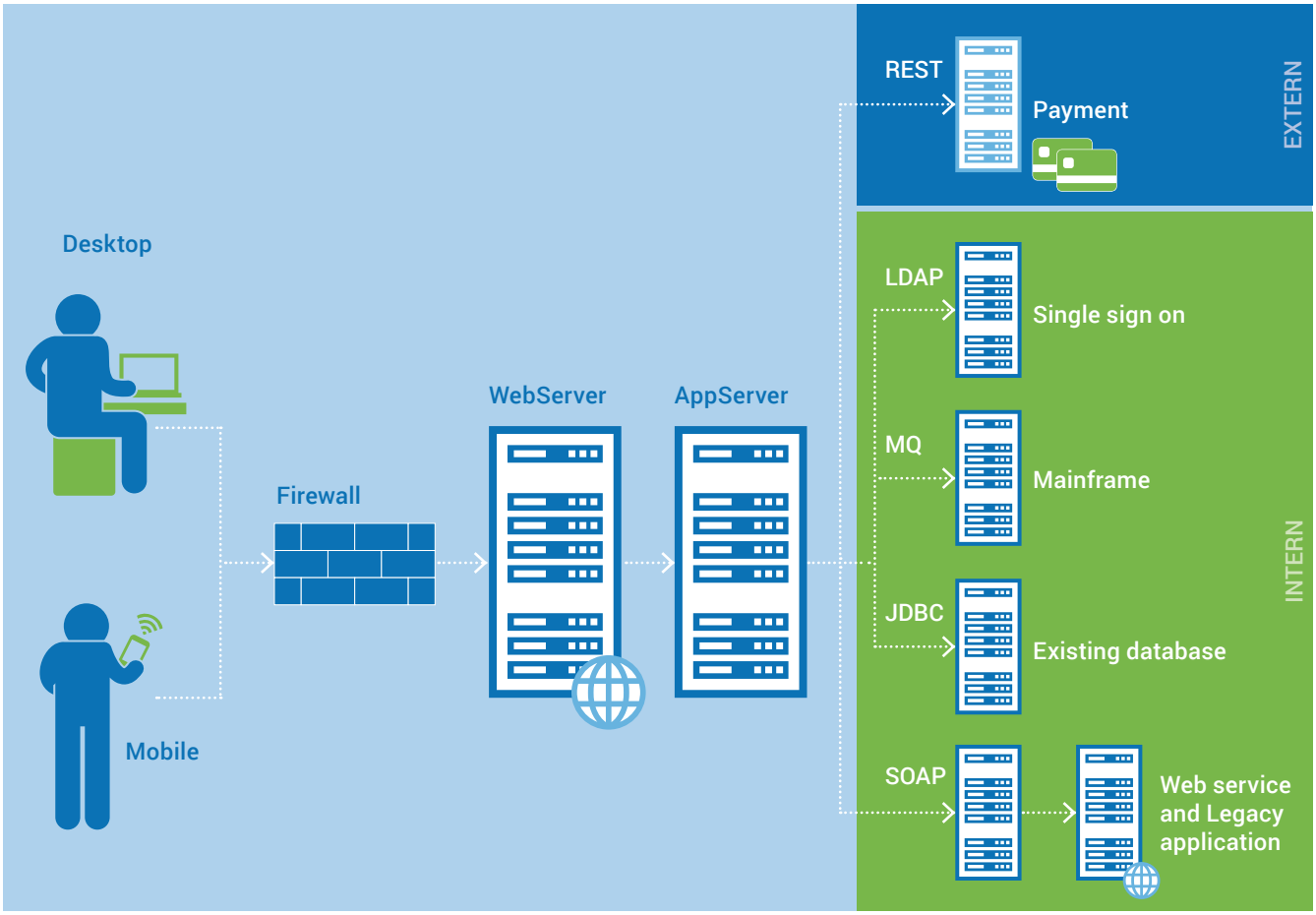

Abbildung 1: Typische Architektur einer heterogenen Systemlandschaft

Abgrenzung zu Mock / Stub

Auf den ersten Blick scheint die Service Virtualisierung nichts Anderes zu sein wie ein gewöhnlicher Mock oder Stub. Das Resultat – für sich allein betrachtet die Antwort auf eine Anfrage – wirkt zunächst gleich. **Die Wirkungsweise der Service Virtualisierung und des Mock/Stub ist jedoch grundlegend verschieden.**

