Christian Groneberg
(christian.groneberg@cgi.com)
ist Berater für Softwaretest und IT-Qualitätssicherung bei der CGI Deutschland Ltd. & Co. KG. Zusätzlich verantwortet er als Produktmanager für das Serviceangebot „SOA Testing" die strategische Weiterentwicklung des Themas innerhalb des Unternehmens.

Service-Nutzung auf eigene Gefahr: Risikobasiertes Testen unter SOA

Das Testen ist heutzutage eine etablierte Maßnahme zur Qualitätssicherung von Software. Häufig finden risikobasierte Teststrategien Anwendung, da sie eine zweckmäßige Priorisierung der Testressourcen versprechen. Im Kontext einer Software, die als SOA realisiert wird, ergeben sich bei diesem Ansatz jedoch spezielle Hürden für die Testorganisation, die zu Fehleinschätzungen von Produktrisiken führen können. Der Artikel benennt konkrete Gefahren und erläutert, wie die Testabteilung mit Hilfe von SOA-eigenen Konzepten strukturiert Information gewinnen und zur Einschätzung von Risiken verwenden kann. Darüber hinaus zeigt er Möglichkeiten auf, wie die gewonnenen Risikoinformationen bei der Wahl der Testmethoden berücksichtigt werden können.

Serviceorientierte Architekturen (SOAs) als Entwurfsparadigma für verteilte Systeme haben in der Informationstechnologie eine bewegte Historie. Anfänglich als Allheilmittel zur *Enterprise Application Integration (EAI)* beworben, wurde SOA schließlich Anfang 2009 für tot erklärt. Ist SOA damit vom Markt verschwunden? Mitnichten! Zwar hat das marketingwirksame Akronym an Bedeutung verloren, die geistigen Nachfolger wie Mashups, Cloud-Computing und Mobile Computing halten den Servicegedanken jedoch am Leben (vgl. [Man09]).

SOA zeichnet sich durch die funktionale Kapselung und Wiederverwendung betrieblicher IT-Leistungen in Form von Diensten (*Services*) aus. Im Sinne einer effizienten Ressourcennutzung strebt SOA an, einen einmal implementierten Service möglichst breitflächig anderen Funktionsträgern anzubieten und durch das Zusammenspiel mehrerer Services (Service-Komposition) fachlich höherwertige Leistungen zu erbringen (vgl. [Til07]). Auf diese Weise werden betriebliche Prozesse IT-seitig abgebildet, die über die Grenzen der eigenen Organisationseinheit hinweg reichen und dabei sogar verschiedene Unternehmen umspannen können, etwa die von Lieferanten und Kunden.

So entstehen informationstechnische Lieferketten, deren Lieferfähigkeit von der Qualität jeder einzelnen Leistungsstufe im Sinne der zugrunde liegenden Services abhängt (siehe Abbildung 1).

No Risk, no test

Das gängigste Mittel zur Prüfung der Qualität einer Software ist nach wie vor das Testen. SOA-basierte Anwendungen stellen diesbezüglich keine Ausnahme dar. In der Praxis haben risikobasierte Teststrategien (*Risk-based Testing*) weite Verbreitung gefunden. Bei diesem Vorgehen wird das Softwareprodukt funktional zerlegt und jedes Teilprodukt wird einer Risikobewertung mit Blick auf das Kerngeschäft unterzogen. Das Geschäftsrisiko wird aus dem Produkt von Fehler-Eintrittswahrscheinlichkeit und zu erwartender Fehlerwirkung für jedes Teilprodukt ermittelt. Beide Größen werden dabei anhand geeigneter Kriterien bewertet.

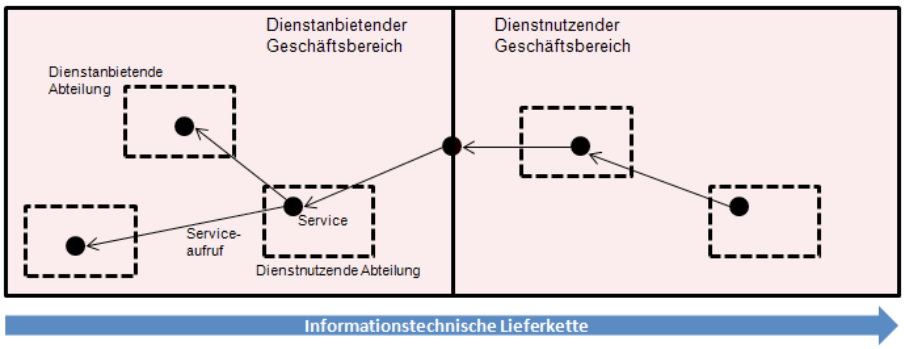

Abb. 1: Informationstechnische Lieferketten in einer SOA.

Fachfunktion				RISK ASSESSMENT: FEHLERWIRKUNG					FEHLEREINTRITTSW.				RISIKO
Number			Name	Prozess-typ	Rechtl. Relevanz	Benutz. Häufigk.	Kunden-gruppe		Änderung	SW-Reifegrad	Komplexität		
1			**Anwendungsfall**										
1	1		Registrierung	2) Daten ändern	2) Falsche Information	3) selten	2) einige	W2	3) keine Änderung	1) Individual-software	1) hoch	E2	B
1	2		Kundendaten löschen	2) Daten ändern	1) Rechtl. relevant	3) selten	3) wenige	W2	1) neu eingefürt	2) Customized	3) gering	E3	C
1	3		Passwort vergessen	2) Daten ändern	3) nicht relevant	2) häufig	3) wenige	W3	3) keine Änderung	1) Individual-software	2) mittel	E2	C
1	4		Produktsuche	3) Daten anzeigen	3) nicht relevant	1) regelmäßig	1) alle	W2	2) Änderung	1) Individual-software	2) mittel	E2	B
1	5		Bestellvorgang	1) Daten kalkulieren	1) Rechtl. relevant	1) regelmäßig	2) einige	W1	2) Änderung	1) Individual-software	1) hoch	E1	A
1	6		Kontaktanfrage	2) Daten ändern	3) nicht relevant	2) häufig	1) alle	W2	1) neu eingefürt	3) Standard-software	3) gering	E2	B

Abb. 2: Beispielhaftes Ergebnis eines klassischen Risk Assessments.

