Simulation von Web Services

Seminararbeit

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Universität Mannheim

im

Juli 2004

von

Swen Habenberger
aus Mannheim

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Webservice- und auch Webserverprovider wollen ihren Kunden einen möglichst hohen Standard an Qualität, Komfort und Zuverlässigkeit liefern. Um dies zu erreichen, müssen vor dem Einsatz in Produktivsystemen zunächst ausgiebige Tests und Simulationen durchgeführt werden. Bei solcherlei Tests kann man zwei prinzipielle Arten von Simulationen unterscheiden. Eine streng mathematische und eine computerunterstützte Simulation. Die mathematisch-formale Spezifikation ist jedoch im Geschäftsbereich zu kostenintensiv, und in vielen Bereichen auch nicht nötig, so dass sich hier die computergestützte Simulation mit Black- und Whiteboxtests durchgesetzt hat. Ein entscheidender Nachteil der formalen Simulationen ist die fehlende Dynamik des Systems, welches aber beim Einsatz von Webservices im Internet unabdingbar ist. Sofern Simulationen möglich sind, helfen Simulationen den Entwicklern Kosten zu sparen. Einen Atombombentest am Computer zu simulieren ist um ein vielfaches kostengünstiger und mit weniger Nebenwirkungen verbunden als ein real durchgeführter Test.
Im Bereich von Webservices kann man jedoch von allen möglichen bekannten Vor- und Nachbedingungen ausgehend, Simulationen mit einem hohen Grad an Exaktheit durchführen. Der Bereich der Webservice-Komposition ist ein neuer Forschungsbereich, der Web Service-Technologie und Prozess-Komposition kombiniert. Hierbei wird vor allem die Interoperabilität von Webservices getestet. Welche Kriterien für den erfolgreichen Einsatz von Webservices gelten müssen, wird in Kapitel 2 beleuchtet. In Kapitel 3 werden verschiedene Simulationsprogramme und ihre Stärken und Schwächen vorgestellt. Kapitel 4 widmet sich dann dem Simulationsprgramm Karmasim, welches an Hand einer Beispielanwendung vorgestellt wird. Im letzten Kapitel wird noch ein Ausblick auf die zukünftige Entwicklung gegeben.

Kapitel 2

Dienstgüte

Die vermehrte Anwendung von Webservices und der Zuwachs von Web-Serviceanbietern machen es dem Nutzer nicht leicht sich zwischen verschiedenen Anbietern zu entscheiden. Um die Auswahl zu erleichtern kann man die Dienstgüte, im englischen Quality of Service (QoS) genannt, eines Webservices betrachten und entscheiden, welcher Service am besten für den Einsatz geeignet ist. Gerade im E-Business Bereich sind garantierte Service-Level Vereinbarungen ein wichtiges Kriterium für den erfolgreichen Einsatz und die Integration von Webservices in Geschäftsprozessen. Standards wie UDDI, WSDL und SOAP können hierbei helfen, allerdings müssen auch andere Dinge dazu beitragen um einen korrekt funktionierenden Webservice zu entwerfen und zu betreiben.

2.1 Dienstgütekriterien

Die verschiedenen Aspekte der Dienstgüte kann man wie folgt grob in sieben Anforderungen aufgliedern[2]:

- Verfügbarkeit (Availability)
- Erreichbarkeit (Accessibility)
- Integrität (Integrity)
- Performance
- Zuverlässigkeit (Reliability)
- Regelwerk (Regulatory)
- Sicherheit

2.1.1 Verfügbarkeit

Das Kriterium der Verfügbarkeit repräsentiert die Wahrscheinlichkeit, dass ein Webservice verfügbar oder für die unmittelbare Verwendung bereit ist. Wichtigstes Meßkriterium ist die Time-to-repair (TTR). Die TTR ist diejenige Zeit, die benötigt wird um einen schadhaften Webservice nach einem Fehler zu reparieren. Entscheidungsträger für den Einsatz des Webservices müssen sich im Vorfeld über ein eventuelles Clustering und Management des Webservices Gedanken machen. Es kann hierbei um einfache Fragen, wie zum Beispiel wie viele Webservices auf einem einzelnen Webserver zur Verfügung stehen, handeln aber auch um komplexere Sachverhalte, beispielsweise bei einem Weltkonzern, welche Webservices an welchen Standorten angeboten werden sollen, insbesondere mit dem Hinblick auf eventuelle Fehlerkorrekturmöglichkeiten.

2.1.2 Erreichbarkeit

Erreichbarkeit bedeuetet im Zusammenhang mit Webservices der Prozentsatz der erfolgreichen Zugriff auf den Webservice. Es kann Situation geben, in den der Webservice zwar verfügbar, aber nicht erreichbar ist, beispielsweise kann durch fehlerhaftes Load Balancing ein Webserver überlastet sein, wobei andere Webserver theoretisch noch erreichbar sind. Durch Service-pooling kann erreicht werden, dass ankommende Anfragen in einem Zwischenspeicher gelagert werden, aus welchem sie nach und nach abgearbeitet werden. Hier kann es ebenfalls vorkommen, dass die Warteschlange zu klein gewählt wurde und eine Anfrage, trotz positiver Verfügbarkeit, nicht bearbeitet wird. Ein möglicher Angriff in diesem Zusammenhang ist die Denial-of-Service Attacke. Eine hohe Erreichbarkeit wird durch skalierbare Systeme erreicht. Ein gutes Beispiel hierfür ist der Webservice von Google, welcher mehr als 2000 Anfragen in der Sekunde verarbeiten kann.