Mocks/Stubs bilden in der Softwareentwicklung eine feste Größe, um Softwarekomponenten für Testzwecke nachzubilden. Es bestehen jedoch einige wichtige Unterschiede im Vergleich zur Service Virtualisierung. Mock/Stub-Funktionen sind sehr kontextspezifisch implementiert. So simuliert der Mock/Stub ein spezifisches Antwortverhalten zu gegebenen Entwicklungsbedürfnissen.

Ein Beispiel ist die Implementierung einer fehlenden Abhängigkeit in einer Client/Serverarchitektur mit dem Ziel, die Abstinenz des Servers für spezifische Unit Tests zu umgehen. Dies kann auch in Isolation vom Rest der »Application under Test« (AUT) erfolgen, um vereinzelte Komponenten der AUT zu testen. Dabei können die Antworten einfache Berechnungen von Währungsumrechnungen, Validätsprüfungen von Kunden oder ähnliches beinhalten.

Die Funktionalität von Mocks und Stubs ist somit sehr eingeschränkt. Reale Systeme damit umfangreich nachzubilden ist sehr aufwändig.

Im Gegensatz dazu können virtuelle Services im gesamten Produktionszyklus genutzt werden. Diese liefern zu jedem Zeitpunkt das gleiche Verhalten und die gleiche Funktionalität, nahezu unabhängig vom Softwarezustand. Sind diese virtuellen Komponenten folglich einmal erstellt und nutzbar, wird das Erstellen und Anpassen von Mocks / Stubs obsolet!

Ein zweiter wichtiger Unterschied ist, dass beim Mock/Stub individuelle Klassen simuliert werden, währenddessen bei der Service Virtualisierung das volle Spektrum (siehe Abbildung 2) der Backend-Services-simuliert werden kann.

Die Antworten des virtuellen Services können beliebig komplex ausfallen, und falls es die Applikation benötigt, kann die virtualisierte Antwort genau der realen entsprechen. Im Gegensatz dazu ist die Lebensdauer bei Mocks/Stubs zeitlich limitiert, da die kontextspezifische Umsetzung meist nur für ein Release funktionsfähig ist.

Aus Perspektive der QA ist zusammenfassend zu sagen, dass sich Mocks/Stubs am besten für Unit Tests eignen. **Service Virtualisierung spielt seine Vorteile hingegen besser bei Integrations-, System- und Performancetests aus.**

Idealerweise bezieht man beide Möglichkeiten mit in den Entwicklungsprozess ein und schöpft Synergieeffekte.

