Software Engineering II (IB)
Serviceorientierte Architektur

Prof. Dr. Oliver Braun

Letzte Änderung: 16.05.2017 21:17

Webservices

- Ziel: flexible programmatische Zusammenarbeit zwischen Servern
- Bereitstellung eines Webservices
  - Veröffentlichung einer Schnittstelle
- Standardrepräsentation für eine Rechen- oder Informationsressource
- Servicedefinition von Lovelock et al., 1996:

  Eine Handlung oder Leistung, die ein Beteiligter einem anderen anbietet. Obwohl der Prozess an ein physisches Produkt gekoppelt sein kann, ist die Leistung im Wesentlichen immateriell und führt in der Regel nicht zum Besitz irgendeiner der Produktionsfaktoren.

Serviceorientierte Architekturen (SOA)

- Methode um verteiltes System zu entwickeln, deren Komponenten eigenständige Services sind
- XML-basierte Standardprotokolle wie SOAP und WSDL für Kommunikation
- Services unabhängig von Plattform und Programmiersprache
- Architektur
  - Serviceregistrierung
  - Serviceanforderer
  - Serviceanbieter
- oft als sehr schwergewichtig kritisiert
  - Alternative: REST (nächstes Kapitel)

Standards für Webservices

- Transport (HTTP, HTTPS, SMTP, ...)
- Nachrichtenaustausch (SOAP)
- Servicedefinition (UDDI, WSDL)
- Prozess (WS-BPEL)
- Unterstützung (WS-Security, WS-Addressing, ...)

Service Szenario

- Kfz-Informationssystem mit Informationen zu
  - Wetter
  - Verkehr
  - lokale Informationen
- verbunden mit dem Autoradio
- zusätzlich GPS-Empfänger um lokale Information zu identifizieren

Serviceorientiertes Software Engineering

- Service Engineering
  - Entwicklung unabhängiger und wiederverwendbarer Services
- Softwareentwicklung mit Services
  - Entwicklung einer Anwendung unter Verwendung von Services

Services als wiederverwendbare Komponente

Ein Service ist eine locker verbundene wiederverwendbare Softwarekomponente, die skalierbare Funktionen kapselt, welche verteilt sein können und per Programm erreichbar sind. Ein Webservice ist ein Service, der sich über übliche Internet- und XML-basierte Protokolle erreichen lässt.

- wesentlicher Unterschied zu Komponenten
  - ein Service hat keine `requires`-Schnittstelle
  - Services basieren auf einer nachrichtenorientierten Kommunikation

Web Service Description Language (WSDL)

- die Service-Schnittstelle ist als Service Description in WSDL beschrieben
- die WSDL-Spezifikation definiert
  - welche Operationen der Service bietet und das Format der Nachrichten
  - wie der Service genutzt werden kann
  - wo sich der Service befindet (üblicherweise als URI)

Service Engineering

- Entwicklung von Services zur Wiederverwendung in einer SOA
- der Service muss als wiederverwendbare Abstraktion entworfen werden, so dass er in verschiedenen Systemen nutzbar ist
- allgemein nützliche Funktionalität
- so dokumentiert, dass er gefunden und verstanden werden kann

Stadien des Service Engineerings

1. Ermittlung von Servicekandidaten
   - zu implementierenden Dienste identifizieren
   - Serviceanforderungen festlegen
2. Serviceentwurf
   - logische Serviceschnittstelle
   - WSDL-Schnittstelle
3. Serviceimplementierung und -bereitstellung
   - validierter und bereitgestellter Service

Identifikation eines Servicekandidaten

- Services müssen Geschäftsprozesse unterstützen
- Ermittlung erfasst
  - Verstehen und Analysieren der Geschäftsprozesse
- 3 grundlegende Arten nach Erl, 2004:
  - Hilfservices
    - z.B. Währungskonvertierungsservice
  - Geschäftsservices
    - z.B. Einschreibung von Studierenden für eine Vorlesung
  - Koordinations- oder Prozessservices
    - z.B. Bestellservice in einem Unternehmen, der Bestellungen mit Lieferanten, gelieferten Gütern und geleisteten Zahlungen entgegen nimmt

Auftrags- und entitätsorientierte Services

- mit Tätigkeit bzw. mit Objekten verknüpft
- Hilfs- und Geschäftsservices können entitätsorientiert sein
- Koordinationsservices immer auftragsorientiert

Fragen zur Ermittlung

- Ist der Service mit einer einzigen logischen Entität verknüpft, die in unterschiedlichen Geschäftsprozessen verwendet wird?
- Wird der Auftrag von verschiedenen Personen im Unternehmen ausgeführt?
- Ist der Service unabhängig?
- Muss der Service für seinen Betrieb einen Zustand verwalten? Ist eine Datenbank notwendig?
- Könnte der Service von Clients außerhalb des Unternehmens genutzt werden?
- Stellen unterschiedliche Benutzer unterschiedliche nichtfunktionale Anforderungen?