2.1.3 Integrität

Das Kriterium der Integrität stellt die Korrektheit der Ergebnisse und Datenintegrität sicher. Bei Abbruch einer Transaktion müssen getätigte Veränderungen durch ein so genanntes roll-back rückgängig gemacht werden können.

2.1.4 Performance

Das Ziel einer guten Performance sind hoher Durchsatz und niedrige Latenzzeiten. Unter dem Durchsatz (Throughput) versteht man die beantworteten

Anfragen in einer bestimmten Zeiteinheit. Die Latenzzeit (Latency) ist die round-trip-time, welche benötigt wird um eine Antwort (response) auf eine Anfrage (request) zu bekommen.

2.1.5 Zuverlässigkeit

Die Zuverlässigkeit wird in Fehlern pro Monat oder Jahr gemessen. Hauptaugenmerk liegt auf der Anzahl der zugesicherten und in der richtigen Reihenfolge zugestellten Nachrichten. Bei einer Bank wäre es fatal, wenn ein Kunde Geld auf sein neu eingerichtetes Konto einzahlen wollte und anschliessend das Geld wieder abheben möchte, bei der Bank jedoch zuerst die Abhebung, welche veweigert wird, ankommt und erst im Anschluss daran die Geldeinzahlung.

2.1.6 Regelwerk

Der Aspekt des Regelwerks ist vor allem für Webserviceprovider wichtig, da die Kunden darauf vertrauen müssen, eindeutig definierte Schnittstellen und Zugriffsmöglichkeiten auf den Webservice zu erhalten. Eine strikte Einhaltung der Standards (wie SOAP, UDDI, WSDL), aber auch die Berücksichtigung der Gesetze und Servicelevelvereinbarungen sind entscheidend dafür, wie regelkonform ein Webservice ist. Die Angabe der verwendeten Version (z.B. SOAP Version 1.2), erleichtern den Entwicklern für Anwendungen des Webservices die zeitintensive Anwendungsentwicklung. Auch sollte zum Beispiel bei grenzüberschreitenden Handel die Import- und Exportbestimmungen, beispielsweise bei Medikamenten und Alkohol, der betroffenen Länder eingehalten werden.

2.1.7 Sicherheit

Sicherheit ist einer der wichtigsten Qualitätsaspekte der Dienstgütekriterien, da Webservices zumeist über das öffentlich Internet laufen. Die Möglichkeit zur Verschlüsselung von Nachrichten und die Authentifizierung tragen dazu bei, Webservices sicherer zu machen. Die Zugangskontrolle kann auch dazu dienen verschiedene Rollen zu definieren. So kann es bei einem Internetgroßbuchhändler die Rollen des Einzelkäufers, der Buchhandlung, des Verlages und des Autors geben, welche alle verschiedene Zugriffsmöglichkeiten auf den Buchhandel haben müssen. Auch spielt die Vertrauenswürdigkeit eine große Rolle. Wichtigste Standards sind hierbei XML Encryption oder Key Management Specification, sowie P3P (Platform for Privacy Preferences).

2.2 Dienstgütekriterien für Webservices

Von den oben aufgeführten Kriterien sind für die Unterstützung von Quality of Service in Web Services die folgenden vier Kriterien entscheidend[2]:

- Verfügbarkeit
- Erreichbarkeit
- Sicherheit
- Interoperabilität

Während Verfügbarkeit und Erreichbarkeit nur sekundär für Webserviceprogrammierer entscheidend sind, sind Sicherheit und Interoperabiltät Kriterien für den Entwickler von Webservices. Verfügbarkeit und Erreichbarkeit können durch den Einsatz von mehr Rechenleistung und besseren Kommunikationsprotokollen und Bandbreitenvergrößerung, also vorwiegend durch einen erhöhten Geldeinsatz, erhöht werden. Im Bereich der Interoperabiltät ist es entscheidend, dass die Punkte Regelwerk und Datenintegrität während der Anwendungsentwicklung eingehalten werden. Zum einen können künftige, komplexe Anwendungen aus mehreren einzelnen Webservices bestehen, zum anderen müssen bei fehlerhaften Schnittstellendefinitionen getätigte Transaktionen rückgängig gemacht werden können. Es soll durch die Interoperabiltät auch eine Weiterverwendung von Webservices ermöglicht werden.

Kapitel 3

Simulationsprogramme

Es gibt für die verschiedensten Einsatzgebiete und für die verschiedensten eingesetzten Sprachen seperate Programme. Einige seien hier stellvertretend vorgestellt.