Beispielhafte Kriterien zur Bewertung der Fehler-Eintrittswahrscheinlichkeit sind etwa der Softwarereifegrad oder die funktionale Komplexität. Mögliche Kriterien zur Erfassung der Fehlerwirkung sind zum Beispiel die Anzahl potenziell betroffener Kunden oder die Gerichtsbarkeit eines möglichen Schadens.

Die einzelnen Bewertungskriterien werden schließlich durch Skalierung und Gewichtung quantifiziert, sodass als Ergebnis eines solchen *Risk Assessments* für jeden Softwarebaustein eine Risikoklasse – A (hoch), B (mittel) und C (niedrig) – ermittelt wird. Abbildung 2 zeigt ein Beispiel hierfür. Die Risikoklassen sind wiederum Grundlage für die Priorisierung der Testressourcen in der Testplanung.

Hürden des „Risk-based SOA-Testing"

Um eine Risikobewertung sinnvoll durchführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. So müssen die Bewertungskriterien über alle zu bewertenden Produktbestandteile hinweg sinnvoll anwendbar sein, was wiederum voraussetzt, dass das zu bewertende Softwareprodukt zum Zeitpunkt der Bewertung vollständig bekannt ist. Darüber hinaus sollte die Bewertung auf einer möglichst einheitlichen Granularitätsebene durchgeführt werden, zum Beispiel auf den Dialogmasken einer grafischen Benutzungsoberfläche oder auf der Ebene von Web-Pages einer Website. Dies gewährleistet die Vergleichbarkeit der verschiedenen Bestandteile des Testobjekts untereinander.

Grundsätzlich ist es für die Anwendung eines risikobasierten Testansatzes nicht relevant, nach welchen Designprinzipien die Software entworfen wurde. Bei SOA kommen jedoch einige Eigenarten des Architekturstils zum Tragen, die einer Risikobewertung nach bekanntem Muster im Wege stehen:

- Die Risikobewertung einer SOA-basierten Software kann aus Testsicht nicht auf der einheitlichen funktionalen Granularitätsebene einer Benutzungsschnittstelle durchgeführt werden, da die Funktionalität nicht zwangsläufig in eine solche integriert wird. Schließlich ist es durchaus möglich, dass Services für eine rein maschinelle Nutzung vorgesehen sind (Maschine-zu-Maschine-Kommunikation).
- Aufgrund des Wiederverwendungsgedankens ist eine SOA in ihrer fachlich-funktionalen Gesamtheit nicht klar eingrenzbar und die Grenzen zwischen Applikationen verschwimmen. Das ist insbesondere dann der Fall, wenn Services für die Verwendung durch mehrere Service-Konsumenten gedacht sind, diese (zukünftigen) Konsumenten aber zum Zeitpunkt der Testplanung noch nicht absehbar sind.
- Die Service-Konfiguration einer SOA kann sich zur Laufzeit dynamisch ändern. Als Folge sind die tatsächlichen Service-Aufrufe zu einem geplanten Zeitpunkt in der Test- und auch in der Produktivumgebung nicht bekannt und aus Sicht des *Risk Assessments* (Risikobewertung) ist das zu bewertende Softwareprodukt somit funktional nicht stabil. Diese Volatilität muss bei der Risikoeinschätzung der zu testenden Software berücksichtigt werden.
- Die organisatorisch und/oder räumlich getrennten Verantwortlichkeiten bei der Anforderung, der Entwicklung und dem Betrieb von Services erhöhen die Wahrscheinlichkeit, dass aufgrund von Abstimmungsproblemen und Informationsasymmetrien Produktrisiken unterschiedlich wahrgenommen oder bewertet werden.
- Die Entwicklung der eigenen SOA kann auf Services aufsetzen, die außerhalb der eigenen Organisationseinheit (z. B. des eigenen Projekts, Abteilung, Geschäftsbereichs) gehostet und damit im Sinne einer Blackbox-Sicht auf diese Services für die Testverantwortlichen schwer einschätzbar sind.*

Die genannten Herausforderungen konnte ich während der Zusammenarbeit mit der Testorganisation eines Logistikunternehmens wiederholt beobachten, woraus der im Folgenden vorgestellte Lösungsansatz entstand.