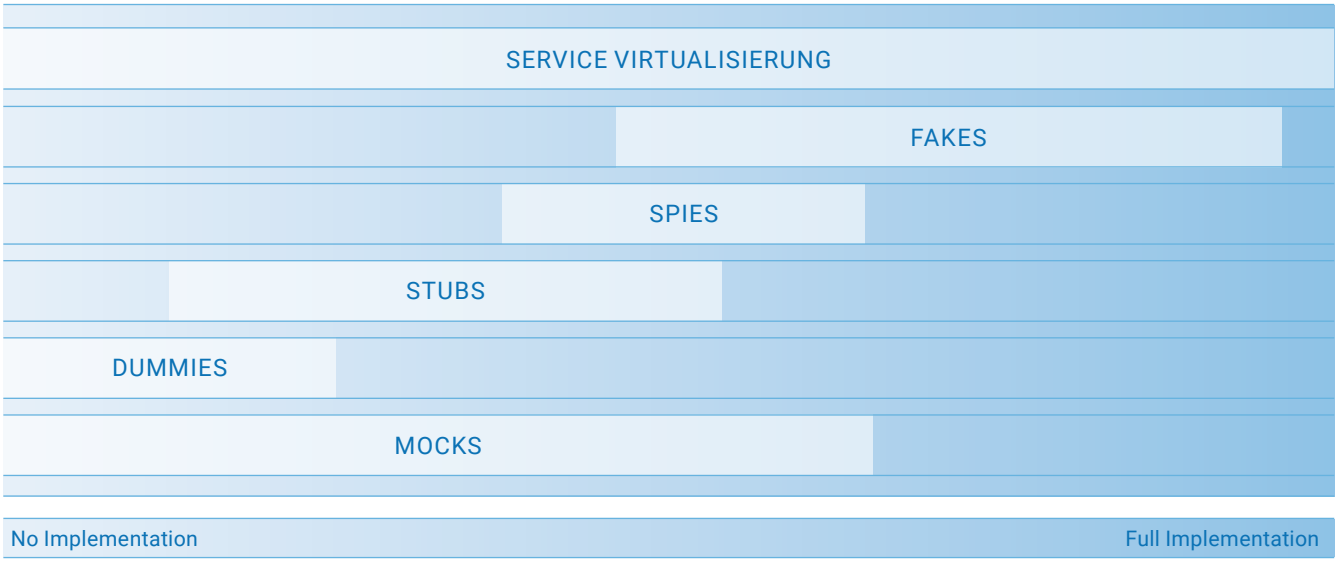

Abbildung 2: Testspektrum

Angrenzende Einsatzmöglichkeiten

Service Virtualisierung bildet die Schnittstelle zwischen klassischen Testdatenmanagementlösungen (TDM) als auch üblichen Netzwerkvirtualisierungslösungen (NV).

Das TDM stellt hierbei Testdaten für einen Testzyklus zur Verfügung. Die Erstellung kann synthetisch (Neu-Anlegen der Daten ohne Vorkenntnisse der Produktionsdaten) oder mittels einer Maskierung (Pseudonymisierung auf Basis eines Live-Abzugs einer im Betrieb befindlichen Datenbank) erfolgen.

NV Lösungen bieten die Möglichkeit, Netzwerke in ihrem realistischen Verhalten nachzubilden. Sei es auf empirisch erhoben Netzwerkstatistiken oder einfach simulierten virtuellen Kenngrößen für das Netzwerk. Bei der Netzwerk-Simulation können Breitbandverbindungen als auch typische Vertreter aus dem Mobilfunkbereich (LTE, HSDPA, UMTS, EDGE, GPRS) mit ins Kalkül gezogen werden.

Seit neuestem bieten einige Hersteller bereits integrierte NV in ihrem Service Virtualisierungs-Paket mit an, andere beinhalten TDM Lösungen. Das Unternehmen Forrester hat hierzu eine Übersichtsarbeit geliefert[8].

(8) The Forrester Wave™: »Service Virtualization And Testing Solutions, Q1 2014« (by Diego Lo Giudice, January 27, 2014)

Tools im Einsatz

Es gibt eine Vielzahl an Tools für Service Virtualisierung auf dem Markt. Bei der Auswahl an Werkzeugen beschränken wir uns auf die gebräuchlichsten Open Source-Löungen und kommerziellen Produkte auf Basis der relevanten Eigenschaften Marktpräsenz, Zukunftsvision, Schnittstellenunterstützung und Referenzen.

Folgende Auswahlliste ist dabei entstanden:

Open Source	Kommerziell
SoapUI [9]	SmartBear ServiceV Pro [17]
Mockito [10]	Micro Focus Service Virtualization [18]
WireMock [11]	CA Service Virtualization [19]
Mountebank [12]	Parasoft Virtualize [20]
Hoverfly [13]	IBM Rational Test Virtualization Server [21]
Mirage [14]	Tricentis Orchestrated Service Virtualization [22]
stubby4j [15]	
Wilma [16]	

Auf Basis der Auswahl wurde entschieden, die Open Source-Tools in einem Abschnitt zusammenzufassen da hier Funktionalität und Marktpräsenz ein nicht so hohes Ausmaß besitzen wie bei den Enterpriseprodukten. Unter diesem Gesichtspunkt wird auch das Tool von Tricentis für unsere Vergleichsstudie nicht weiter betrachtet. Im Folgenden wollen wir die Bewertung der Tools textuell diskutieren und in Form einer Abschlussnote bewerten.