- HaarnessIT for .Net[1]
  Die Entwickler von HernessIt haben schon früh erkannt, dass es keine spezielle Testsoftware für das .Net Framework gibt. Mit HaarnessIT kann man Tests für .Net und C++ entwickeln, sowie durch die Interop-Features von .Net können die COM-komponenten von VB6 oder VC++ getestet werden. Zum testen steht eine 45-Tage Testversion bereit.

- Optimyz WebServiceTester 3.0[2]
  Optimyz WebServiceTester 3.0 EA unterstützt BPEL4WS und WS-Security. Die Untersuchung der funktionalen Korrektheit, sowie die Skalierbarkeit eines Business Process Flows ist ein komplexer Sachverhalt. Den Flaschenhals (bottleneck) einer Anwendung zu finden ist eine der wichtigsten Aufgaben zur Verbesserung eines Zusammengesetzten Webservices. Durch die Unterstützung von WS-Orchestration, basierend auf der BPEL4WS Spezifikation, können grafische Darstellungen der Abhängigkeiten erreicht werden. WebServiceTester stellt zahlreiche Lösungen für HTTP Authentication, SOAP Authentication, Binary Security Tokens, XML Signatures und XML Encryption bereit.
  Die Kosten belaufen sich auf $499 pro User und Jahr, beziehungsweise auf $199 pro User und Vierteljahr.

[1] http://www.unitedbinary.com/harnessit.aspx
[2] http://www.optimyz.com

- SOAPscope[3]
  SOAPscope 3.0 Web Services Diagnostics System mit Microsoft Visual Studio .Net Integration ist ein plattformunabhängiges Webservice Diagnosesystem für das Untersuchen, Debuggen, Testen und Tunen von Webservices. Insbesondere im Schwerpunkt der Interoperabilität versucht SOAPscope zu punkten. Durch die einfache grafische Möglichkeit in Visual Studio Messagelogs auszuwerten, können Schwachpunkte relativ einfach gefunden werden. WSDL- und Nachrichten-Analysetools erleichtern es Entwicklern, Testern und Supportleuten die genaue Fehlerstelle im Programmlebenszyklus zu erkennen. Durch den Einsatz von WS-I Testing Tools können auch auf Java und C# basierende Werkzeuge eingesetzt werden. Auch eine Unterstützung für Secure Messaging, und andere Advanced HTTP Features inklusive SSL/TLS und Kompression (gzip/deflate) ist vorhanden. Der Preis von $99 pro Jahr beinhaltet alle Updates und Upgrades.

- SOAPtest[4]
  SOAPtest 2.1 kann Webservices testen, welche den ebXML Message Service implementieren. Falls eine WSDL Beschreibung des Services verfügbar ist, kann SOAPtest den WSDL-Code mit dem ebXML Schema validieren. SOAPtest kann auch als Proxy zwischen Client und Webservice dienen, um so die Schwachstellen herauszufinden. Durch die Unterstützung von BPEL4WS, SOAP und ebXML ist SOAPtest hervorragend für den Einsatz im E-Commercebereich geeignet. SOAPtest 2.1 läuft unter Windows 2000/XP, Linux und Solaris und kostet ab $3495.

- Sitescope[5]
  Sitescope ist ein reines Monitoringtool, welches allerdings an verschiedenen Orten eingesetzt werden kann. Es dient vor allem der Überwachung von weit verteilten Systemen. Die Preise für Sitescope starten bei $3000.

- Testmaker 3.2[6]

  TestMaker 3.2 ist das erste freie Open-source Werkzeug und Framework um Web Services (SOAP, HTTP, HTTPS, XML-RPC) und auch Emailsysteme (SMTP, POP3, IMAP) auf Skalierbarkeit, Performance und

[3]http://www.mindreef.com
[4]http://www.parasoft.com/soaptest
[5]www.freshwater.com
[6]http://www.pushtotest.com/ptt

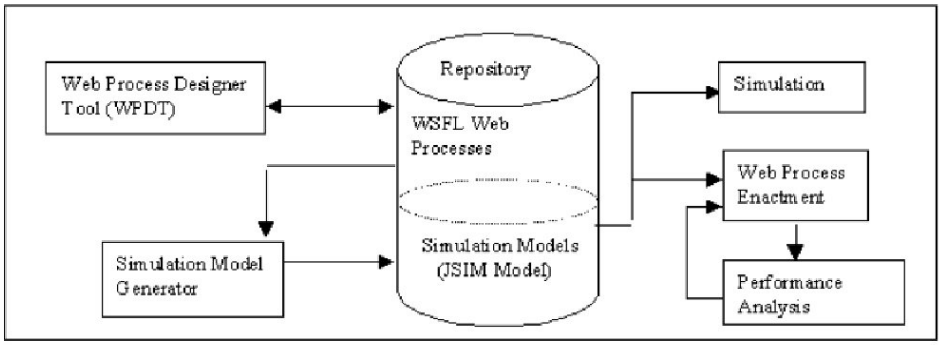

Abbildung 3.1: Schematischer Aufbau der System-Architektur von JSim [4]

Zuverlässigkeit zu überprüfen. Bei der Überprüfung der Mail-Protokolle können einfache Textnachrichten, aber auch MIME-Attachments überprüft werden. Es können auch Cookies für HTTP und HTTPS in den Versionen 1 und 2 gemäss der RFC 295 cookie handling Spezifikation bearbeitet werden

- KarmaSIM
  KarmaSIM [7] ist aus dem FrameNet-Projekt heraus entstanden. Eine Java-Anwendung, der DAML-S Interpreter, liest aus DAML-S Dateien die Daten aus, und erschafft so eine Netzwerkbeschreibung. Der KarmaSIM Simulator kann dann diese Netzwerkbeschreibung grafisch darstellen. Eine eingehende Beschreibung von KarmaSIM und dessen Grundlagen erfolgt im folgenden Kapitel.