Beispiel

Ein großer Anbieter für Computerausrüstung hat einigen Kunden Sonderpreise für genehmigte Systemzusammenstellungen angeboten. Um die automatisierte Bestellung zu vereinfachen, möchte die Firma einen Katalogservice einsetzen, der den Kunden die Auswahl der benötigten Teile ermöglicht. Im Unterschied zu einem Kundenkatalog werden die Bestellungen jedoch nicht direkt über eine Katalogschnittstelle vorgenommen, sondern über das webbasierte Beschaffungssystem der einzelnen Firmen, die auf den Katalog als Webservice zugreifen. Die meisten Formen haben eigene Budgetierungs- und Genehmigungsverfahren für Bestellungen eingerichtet, die einzuhalten sind, wenn Bestellungen vorgenommen werden.

entitätsorientierter Service, der den Geschäftsbetrieb unterstützt

Funktionale Anforderungen an den Katalogservice

- für jede Benutzerfirma spezielle Version des Katalog mit den Systemzusammenstellungen die die Mitarbeiter der Firma bestellen können, sowie den vereinbarten Preisen
- Offline-Version zum Blättern zum Herunterladen
- Spezifikationen und Preise von bis zu 6 Katalogartikeln sollen verglichen werden können
- Blätter- und Suchfunktion
- Benutzer sollen Lieferdatum für eine bestimmte Anzahl von Katalogartikeln finden können
- Benutzer sollen "virtuelle Bestellungen" vornehmen können, bei denen die Artikel für 48 Stunden reserviert werden. Bis zum Ablauf muss eine reale Bestellung über ein Beschaffungssystem bestätigt sein.

Nichtfunktionale Anforderungen an den Katalogservice

- Zugriff auf Mitarbeiter berechtigter Firmen und Organisationen beschränkt
- die einem Kunden angebotenen Zusammenstellungen und Preise sollen vertraulich sein und von Mitarbeitern anderer Firmen nicht eingesehen werden können
- Katalog soll ohne Unterbrechungen von 7 bis 23 Uhr zur Verfügung stehen
- Katalogservice soll bis zu 10 Anfragen pro Sekunde bei Spitzenbelastung verarbeiten können

Entwerfen von Serviceschnittstellen

- Operationen und Nachrichten
- Anzahl der Nachrichten für einen Servicerequest soll minimal sein
- Zustandsinformationen müssen in den Nachrichten übermittelt werden, denn Services sind zustandslos

Stadien beim Entwerfen von Serviceschnittstellen

1. der logische Schnittstellenentwurf
2. der Nachrichtenentwurf
3. die WSDL-Entwicklung

Implementierung und Deployment

- Implementierung mit einer Standard Programmiersprache
- über Input- und Outputmessages testen
- auf einem Webserver veröffentlichen

Services für Altsysteme

- wichtige Anwendung für Services ist Zugriff auf Funktionalitäten in Altsystemen zu bieten, über so. “Hüllen” um diese Systeme
- Altsysteme bieten oft eine große Funktionalitätsvielfalt, die so einfach in neue Systeme integriert werden kann

Softwareentwicklung mit Services

- bestehende Services komponieren und konfigurieren um neue zusammengesetzte Services oder Anwendungen zu erzeugen
- die Basis für die Servicekomposition ist meist ein Workflow
  - Workflows sind logische Sequenzen von Aktivitäten, die zusammen einen Geschäftsprozess ergeben
  - z.B. Reisereservierungen mit koordinierten Flug-, Hotel- und Mietwagenbuchungen

Erstellung von Services durch Komposition

1. Workflow-Skizze formulieren
2. Services suchen
3. mögliche Services auswählen
4. Workflow verfeinern
5. Workflow-Programm schreiben
6. fertigen Service oder Anwendung testen

Entwurf und Implementierung des Workflows

- WS-BPEL ist ein XML-Standard für Spezifikationen von Workflows
  - sehr ausführlich und schlecht lesbar
- graphische Notationen, wie BPMN, sind besser lesbar
  - daraus kann dann WS-BPEL generiert werden

Testen von Services

- Fehler finden und validieren, dass das System die funktionalen und nichtfunktionalen Anforderungen erfüllt
- Services sind “black boxes”
- Probleme
  - der Anbieter kann seine Services ändern, so dass bisher erfolgreiche Tests fehlschlagen würden
  - dynamisches Binden von Services
  - nichtfunktionale Anforderungen hängen von der Last ab
  - wenn für Services pro Nutzung gezahlt werden muss, kann Testen sehr teuer werden
  - Services können wieder andere Services nutzen