Grundlage für die Bewertung der ausgewählten Tools bildeten folgende Kriterien:

1. Wie einfach lässt sich das Tool installieren?
2. Wie umfangreich ist die Protokollunterstützung?
3. Wie gut funktioniert die Korrelationserkennung und Parametrisierung bei Protokollmitschnitten?
4. Ist ein SDK vorhanden um weitere Protokolle zu implementieren? Wie gut ist dieses SDK?
5. Wie gut ist der Support bei Problemen?
6. Wie gut ist die Hilfe im Tool?
7. Wie groß ist die Anzahl an unterstützten Betriebssystemen?
8. Wie einfach gestaltet sich das Lizenzmodell?
9. Wie gut ist die Usability des Tools?
   a. Intuitivät der GUI?
   b. Wartbarkeit von existierenden Services?
10. Möglichkeiten der Integration in andere Tools
    a. Testdatenmanagementlösungen
    b. Build Managementsysteme
11. Wie weitreichend ist die Roadmap gestaltet?

(9) https://www.soapui.org/ (10) http://mockito.org/ (11) http://wiremock.org/ (12) http://www.mbtest.org/ (13) http://hoverfly.io/
(14) https://github.com/lashd/mirage (15) https://github.com/azagniotov/stubby4j (16) https://github.com/epam/Wilma
(17) https://smartbear.com/product/ready-api/servicev/overview/ (18) http://www8.hp.com/de/de/software-solutions/service-virtualization/
(19) http://www.ca.com/us/products/ca-service-virtualization.html (20) https://www.parasoft.com/solution/service-virtualization/
(21) http://www-03.ibm.com/software/products/de/rtvs (22) http://www.tricentis.com/de/tricentis-tosca-testsuite/orchestrated-service-virtualization/

OPEN SOURCE TOOLS

In diesem Marktsegment sind eher die Spezialisten zuhause als die Generalisten. Das heißt, im Gegensatz zu den kommerziellen Produkten unterstützen diese Tools nur wenige Protokolle je Tool. Um damit eine vollständige Produktpalette in einem mittelgroßen Unternehmen abzubilden (SAP, WebServer, JDBC, LDAP u.a.) benötigt man gegebenenfalls mehrere Tools.

Die Open-Source Tools bieten die Möglichkeit, sofort mit der Installation und Nutzung zu starten, ohne sich vorab über eine geeignete Lizensierung Gedanken machen zu müssen. Die Anzahl an gebotenen Tools ist groß und das richtige Tool zu finden kann entsprechend länger dauern. Die nächste Hürde ist der Download der Source. Manche Sourcefiles lassen sich in GitHub finden, manche auf beschwerlichem Weg mittels Suche im Internet. Dokumentation und Support ist für diese Art an Tools auch nicht in dem Maße erhältlich wie für die kommerziellen Produkte, was allerdings auch nicht verwunderlich ist. Fazit ist, dass diese Art von Tools für klein- bis mittelständige Unternehmen oder einzelne Abteilungen geeignet sind, die sehr entwicklungsnah arbeiten und experimentierfreudig sind. **Als standardisierte Plattform für den Einsatz in einem großen Unternehmen ist keine der betrachteten Open Source-Lösungen zur Virtualisierung von Services derzeit geeignet.**

CA SERVICE VIRTUALIZATION

CA bildet neben IBM und Parasoft einen Platzhirsch im Bereich der Service Virtualisierung. Mit der Übernahme der Firma ITKO etablierte sich CA auf diesen Marktbereich im Jahr 2011. Das zeigt sich in einer breiten Unterstützung an Protokollen. Leichte Einschränkungen gibt es allerdings im Bereich SAP. Bei fehlender Unterstützung von Protokollen im nicht-SAP Bereich werden diese durch Partner seitens CA angeboten.