- JSim
  Im Bereich der Business Prozesse sind Workflow Management Systeme am besten geeignet um die Geschäftsprozesse zu simulieren. Im betrachteten Worklflow QoS sind Zeit und Kosten die entscheidenden Kriterien. Beim Modell in sind vier Punkte wichtig: Zeit, Kosten, Zuverlässigkeit und Genauigkeit (Fidelity) [3]. Während Zeit, Kosten und Zuverlässigkeit messbare Größen sind, ist die Genauigkeit nur ein sehr schwer messbarer Begriff.
  DAML-S könnte eigentlich am besten die Semantik von Geschäftsprozessen widerspiegeln, allerdings haben sich die Entwickler für WSFL entschieden, da hierdurch dem Workflow mehr Rechnung getragen wird. In Abbildung 3 kann man die Systemarchitekur erkennen. Für weitere Erklärungen zu JSim siehe [1].

Kapitel 4

KarmaSIM

Wie bereits erwähnt entstand KarmaSIM im Zusammenhang mit FrameNet. Beim FrameNet-Projekt geht es darum, eine semantische Wissensdatenbank mit Ontologien und Lexikas zu erschaffen.
„Hypothesis: People understand things by performing mental operations on what they already know. Such knowledge is describable in terms of information packets called frames.[8]“
„For every target word describe the frames or conceptual structures which underlie them, and annotate example sentences that cover the ways in which information from the associated frames are expressed in these sentences.[8]“
Die Daten werden in einer MySQL-Datenbank verwaltet, welche mittlerweile 7700 lexikalische Einheiten mit 130.000 Sätzen und 550 Frames umfasst. Ein solcher Frame stellt der Gerichtsfall dar, welcher in Abbildung 4.1 schematisch skizziert ist.

4.1 Daml-S

Für den Austausch von Nachrichten ist eine MySQL-Datenbank nur beschränkt einsetzbar. Das World Wide Web (WWW) bietet eine Fülle von Möglichkeiten, Daten zu repräsentieren. HTML ist für Menschen zwar lesbar, aber für Maschinen nicht umsetzbar. XML bietet die Möglichkeit, durch Tags Daten zu charakerisieren. Ontologien sind am besten geeignet um die Semantik darzustellen. Eine Möglichkeit bietet hierfür die DARPA Agent Markup Language (DAML)[5]. Eine auf DAML aufbauende Sprache ist DAML+OIL mit den für Webservices ausgelegten Zusatz DAML-Services (DAML-S) für Ontologien. Der Service beruht auf der Web Service Defintion Language (WSDL). Der schematische Aufbau ist in Abbildung 4.2 aufgezeigt. Im November 2003 ist eine Umbenennung von DAML-S in OWL-S erfolgt. Zur

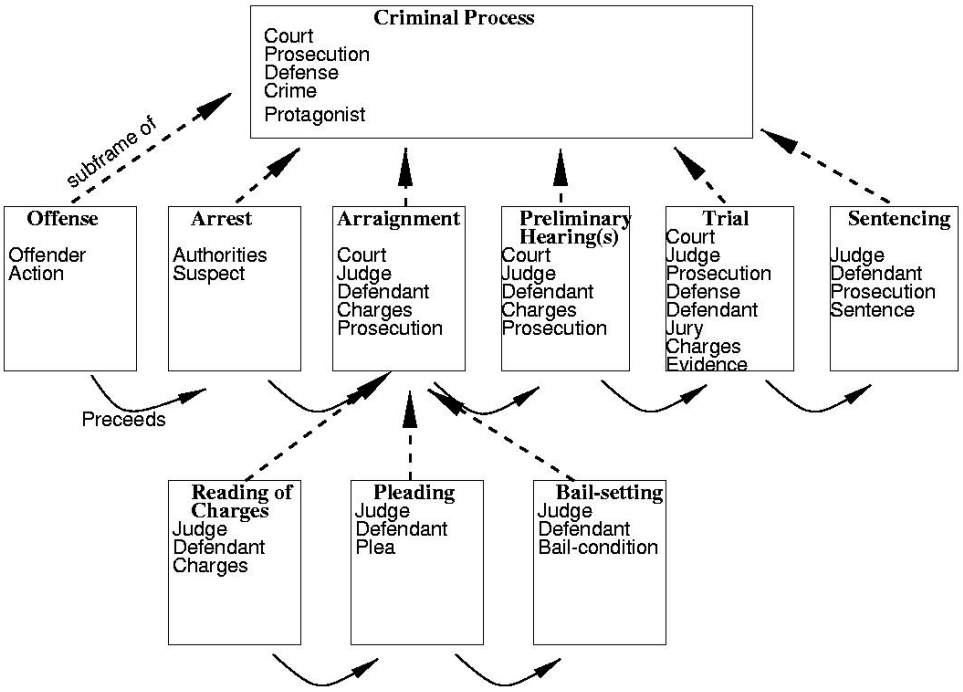

Abbildung 4.1: Gerichtsprozess (schematische Darstellung) [8]