Ein fester Anker: der Service-Vertrag

Der erste Schritt zur Lösung des Problems, eine risikobasierte Teststrategie trotz der

**Services können natürlich auch unternehmensübergreifend zum Einsatz kommen. Der vorliegende Beitrag legt den Schwerpunkt bewusst auf eine Inhouse-SOA und gibt im Fazit einen Ausblick auf weiterführende Aspekte wie externe Services.*

Abb. 3: Die vier Vertragsarten eines Service.

oben genannten Hürden im Kontext von SOA zu implementieren, findet sich in dem Architekturstil selbst: SOA gibt die Struktur der Anwendungsfunktionalität bereits sehr klar vor, nämlich durch die Definition einzelner Services, die jeweils eine wohldefinierte Teilfunktionalität des Gesamtsystems darstellen. Obwohl die Granularität der Services stark variieren kann – von einfachen Basis-Services zur Datentransformation, bis hin zu komplexen fachlichen Prozess-Services –, so gibt es doch eine gemeinsame Konstante: den Service-Vertrag. Er bietet sich daher geradezu als Bewertungsgrundlage an, wenn es darum geht, über das gesamte Service-Portfolio einer SOA hinweg einheitlich und flächendeckend Risikokriterien anzuwenden.

Der Service-Vertrag spezifiziert alle funktionalen und nicht-funktionalen Eigenschaften eines Service. Hierzu zählt insbesondere die Serviceschnittstelle, das heißt die von dem Service angebotenen Operationen, Ein- und Ausgabeparameter. In Anlehnung an das *Design-by-Contract*-Paradigma lassen sich bei einem Service-Vertrag vier Vertragsarten unterscheiden (siehe auch Abbildung 3):

- Der *Basic Contract* beschreibt die Syntax der Service-Operationen und ihrer Parameter.
- Der *Behavioural Contract* gibt das funktionale Verhalten des Service vor.
- Der *Synchronization Contract* macht Vorgaben zur Aufrufreihenfolge von Service-Operationen oder zur Reihenfolge ihres Nachrichtenaustauschs.
- Der *QoS Contract (QoS) Contract* definiert schließlich nicht-funktionale Eigenschaften des Service (vgl. [Mey92]).

Der Blick zurück: Fehler-Eintrittswahrscheinlichkeit

Der Service-Vertrag wird datentechnisch in der sogenannten *Service-Registry* erfasst. Die in ihr enthaltenen Informationen sind zunächst einmal aus rein technischen Gründen notwendig, um den Service zur Laufzeit überhaupt zur Ausführung bringen zu können. In einem risikobasierten Testvorgehen sind diese Informationen aber weitergehend von Interesse. Mit ihrer Hilfe ist es möglich, Kriterien abzuleiten, die Rückschlüsse zulassen auf die technische und fachliche Komplexität der Service-Implementierung und damit indirekt auf die Fehlerträchtigkeit eines entsprechenden Service-Aufrufs. Diese Kriterien sind zudem über das gesamte Service-Portfolio einer SOA vergleichbar, da die oben beschriebenen Vertragsarten für jeden Service existieren. Folgende Kriterien sind für die verschiedenen Vertragsarten darstellbar:

- *Basic Contract:* Anzahl an Service-Operationen und -Parameter.
- *Behavioural Contract:* Abhängigkeiten zwischen den Eingabeparametern einer Service-Operation, Komplexität der Parametertypen von Ein- und Ausgabeparameter.
- *Synchronization Contract:* Nachrichtenaustausch-Muster, Abhängigkeit der Service-Operationen untereinander, Abhängigkeiten zu anderen Services.
- *QoS Contract:* Zugesicherte Verfügbarkeit gemäß *Service Level Agreement (SLA)*.

Die Service-Registry ist also eine wesentliche Quelle, um an Bewertungsgrundlagen für einen risikobasierten Testansatz für SOA-basierte Software zu gelangen. Eine andere Quelle ist das Service-Repository. Im Unterschied zur Registry sind die im Repository gesammelten Informationen nicht für rein technische Zwecke gedacht, sondern zur Verwendung durch die Stakeholder einer SOA. Inhalte des Repositorys können beispielsweise organisatorische Verantwortlichkeiten in Bezug auf einen Service sein, eine fachliche Beschreibung seines Funktionsumfangs oder Informationen zu dessen Lebenszyklus.

Während die Inhalte der Service-Registry durch technologische Standards weitgehend vereinheitlich sind, sind es die im Service-Repository gehaltenen Informationen nicht. Sie können daher in Umfang und Form stark variieren. Sinnvolle Informationen zum Zwecke der Einschätzung der Fehleranfälligkeit sind etwa folgende:

- *Service-Abhängigkeiten:* Eine Abhängigkeitsanalyse zwischen Services liefert einen Blick „nach hinten“ entlang der informationstechnischen Lieferkette und zeigt, auf welche anderen Services ein Service aufbaut. Merke: eine (Liefer-)Kette ist nur so stark wie ihr schwächstes Glied.
- *Service-Klasse:* Eine Service-Klasse gibt Hinweise auf die fachliche Komplexität des Service, beispielsweise ob es sich um einen atomaren oder orchestrierten Service handelt.
- *Versionsnummer:* Die Versionsnummer lässt Rückschlüsse auf den Reifegrad der Service-Implementierung zu.
- *Anbieter:* Der Anbieter erlaubt die Einschätzung der Fehleranfälligkeit anhand von Erfahrungen oder historischen Daten zu früheren Tests von Services desselben Anbieters.

Der Blick nach vorn: Fehlerwirkung

Die Bewertung der Fehler-Eintrittswahrscheinlichkeit ist jedoch nur der erste Schritt bei der Durchführung einer Risikoanalyse. Der zweite Schritt ist die Beurteilung der Auswirkungen eines potenziell eintretenden Fehlers. Hierfür ist entscheidend, welchen fachlichen Funktionen der zu bewertende Service dient und welche anderen Systembestandteile auf ihm aufbauen. Anders ausgedrückt, muss der Blick nicht wie bei der Beurteilung der Fehleranfälligkeit nach hinten, sondern stattdessen nach vorn entlang der Service-Lieferkette gerichtet werden.