Die Oberfläche von CA Service Virtualization wirkt sehr ausgereift und ist sehr schnell intuitiv erlernbar. Hier fällt positiv auf, dass sehr viel Anwenderfeedback in die Produktweiterentwicklung geflossen ist. Des Weiteren besticht CA Service Virtualization durch das vollständigste Spektrum an unterstützten Features unter den verglichenen Tools. **Als Stärken sind hier die automatischen Agenten für Protokollanalyse (Korrelation, Parametrisierung, Protokollerweiterung) genannt.** Diese ermöglichen auf einfache Weise das Dynamisieren des Services. Alle anderen Features, die man auch bei den anderen Herstellen findet, sind auch hier anzutreffen. Dazu zählt die Möglichkeit, virtualisierte Services auch scripten zu können und Testdaten zu verwalten. Auch ein Modus zum Erlernen eines Services wird geboten. Integrationen in eine Entwicklungslandschaft über einen Buildserver wie Jenkins/Hudson stellen CA vor keine Probleme. Zusammenfassend überzeugt CA Service Virtualization mit Funktion und Design. Einzig das Nichtbereitstellen einer Demolizenz als auch das Nichtbereitstellen einer »Community«-Edition zum sanften Einstieg sind negativ anzumerken.

NOTE 1.4

1 2 3

MICRO FOCUS SERVICE VIRTUALIZATION

Neben SmartBear zählt auch Micro Focus im Bereich Service Virtualisierung zu den jüngeren Marktteilnehmern. Dies stellt sich vor allem in der lückenhaften Unterstützung von älteren Protokollen wie beispielsweise LDAP, DB2, (S)FTP, MS Active Directory dar. Andererseits werden neuere Protokolle wie JMS, SOAP, REST oder diverse Message Queue-Lösungen (WebSphere MQ; ActiveMQ, OpenMQ, MQTT) sehr gut unterstützt.

Micro Focus besticht vor allem in den Punkten Design und intuitiver Bedienung. Mittels der bereitgestellten Agenten können – auch ohne große Vorkenntnisse in der Service Virtualisierung – sehr schnell Ergebnisse erzielt werden. Reichen die Micro Focus-eigenen Bordmittel einmal nicht aus, um das Protokoll vollständig zu unterstützen, so kann man den virtualisierten Service genau an den Punkten wo nötig mittels Programmierung

(C# oder JavaScript) erweitern. Zusätzlich kann mit dieser Funktionalität das Antwortverhalten des Services definiert werden oder geeignete Parser implementiert werden. Diese Leichtigkeit bei der Erstellung der virtuellen Services wird ersichtlich, wenn man die Liste an unterstützenden Funktionen betrachtet. Hier glänzt Micro Focus Service Virtualization mit vielen Standardfunktionen, welche die Erstellung des virtuellen Services sehr einfach gestalten. Neue Agenten für noch nicht unterstützte Transportprotokolle können über ein SDK implementiert werden.

Micro Focus Service Virtualisierung kann derzeit auf Windows- und Linuxsystemen sowie als Docker-Container betrieben werden. Die Verfügbarkeit von Docker-Images ermöglich einen schnelle Bereitstellung der Lösung. Integrationen in Testmanagementsysteme und Continous-Integration-Systeme (Jenkins / Hudson) sind vorhanden. Ein Vorteil ist die direkte Integration in die meisten Micro Focus Tools (z.B. Micro Focus LoadRunner, Micro Focus Performance Center, Micro Focus Unified Functional Testing, Micro Focus Application Lifecycle Management). Durch die Einbindung von Micro Focus Network Virtualization können verschiedene Netzwerktopologien und -geschwindigkeiten simuliert werden. Micro Focus bietet ein gut skalierbares Lizenzmodell als auch eine »Community«-Edition zur kostenfreien Verwendung an. Dies ermöglicht die Lösung unabhängig von Evaluationslizenzen zu prüfen. Ein weiterer Pluspunkt ist die zielstrebige Roadmap seitens Micro Focus und die stetige Erweiterung der Lösung um weitere Protokolle. Damit überzeugt der Gesamteindruck des Tools.