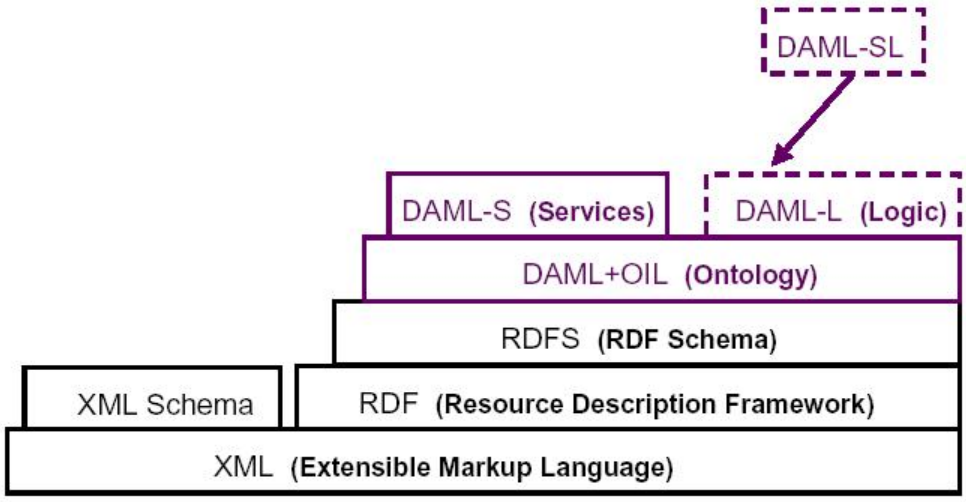

Abbildung 4.2: Schematischer Aufbau[10]

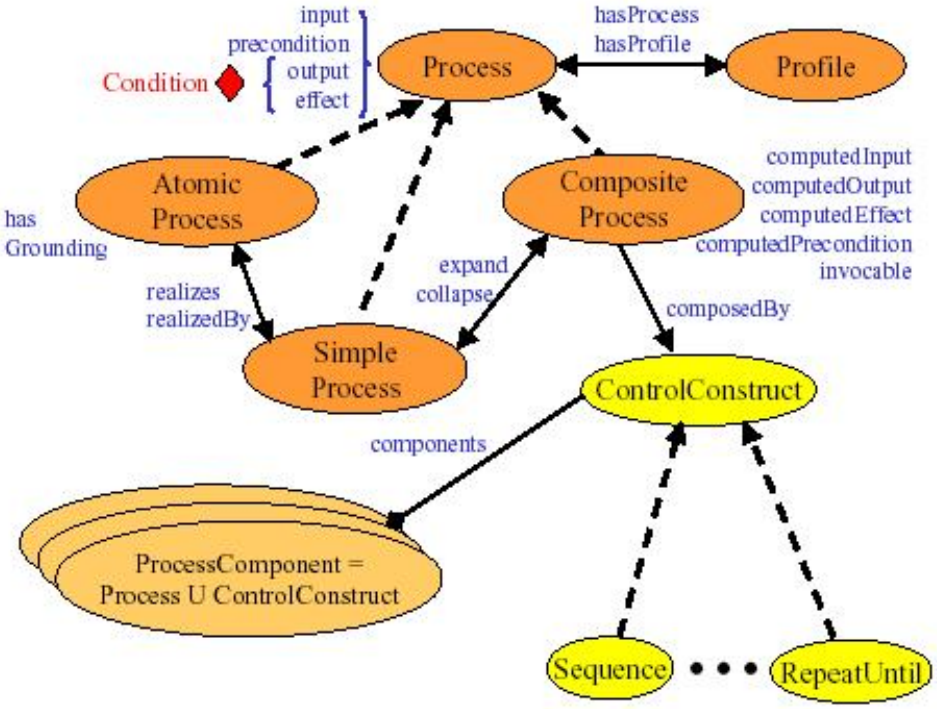

Abbildung 4.3: Top Level der Prozess Ontologie [11]

Zeit existieren 282 Anwendungsontologien, wovon die meisten im militärischen Bereich angesiedelt sind. Jedoch haben auch beispielsweise universitäre Einrichtungen Ontologien für ihre Vorlesungen entwickelt.