Eine erste Anlaufstelle, um zu einer Einschätzung der Fehlerwirkung eines Service zu gelangen, ist erneut der in der Service-Registry enthaltene Service-Vertrag. Eine Analyse der vier Vertragsarten kann konkrete Hinweise auf den Nutzungskontext des Service geben. Beispiele dafür könnten sein:

- *Basic Contract:* Fachlich getriebene Namenskonventionen für Service-Operationen oder Service-Parameter.
- *Behavioural Contract*: Ausprä-

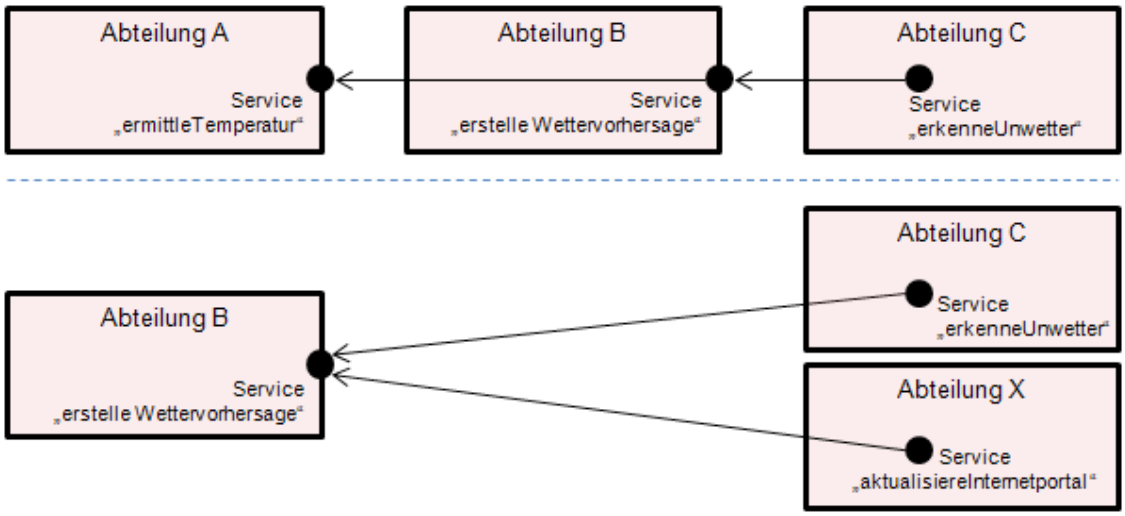

Abb. 4: Wiederverwendung (oben) und Service-Mehrfachverwendung (unten).

gung der vom Service angebotenen Funktionalität, etwa ob der Service fachliche Aufgaben übernimmt oder technische (z. B. Datentransformation). Insbesondere die Datentypen der Ein- und Ausgabeparameter des Service können diesbezüglich wichtige Informationen preisgeben.

- *Synchronization Contract:* Abhängigkeiten zwischen Services (erkennbar anhand semantischer Schnittstellendaten oder durch Analyse der Ein-/Ausgabeparameter verschiedener Services) sowie das verwendete Nachrichtenaustausch-Muster lassen Rückschlüsse zu, welche Systembestandteile durch eine Fehlerwirkung beeinträchtigt werden und in welchem Ausmaß (etwa Verzögerung oder Deadlocks bei synchronem Nachrichtenaustausch).
- *QoS Contract:* Vereinbarte Service-Gebühren oder verwendete Security-Mechanismen.

Einer besonderen Bedeutung in Bezug auf die Auswirkung eines Fehlers unter SOA kommt den Konzepten *Mehrfachverwendung* und *Wiederverwendung* von Services zu.

Unter Mehrfachverwendung versteht man den direkten Aufruf des Service durch unterschiedliche Service-Konsumenten. Die Wiederverwendung eines Service bedeutet hingegen, dass dieser indirekt durch eine Softwarekomponente in Anspruch genommen wird. Abbildung 4 verdeutlicht beide Prinzipien. Was haben diese aber nun mit risikobasiertem Testen zu tun?

Um zu einer Risikoeinschätzung für einen Service zu gelangen, muss der fachliche Kontext untersucht werden, in den der Service eingebunden ist und der daher im Fall eines Fehlers in der Service-Implementierung beeinträchtigt wird.

Als Beispiel sei der Service eines meteorologischen Instituts genannt, der von Abteilung A bereitgestellt wird und der die Messwerte mehrerer geografisch verteilter Temperatursensoren zurückliefert. Dieser Service könnte von dem IT-System der Abteilung B in Anspruch genommen werden, welches die Daten zur Erstellung einer Wetterprognose weiterverwendet. Ein Fehler im Temperatur-Service von Abteilung A könnte so zu einer verringerten Aussagekraft der Wettervorhersage durch Abteilung B führen. Abteilung B könnte aber wiederum selbst einen Service anbieten, der umfangreiche Wetterinformationen an Abteilung C bereitstellt, die damit beauftragt ist, Katastrophen und Unwettergefahren frühzeitig zu erkennen und darauf entsprechend zu reagieren. Der ursprüngliche Temperatur-Service aus Abteilung A wird damit indirekt Teil der IT von Abteilung C. In diesem neuen fachlichen Kontext der Katastrophen- und Unwettererkennung könnte ein Fehler im Temperatur-Service wesentlich gravierender Folgen haben als im Kontext der Wettervorhersage, zum Beispiel durch eine ausgebliebene Hochwasser-Meldung.