1 2 3

IBM GREENHAT

Ein weiterer Global Player ist IBM. Nach der Übernahme der Firma »Green Hat« im Jahr 2012 hat IBM das Tool beständig weiterentwickelt und zu einem sehr kompletten Tool ausgebaut. So besticht die Suite durch eine Vielzahl an unterstützten Protokollen, hier vor allem zu nennen ist der Bereich des Mainframe (Z-Systeme) und die gute Anbindung an SAP und TIBCO.

IBM glänzt mit Anbindungen an die eigene Rational TestSuite (nativ), als auch sehr gute Anbindung an die TestSuite von Micro Focus. Dies allerdings nicht nativ, sondern über Umwege einer Nachinstallation. Die Toollandschaft bei IBM ist recht verzweigt, so besteht IBM Rational Test Virtualization Server aus mehreren obligatorischen und optionalen Komponenten. Zu den obligatorischen zählen der IBM Rational Integration Tester (Erzeugung der virtuellen Services), IBM Rational Test Control Panel (Verwaltung der virtuellen Services), und den IBM Rational Integration Tester Agent (Laufzeitumgebung für den virtuellen Service). Die Oberfläche des Rational Integration wirkt sehr aufgeräumt und die Erstellung virtueller Services ist grafisch sehr gut unterstützt. Eine volle Flexibilität für die Erstellung der Services, z.B. durch die Unterstützung mittels Skriptsprachen, ist leider nicht gegeben. Dafür bietet IBM eine sehr elegante Lösung von integrierten Service Tests und der Erstellung dieser Services. Das testgetriebene Vorgehen hat den Vorteil, sehr schnell Tests für die Services zu erzeugen und auszuführen.

Eine direkte Anbindung hin zu IBM InfoSphere Optim hinsichtlich der Bereitstellung und Nutzung von Testdaten ist sehr hilfreich. Neben der Jenkins-Anbindung bietet IBM eine direkte und einfache Ankopplung für das tägliche Deployen der Services an IBM UrbanCode.

Ein negativer Aspekt ist, dass IBM keine Demo-Lizenzen anbietet. Die Möglichkeit, das Tool ohne direkten Kontakt zum Hersteller auf seine Bedürfnisse hin selbst einmal auszuprobieren, ist somit nicht gegeben.

1 2 3

PARASOFT VIRTUALIZE

PARASOFT

Parasoft zählt zu den »Dinosauriern« in diesem Business. Das Unternehmen beschäftigt sich seit zirka 20 Jahren mit Lösungen im Bereich Service Virtualisierung, gleichwohl es diesen Begriff noch nicht so lange gibt. Das lässt auf ein ausgereiftes Produkt schließen. Die auf der Eclipse-Plattform basierte Lösung besticht mit einer der vollständigsten Unterstützungen unter den verglichenen Tools. Abstriche gibt es im SAP-Bereich, diese Protokolle werden allerdings über Partner abgebildet. Wenig Unterstützung gibt es im Mainframe- und DB2-Bereich, dafür besticht die Unterstützung von diversen MQ- und FTP-, SFTP-, FIX- und ähnlichen Protokollen. Hier zeichnet sich die Stärke des Tools durch dessen Flexibilität aus. Als Java-basierte Anwendung läuft es auf verschiedensten Systemen wie Linux, Windows, oder Mac OS.

Service Virtualisierung ist ebenso wie bei anderen Konkurrenten »scriptless« möglich, allerdings nicht so intuitiv wie bei CA oder Micro Focus. Die Möglichkeiten zum Lernen eines Services sind bei Parasoft limitiert und es ist mehr manuelle Vorarbeit für die Erstellung eines validen und stabilen Datenmodells notwendig als bei den Konkurrenzprodukten. Ansonsten besticht Parasoft durch eine hervorragende API und Jenkins/Hudson-Anbindungen. Einzig die mäßige Online Tutorial Präsenz ist etwas hinderlich.

NOTE 1.7

1 2 3

SMARTBEAR SERVICEV PRO

SMARTBEAR

Die im Gegensatz zu den anderen Herstellern noch junge Firma SmartBear ist ein Neuling auf dem Gebiet der Service Virtualisierung. Dies äußerst sich vor allem in der geringen Unterstützung an Protokollen, hier vor allem zu nennen der SAP- und MQ-Bereich und einige andere. Standardprotokolle wie HTTP, SOAP u.a. werden allerdings unterstützt. Auch spürt man den Marktneuling, indem man einige Features vermisst die andere Mitbewerber anbieten.