4.1.1 DAML-S Prozesse

DAML-S betrachtet die Funktionsweise von Webservices als Prozesse. Es existieren drei verschiedene Arten von Prozessen:

- Atomare Prozesse
  Ein atomarer Prozess ist ein nicht weiterzerlegbares Programm. Es wird durch einen einzelnen, beispielsweise http-Aufruf ausgeführt und gibt eine Antwort zurück. [9, 11]

```
Struktureller Aufbau:
<daml:Class rdf:ID="AtomicProcess">
   <daml:subClassOf rdf:resource="#Process"/>
</daml:Class>
```

- Einfache Prozesse
  Einfache Prozesse, im englischen Simple Process, werden als Abstraktionselemente benutzt. So können einfache Prozesse zum Beispiel als einfache Sicht eines zusammengesetzten Prozesses betrachtet werden. Im Beispiel 1 wird ein einfacher Prozess zu einem zusammengesetzten Prozess erweitert. Die Umkehrung dieser Erweiterung wird bereits mit übergeben.

```
Struktureller Aufbau:
<daml:Class rdf:ID="SimpleProcess">
  <daml:subClassOf rdf:resource="#Process"/>
</daml:Class>

Beispiel 1:
<rdf:Property rdf:ID="expandsTo">
  <rdfs:domain rdf:resource="#SimpleProcess"/>
  <rdfs:range rdf:resource="#CompositeProcess"/>
  <daml:inverseOf rdf:resource="#collapsesTo"/>
</rdf:Property>
```

- Zusammengesetzte Prozesse
  Zusammengesetzte Prozesse bestehen aus mehreren atomaren oder zusammengesetzten Prozessen, sowie verschiedenen Kontrollstrukturen[10].

```
<daml:Class rdf:ID="CompositeProcess">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Process"/>
    <daml:Restriction daml:minCardinality="1">
      <daml:onProperty rdf:resource="#composedOf"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>

<rdf:Property rdf:ID="composedOf">
```

```
    <rdfs:domain rdf:resource="#CompositeProcess"/>
    <rdfs:range rdf:resource="#ControlConstruct"/>
  </rdf:Property>
```

Der Zusammenhang zwischen den einzelnen Prozessen und dem gesamten Prozessmodell wird in Abbildung 4.3 deutlich.

4.1.2 Kontrollstrukturen in DAML-S

Damit Prozesse zusammengefügt werden können, müssen sie über gewisse Kontrollstrukturen verfügen. DAML-S stellt hiervon acht zur Verfügung [11].

- Sequenz:
  Bei einer Sequenz werden die einzelnen Prozesse hintereinander ausgeführt.
- Split:
  Split ist vergleichbar mit dem Fork-Befehl aus anderen Programmiersprachen. Es werden ausgehend von einem Prozess mehrere unabhängige Prozesse ausgeführt.
- Split-and-Join:
  Bei der Split-and-Join-Kontrollstruktur wird an einem bestimmten Punkt in der Programmausführung auf eine Synchronisation gewartet.
- Unordered:
  Die Prozesse werden bei einer Unorderd-Kontrollstruktur willkürlich nacheinander ausgeführt.
- Choice:
  Es wird aus mehreren Prozessen ein Prozess ausgewählt.
- Iterate:
  Prozesse werden solange ausgeführt bis eine Bedingung falsch wird.
- If-Then-Else:
  Es werden bedingte Verzweigungen ausgeführt.
- Repeat-Until:
  Prozesse werden solange ausgeführt bis eine Bedingung wahr wird.

4.1.3 Datenfluss

Jeder Prozess ist an bestimmte Parameter gebunden. In DAML-S existieren die formalen Parameter Input und (bedingter) Output, sowie Vorbedingung und (bedingter) Effekt. Während der Input und die Vorbedingung entweder zwingend oder optional vorgschrieben sind, sind der Output und der Effekt eines Prozesses meist bedingt. Zum Beispiel kann eine Anfrage ob ein Buch in der Bibliothek vorhanden ist, mit Ja oder Nein benatwortet werden. Diese Antwort ist ein bedingter Output. Eine Vorbedingung wäre beispielsweise, dass der Buchname, also der Inputparameter des Prozesses, dem Agenten bekannt ist. Ein Effekt oder auch Nachbedingung wäre beispielsweise beim Bücherkauf, sofern das Buch vorrätig und man genug Geld zum Erwerb desselbigen hatte, dass man anschliessend das Buch besitzt. Dies wäre ein bedingter Effekt.

4.2 Situation Calculus Language

Da die DAML+OIL Sprache nicht ausreicht um in DAML-S alle und nur die gewünschten Interpretationen auszudrücken, musss eine weitere Sprache hinzugenommen werden. Wir verwenden hierzu die Situation Calculus Language, welche eine logische Spracher erster Ordnung ist und in der die Veränderungen einer sich dynamisch veränderenden Welt das direkt Resultat einer von einem Agenten ausgelösten Aktion ist. [10, 79] Ausgehend von einer Anfangssituation S_0 sind alle anderen Situation Folgen einer Sequenz von Aktionen. Aktionen werden mit a(y) bezeichnet und entsprechen den atomaren Prozessen in DAML-S. Wenn man sich in der Situation S befindet und man Aktion a(y) ausführen möchte dann entspricht das der Funktion do(a,s). Der Parameter y entspricht den Inputparametern in DAML-S. Poss(a(y),s) symbolisiert die mögliche Ausführung in Abhängigkeit vom Paramter y.
Preconditons und Inputs können durch einfache Formeln, sowie Effekte und Outputs durch Axiome beschrieben werden. Für zusammengesetzte Prozesse ist eine vollständige Definition und Verständniss der Situation Calculus Language nicht nötig, da das Programm Golog eine automatische Umwandlung von DAML-S in die Situation Calculus Language vornimmt.[10, Seite 80]