Mehrfach- und Wiederverwendung liegen in der Natur einer SOA. Für ein risikobasiertes Testvorgehen im Umfeld einer solchen Architektur ist es daher erforderlich, diese Aspekte zu würdigen, um zu einer Einschätzung der größtmöglichen Fehlerwirkung durch einen Service zu kommen. Diese wird bestimmt durch den tatsächlich vorgesehenen oder in Zukunft denkbaren fachlichen Kontext der Service-Nutzung. Erneut kann ein Blick in das Service-Repository weiterhelfen. Hier können detaillierte Information über die geplanten Szenarien zur Service-Nutzung enthalten sein, aus denen wiederum Kriterien für Risikoaussagen extrahiert werden können. Beispiele dafür können sein:

- Anzahl der tatsächlichen oder planmäßig vorgesehenen Service-Konsumenten.
- Anzahl der potenziellen Service-Konsumenten im Zeitverlauf des Service-Lebenszyklus.
- Klassifikation der Service-Konsumenten (z. B. Applikationen mit Außenwirkung, wie etwa ein Kundenportal oder ausschließlich unternehmensintern genutzte Software).
- Zugesicherte Service-Qualität (je niedriger die QoS des Service, umso geringer ist der zu erwartende Nutzungsradius).
- Kostenmodell (eine kostenlose Service-Nutzung lädt zur breitflächigen Anwendung des Service ein).
- Geplante Nutzungsdauer im Service-Lebenszyklus (z. B. Unterscheidung, ob ein Service neu eingeführt wird oder ob er bereits betrieblich ausgeplant wurde).

Der Kluge häuft Wissen, der Weise sortiert es

Die mit Hilfe der Service-Registry sowie des Service-Repositorys identifizierten Kriterien, anhand derer jeder Service hinsichtlich der Fehler-Eintrittswahrscheinlichkeit sowie der potenziellen Fehlerwirkung bewertet werden kann, können nun in ein Risk Assessment nach klassischem Muster einfließen. Abbildung 5 zeigt exemplarisch, wie das Ergebnis eines Risk Assessments für eine SOA aussehen könnte.

RISK ASS

	FEHLERWIRKUNG				
	Service Repository				
Servicebezeichnung	**Servicenutzer-kategorie**	**Anzahl Servicenutzer**	**Kostenmodell**	**Tendenz Servicenutzung**	
erstelleKunde	1) extern	1) Masse	2) kostenlos	3) abnehmend	W2
leseKunde	1) extern	1) Masse	2) kostenlos	2) konstant	W2
schreibeKunde	3) interne Abteilung	2) Gruppe	3) nur intern	2) konstant	W3
erstelleVertrag	1) extern	1) Masse	1) kommerziell	1) zunehmend	W1
schreibeVertrag	3) interne Abteilung	3) einzeln	3) nur intern	2) konstant	W3
leseVertrag	2) Unternehmensweit	2) Gruppe	2) kostenlos	2) konstant	W2

ESSMENT

FEHLEREINTRITTSWAHRSCHEINLICHKEIT					RISIKO
Service Registry					
Basic Contract	**Behavioural Contract**	**Synchronization Contract**	**QoS Contract**		
1) Operationen >10	1) Abh. Parameter > 2	2) async. ODER zustandsbeh.	3) Verf. > 99%	E2	B
3) Operationen <5	3) keine abh. Param.	3) sync. UND zustandslos	1) Verf. < 97%	E3	C
3) Operationen <5	3) keine abh. Param.	2) async. ODER zustandsbeh.	3) Verf. > 99%	E3	C
1) Operationen >10	1) Abh. Parameter > 2	1) async. UND zustandsbeh.	3) Verf. > 99%	E1	A
2) Operationen 5-10	3) keine abh. Param.	2) async. ODER zustandsbeh.	3) Verf. > 99%	E3	C
2) Operationen 5-10	2) Abh. Parameter = 2	3) sync. UND zustandslos	2) Verf. 97%-99%	E2	B

Abb. 5: Beispiel eines SOA Risk Assessments.

Wie kann die Testabteilung nun die Ergebnisse eines SOA Risk Assessments nutzen und wie sehen die Optionen für unterschiedliche Testansätze in Abhängigkeit von der Risikoinhärenz eines Service konkret aus? Beim Test eines einzelnen, isolierten Service können als Testdeckungskriterien die oben erläuterten Vertragsarten herangezogen werden. Die einzelnen Arten des Service-Vertrags bauen aufeinander auf und erlauben so, modulare Ausbaustufen für den Test eines Service gegen seine Schnittstellenspezifikation durchzuführen. Beispielsweise könnte für weniger risikobehaftete Services der Test auf die Verifikation seiner fachlichen Kernfunktionen beschränkt werden, das heißt Abdeckung des Basic Contract sowie des Behavioural Contract.

Für risikoreichere Services könnten hingegen auch aufwendigere Tests des Synchronization Contract vorgesehen werden. Ein sinnvolles Vorgehen hierzu ist das sogenannte modellbasierte Testen. Hierbei werden mögliche Interaktionsszenarien des Service im Zusammenspiel mit anderen Services (Nachrichtenaustausch-Muster, Aufrufsequenzen der Service-Operationen, Verhalten bei parallelen Service-Aufrufen) anhand geeigneter Modelle nachgebildet und daraus werden wiederum Testfälle abgeleitet – also genau das, was der Synchronization Contract definiert. Je nach verfügbarer Dokumentationsbasis der Service-Spezifikation kommen Zustands-, Interaktions-, Aktivitäts-, Kollaborations- oder Sequenzdiagramme für die Modellierung der Testfälle in Betracht (detaillierte Erläuterungen hierzu inklusive einer Diskussion sinnvoller Deckungskriterien finden sich in [Gro05]).