So fehlen beispielsweise Agenten zur Erstellung des Virtuellen Services beziehungsweise werden diese nicht vollständig unterstützt. Das heißt, dass die Generierung virtueller Services ohne Coding-Anteil nur sehr schwer möglich ist. Ebenso wurde ein Lernmodus zum »Anlernen« eines virtuellen Services nur rudimentär implementiert. Verglichen mit den Produkten der Mitbewerber geht das Umstellen zwischen realem und echtem Service nicht so leicht von der Hand. Der typische Anwender sollte demnach im Umgang mit Skriptsprachen sehr vertraut sein.

Eine integrierte Lösung für die dynamische Modellierung von Performanceverhalten (Netzwerk) ist vorhanden. Hier gefällt vor allem die Implementierung von bereitgestellten Standardfehlern, z.B. für HTTP (50x, ...). Die Möglichkeit, nicht standardmäßig unterstützte Protokolle selbstständig zu erweitern (proprietäre Protokolle, o. ä.), sind sehr eingeschränkt.

Usability und inutitive Bedienung sind nicht die Stärken von SmartBear ServiceV Pro. Hier steht Funktionalität ganz klar vor Design. Auch hier punkten eher andere Mitbewerber. Die Hilfsfunktionen im Tool selbst oder online sind sehr stark ausbaufähig. Als einziges von den verglichenen Tools glänzt Service Pro mit einer partiellen Integration in die beiden IDE's Eclipse und Visual Studio.

NOTE 2.7

1 2 3

profi.com®
we make IT work

FAZIT

In den kommenden Jahren wird die Nutzung von Service Virtualisierung in der Soft- und Hardwareentwicklung (IoT) immer mehr zunehmen.

Mit den steigenden Ansprüchen an die Qualität einer Software, der zunehmenden Komplexität und der ständigen Verfügbarkeit von Software wird es unerlässlich sein, Entwicklungs- und Testzyklen ständig und schneller durchzuführen. Dies bedingt den Einsatz diverser Virtualisierungstools.

Ob man sich für einen Mock/Stub oder ein komplexeres Virtualisierungstools entscheidet, hängt vom eigenen Budget, der Komplexität der Anwendung, dem Scope der Testabdeckung, der Teamgröße und der entsprechenden Testphase ab. So findet ein Mock/Stub seine Berechtigung in kleinen, lokalen Entwicklungsprojekten. Dahingegen ist der langfristige Einsatz eines Virtualisierungstools in einem Unternehmen mit mehreren verschiedenen ineinandergreifenden Entwicklungsprojekten die günstigere Variante.

Es bleibt spannend, wie sich der Markt entwickeln wird und ob alle vorgestellten Hersteller ihre selbst auferlegte Roadmap einhalten. Ebenso interessant wird auch die Entwicklung dieser Tools im Bereich Open Source sein. Hier bleibt abzuwarten, ob sich ein Tool – ähnlich JMeter im Lasttestbereich – von den Mitbewerbern absetzen kann. Derzeit gibt es hier nichts Vergleichbares, so dass eine seriöse Vorhersage nicht zu treffen ist.

Die Integration in die Testlandschaft ist bei den kommerziellen Tools bereits gegeben. So ist es möglich, die erstellten Services mittels eines Hudson oder Jenkins-Servers in die Build-Pipeline zu integrieren und bereitzustellen. Das ermöglicht die automatische Bereitstellung von Testumgebungen, bestehend aus echten und simulierten Systemen. Diese Testumgebungen stellen die Basis für frühzeitige automatisierte End-to-End Tests.

Ein letzter Punkt: Wir sind gespannt, welchen Einfluss diese Tools auf den Testprozess insgesamt haben werden. Im besten Fall ähnelt der Einfluss dem des DevOps-Ansatzes und damit einhergehend der Etablierung des Einsatzes von Sourcecode- und Buildmanagement-Werkzeugen zur kontinuierlichen Integration (CI).