4.3 Petri-Netze

Ein Petri-Netz ist eine ausführbare Semantik von DAML-S. Durch die Erstellung einer Teilmenge von DAML-S mit Hilfe der Situation Calculus Language ist es uns möglich, einfache Petri-Netze zu erstellen. Als Begründung liefern

Narayanan und McIlraith: *„We selected Petri Nets for its combination of compelling computational semantics, ease of implementation, and its ability to address both offline analysis tasks such as Web service composition and online execution tasks such as deadlock determination resource satisfaction, and quantitative performance analysis. We also note the existence of several well-known techniques mapping from Petri Nets to process logics and vice versa[10].“*

4.3.1 Definition von Petri-Netzen

Ein Petri-Netz ist eine algebraische Struktur (P, T, I, O) bestehend aus:

- Endliche Anzahl an Stellen, P = p1, p2, ... ,pn
- Endliche Anzahl an Transitionen, T = t1, t2, ... , tm
- Transition Input Funktion I
- Transition Output Funktion O

„Graphisch werden Stellen als Kreise und Transitionen als Rechtecke (manchmal auch als Striche) dargestellt[6].“ Die Inputfuntkion I und die Outputfunktion O wird durch eine gerichetete Kante symbolisiert. Jede Transition kann mehrere Input- und Outputfunktionen besitzen.
Dynamische Eigenschaften von Petri-Netzen werden durch Markierungen (Tokens) dargestellt. Stellen erhalten eine gewisse Anzahl an Markierungen, beginnend mit einer Initialbelegung. Falls die grafische Repräsentation nicht ausreicht, werden die Anzahl der Markierungen durch Zahlen repräsentiert. Zustandsänderungen werden durch Schalten (firing) von Transitionen dargestellt. Eine Transition kann schalten, wenn die Schaltregeln (firing rules) erfüllt sind. Eine solche Schaltregel ist, dass alle Stellen des Vorbereichs einer Transition mindestens eine Markierung besitzen. Nach Ausführung einer Schaltung wird aus jeder Stelle des Vorbereiches eine Markierung entnommen und im Nachbereich in jeder Stelle eine solche Markierung erzeugt. Es können auch mehrere Markierungen entnommen und hinzugefügt werden[6].

4.4 Anwendungsbeispiel KarmaSIM

Karmasim ist ein auf Petri-Netzen beruhendes Simulations- und Umgebungsmodellierprogramm. Es existiert ein Übersetzer von DAML-S über Situation-Calculus zu einem Input für Karmasim. Das KarmaSIM Werkzeug erlaubt eine interaktive Simulation und unterstützt die verschiedenen Verifikations- und Performanceanalysetechniken[10].