Vervollständigt wird die Testabdeckung durch eine Verifikation des QoS Contract, was etwa für Services der höchsten Risikoklasse anzuraten ist. Ein Beispiel hierfür wäre die Überprüfung von Szenarien zur Ausfallsicherheit, die in die Testplanung für hochriskante Services mit einfließen.

Noch interessanter als für den Test eines einzelnen Service sind die Ergebnisse des SOA Risk Assessments für die Planung integrativer Tests zwischen mehreren Services. Erst die Gegenüberstellung des Geschäftsrisikos verschiedener Services macht sichtbar, welche informationstechnischen Lieferketten besonders risikobehaftet sind, weil sie eine hohe Zahl risikobehafteter Services integrieren oder in kritische Datensenken (Geschäftsprozesse, Fachfunktionen) münden. Um diese Risikoverteilung innerhalb des Service-Portfolios zu analysieren, können – idealerweise durch automatisiertes Parsen des Service-Repositorys – die Service-Abhängigkeiten ermittelt und grafisch in Form eines Abhängigkeitsgraphen aufbereitet werden. Die Knoten des Graphen repräsentieren die einzelnen Services. Sie können farblich markiert werden, um das Geschäftsrisiko gemäß Risk Assessment zum Ausdruck zu bringen. Abbildung 6 zeigt ein einfaches Beispiel.

In einem solchen Graph verlaufen Service-Lieferketten vertikal und Geschäftsprozesse meist horizontal über unterschiedliche Applikationen hinweg. An den Stellen inner-

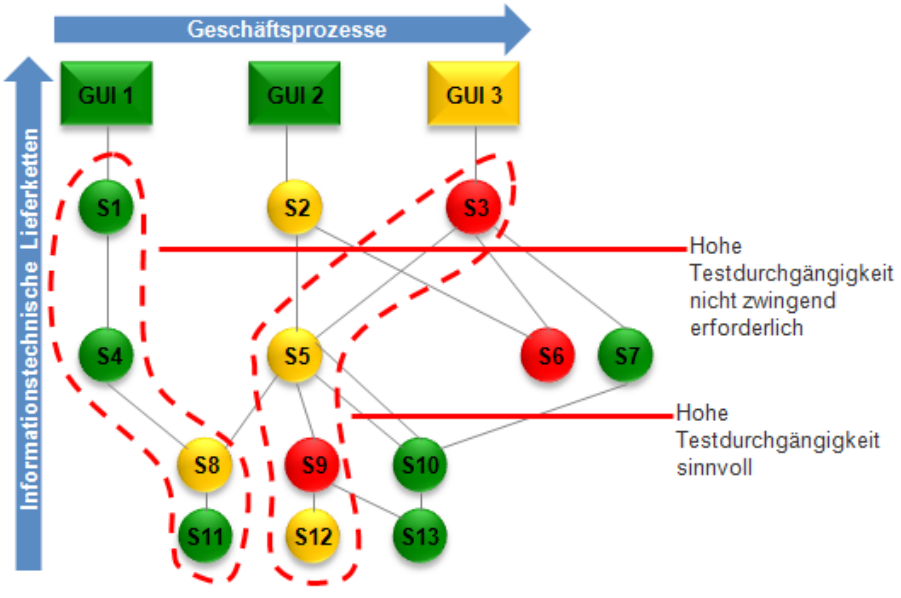

Abb. 6: Visualisierung der Ergebnisse eines SOA Risk Assessments mit Services hohen (rot), mittleren (gelb) und geringen (grün) Geschäftsrisikos.

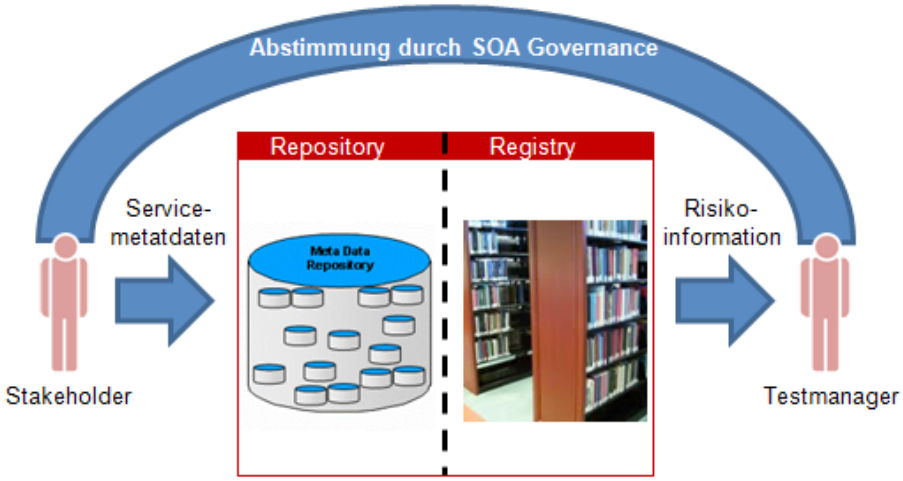

Abb. 7: Einbindung der Testorganisation in die SOA-Governance.

halb des Graphen, an denen sich vertikal verknüpfte Knoten mit hohem Geschäftsrisiko häufen, befinden sich kritische Lieferketten. Verfolgt man diese Lieferketten weiter nach oben in der Darstellung, gelangt man zu den Konsumenten dieser Ketten: den betroffenen Geschäftsprozessen.

Im Rahmen eines komplexen Integrationsvorhabens sollten die kritischen Ketten mit erhöhter Priorität in der Testplanung bedacht werden, um die abhängigen Geschäftsprozesse frühzeitig mit validen Daten versorgen zu können und damit deren Testbarkeit sicherzustellen.

Die Priorisierung der Tests kann anhand der Risikoverteilung erfolgen. So ist ersichtlich, wo es sich lohnt, aufwendige Testansätze vorzubereiten. Integrative Tests können in den beiden Dimensionen *Testdurchgängigkeit* und *Testtiefe* variiert werden.

Mit Testdurchgängigkeit ist die Anzahl miteinander im Verbund getesteter Services gemeint, mithin die Länge der informationstechnischen Lieferkette, die in der Testumgebung ohne den Einsatz von Mocks abgebildet werden kann. Ein möglichst durchgängiger Test erhöht die Chance, technische oder semantische Fehler in der Verarbeitung von Prozessdaten zu entdecken. Grundsätzlich ist anzuraten, mit zunehmender Anzahl risikobehafteter Services in einer Lieferkette auch die Testdurchgängigkeit zu erhöhen.

Unter Testtiefe ist die Art und Anzahl der Prüfpunkte zu verstehen. Grob kann zwischen integrativen Blackbox-Tests, Greybox-Tests und Whitebox-Tests unterschieden werden, wobei Blackbox-Tests die geringste Anzahl an Prüfpunkten während der Testausführung beinhalten und Whitebox-Tests die höchste Anzahl. So kann im Test der erfolgreiche Aufruf einer Service-Operation anhand ihrer Rückgabewerte bestätigt werden (Blackbox), durch Analyse der resultierenden Änderungen in der Datenhaltungsschicht unterhalb des Service (Greybox) oder durch Checkpunkte innerhalb der Service-Implementierung (Whitebox).

Mit zunehmender Testdurchgängigkeit oder Testtiefe steigt auch der Testaufwand. Durch die Berücksichtigung der Ergebnisse im SOA Risk Assessment kann aber ein Kompromiss gefunden werden zwischen Testaufwand und Restrisiko. Tabelle 1 beschreibt mögliche Testoptionen für Service-Ketten in Abhängigkeit von ihren Risikoklassen.

Und bist du nicht willig ...

Es ist deutlich geworden, dass die Durchführbarkeit und die Aussagekraft eines SOA Risk Assessments – und damit auch das Qualitätsniveau der Software – von den in Service-Registry und -Repository enthaltenen Informationen abhängen. Bereits frühere empirische Untersuchungen weisen auf das Qualitätspotenzial hin, das von den Dokumentationsmöglichkeiten unter SOA ausgeht (vgl. [Sch07]). Aus der Organisationssicht ist daher sicherzustellen, dass die für einen risikobasierten Testansatz relevanten Informationen regelmäßig gepflegt werden.

Für die Service-Registry kann diese Forderung grundsätzlich als erfüllt gelten, da sie ein zwingender Bestandteil der SOA-Infrastruktur ist. Diese Aussage gilt aber nur mit Einschränkungen. Zum Beispiel kann die Service-Schnittstellenspezifikation syntaktische Daten enthalten, aber keine semantischen. Der primäre Zweck der Service-Registry, nämlich technisch die Ausführbarkeit des Service sicherzustellen, wird bereits durch eine rein syntaktische Schnittstellenausprägung erfüllt. Für die Beurteilung der Risikoinhärenz eines Service sind semantische Daten aber von höherem Wert. Ebenso kann der Service-Vertrag SLAs definieren – er muss es aber nicht. Für das Service-Repository gilt ähnliches. Zwar bieten aktuelle SOA-Produktsuiten von Hause aus die Möglichkeit, das Repository umfangreich zu pflegen und zu verwalten. Das bedeutet aber nicht, dass diese Aktivitäten prozessual in die Organisationsstrukturen eingebunden sind und die Belange der Testabteilung dabei berücksichtigt werden.

Um in diesem Punkt Abhilfe zu schaffen, können verbindliche Vorgaben zu Architekturentscheidungen und Zusammenarbeitsmodellen formuliert werden. Diese können

Risikoklasse	Erläuterung	Testansatz: Service-Integration	Merkmale
	Services überwiegend hohen Risikos	Whitebox-Tests mit Datenfluss-Analyse; Vermeidung von Mocks *und* Verwendung repräsentativer Testdaten.	Produktionsnahe Tests und hohe Transparenz im Fehlerfall.
	Services überwiegend mittleren Risikos	Greybox-Tests mit Verifikation der Datenhaltungsschicht; Vermeidung von Mocks *oder* Verwendung repräsentativer Testdaten.	Hohe Testdurchgängigkeit bei vergleichsweise geringem Aufwand.
	Services überwiegend geringen Risikos	Blackbox-Tests mit Verifikation des Service-Verhaltens gegen den Service-Vertrag; Verwendung von Mocks und Fake-Testdaten.	Geringer Aufwand für Testdurchführung und -automatisierung.

Tabelle 1: Empfehlungen für Testansätze bei der Integration von Services.

etwa den Umfang des Service-Vertrags für alle Services vorschreiben und den Informationsbestand des Repositorys verbindlich regeln. Die geordnete Ablage von Service-relevanten Informationen im Repository ist an sich nichts Außergewöhnliches, sondern im Gegenteil eher Standard in jedem ernsthaften SOA-Vorhaben. Die gezielte Berücksichtigung eines spezifischen Informationsbedarfs der Testorganisation beim Design der Repository-Struktur scheint in der Praxis hingegen bislang nicht selbstverständlich zu sein, ist aber Voraussetzung für die Realisierung vieler Testaufgaben im Allgemeinen und die Umsetzung des hier vorgeschlagenen risikobasierten Testvorgehens für SOA im Speziellen.

Ein geeignetes Instrument zur Beherrschung dieses Missstands ist die SOA-Governance. Sie stellt eine Initiative dar, die durch organisatorische Maßnahmen abteilungs- und projektübergreifend Richtlinien durchsetzt, welche die Planung, Implementierung und den Betrieb einer Service-Landschaft betreffen. Ziel ist dabei, die für die Realisierung einer SOA relevanten Belange der Stakeholder aufzugreifen und adäquat zu adressieren. Die Testabteilung ist einer dieser Stakeholder. Geeignete Adressaten für ihren Informationsbedarf sind unter anderem Fachbereiche, Architekturboards und der Betrieb. Ziel ist es, den Informationsfluss über die Governance-Mechanismen so zu steuern, dass Service-Registry und -Repository inhaltlich den Testbelangen Rechnung tragen und ein risikobasiertes Testvorgehen unterstützen. Abbildung 7 verdeutlicht diese Zusammenhänge.

Fazit

Der Artikel hat aufgezeigt auf, wie trotz architekturbedingter Herausforderungen im Umfeld SOA-basierter Softwareentwicklung – Wieder- und Mehrfachverwendung, Abstraktion der Service-Implementierung, organisatorisch und räumlich verteilte Verantwortlichkeiten – eine risikobasierte Teststrategie angewendet werden kann. Grundlage des Lösungsvorschlags ist, die Risikobewertung durchgängig auf Ebene des Service-Vertrags anzuwenden und dabei geeignete Bewertungskriterien anhand von risikorelevanten Informationen aus Service-Registry und -Repository zu extrahieren. Das Ergebnis dieses Risk Assessments ist eine definierte Risikoklasse pro Service. Die Risikoverteilung innerhalb des Service-Portfolios wird anhand eines Abhängigkeitsgraphen visualisiert, um geschäftskritische Service-Abhängigkeiten aufzudecken.

Abschließend wurden testmethodische Vorschläge unterbreitet, um differenziert für Services unterschiedlicher Risikoinhärenz sinnvolle Testmaßnahmen zuordnen zu können. Voraussetzung für das vorgeschlagene Vorgehen zur Ermittlung von Geschäftsrisiken in einer SOA sind einheitlich und zielgerichtet gepflegte Service-Informationen sowohl in der Service-Registry als auch im Service-Repository. Die dort vorgehaltenen Informationen müssen explizite Belange der Testabteilung berücksichtigen – und zwar über den gesamten Service-Lebenszyklus hinweg.

Um sicherzustellen, dass die erforderlichen Informationen regelmäßig und in der erforderlichen Güte von allen Verantwortlichen beigestellt werden, muss die SOA-Governance als übergeordnetes Instrument die Anforderungen der Testabteilung prozessual mit aufnehmen und adressieren.

Der Blick in die Zukunft von SOAs verspricht eine wachsende Vernetzung und auch Intelligenz von interagierenden Systemen. Der mögliche Nutzungsradius von Services wächst damit dramatisch. Der Austausch von Daten auch über Unternehmensgrenzen hinweg muss zur Selbstverständlichkeit werden, soll das volle Potenzial solcher „Ultra Large Systems“ gehoben werden.

So aufregend und vielversprechend die Einbeziehung externer Services in die eigene Applikationslandschaft auch sein mag: Die Beschaffung von Informationen zur Qualität eines fremden Service und zu dessen Auswirkungen auf die Qualität einer informationstechnischen Lieferkette wird damit zunehmend schwieriger. In diesem Zusammenhang wird die Frage aufkommen, wie unternehmensübergreifend Informationen zur Einschätzung der Qualität und des Geschäftsrisikos von Services standardisiert und über eine globale SOA-Governance angeboten werden können. Noch spannender ist die Frage, wie anwendbar ein risikobasierter Testansatz in einer intelligenten SOA ist, die mit semantischen Technologien vollständig automatisiert die Nutzbarkeit von Services für die Realisierung einer fachlichen Funktion analysiert und sich regelmäßig dynamisch neu erfindet. Folgerichtig müsste die Risikoeinschätzung eines Service dann ebenfalls automatisiert zur Laufzeit durchgeführt werden.

Es bleibt abzuwarten, welche Rolle diese Fragestellungen beim technologischen Fortschritt im Umfeld von SOA spielen werden. ■

Literatur & Links

[Gro05] H.-G. Gross, Component-based Software Testing with UML, Springer Verlag, 2005

[Man09] A. T. Manes, SOA is Dead; Long Live Services, 2009, siehe http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html

[Mey92] B. Meyer, Applying „Design by Contract“, in: Computer, Oktober 1992, 25. Jg., Nr. 10

[Sch07] J. Schelp, R. Winter, Agilität und SOA, in: G. Starke, S. Tilkov (Hrsg.), SOA-Expertenwissen, dpunkt.verlag, 2007

[Til07] S. Tilkov, G. Starke, Einmaleins der service-orientierten Architekturen, in: G. Starke, S. Tilkov (Hrsg.), SOA-Expertenwissen, dpunkt.verlag, 2007