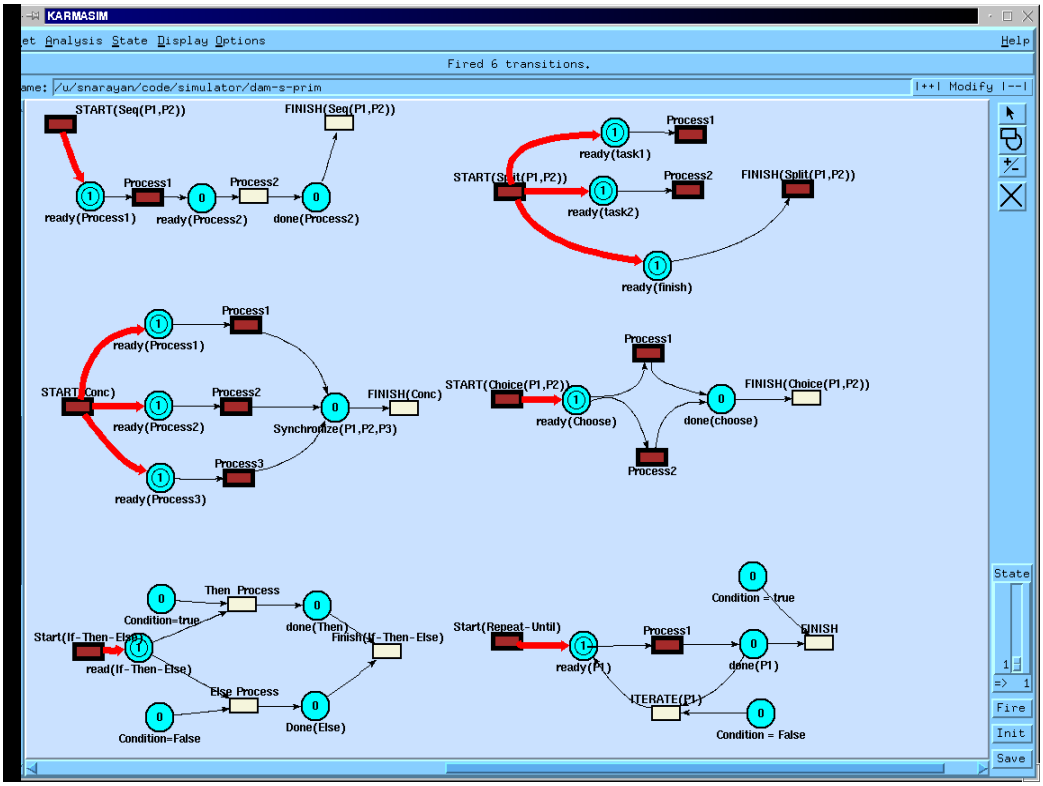

Abbildung 4.4: Einzeldarstellung von Karmasimfunktionen

4.4.1 Umsetzung von DAML-S in KarmaSIM

Wie in Abschnitt 4.1.2 bereits erwähnt, besitzt DAML-S acht Kontrollstrukturen für zusammengesetzte Prozesse. Bis auf die unordered Kontrollstruktur können alle Strukturen durch Petrinetze, und somit in KarmaSIM simuliert werden. Die iterate und repeat-until Funktion können durch Negation der Bedingung in eine Transistion umgewandelt werden. In Abbildung 4.4 sind alle sechs Kontrollstrukturen und ihre Repräsentation in KarmaSIM abgebildet. Die roten, dicken Pfeile entsprechen dem Ergebnis einer abgefeuerten Transition und Markierungstransfer zwischen den Stellen. Die braun ausgefüllten, dicken Transitionen symbolisieren die eingeschaltete Transition.

4.4.2 Gerichtsprozess

Um auf das Beispiel des Gerichtsprozesses zurückzukommen, sind in Abbildung 4.5 und Abbildung 4.6 die Umsetzung des Gerichtsprozesses in KarmaSIM dargestellt. Der genauer Ablauf eines Gerichtsprozesses wird repräsen-

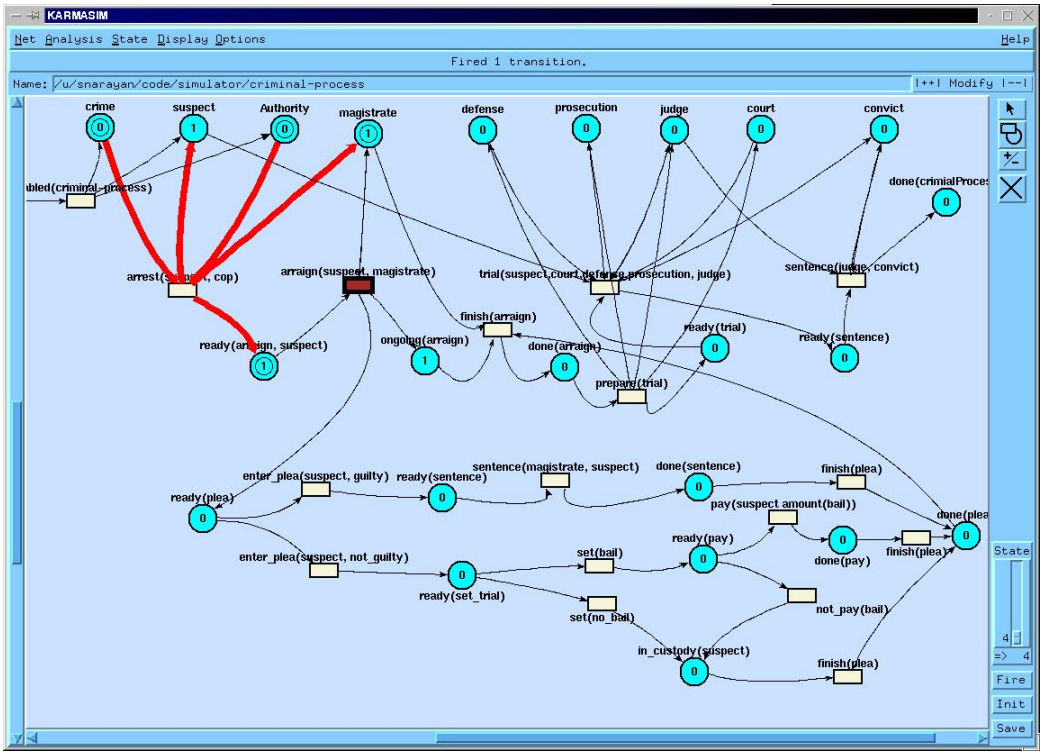

Abbildung 4.5: Gerichtsprozess 1

tiert. Zunächst kommt die verhaftung des Angeklagten, anschliessend eine Anhörung und zum Schluss noch die Verhandlung mit Urteil. Dies ist im oberen Bildbereich dargestellt. Im unteren Bildbereich wird nach der Anhörung über eine eventuelle Haftentlassung und Kaution entschieden. Alle Prozessschritte und Prozessbeteiligten können nur in den von KarmaSIM dargestellten Zusammenhängen und Reihenfolge auftreten. In Abbildung 4.5 ist der Anfang des Gerichtsprozesses dargestellt. Hier bei wird ausgehend vom Verbrechen eine Verhaftung des Verdächitgen vorgenommen. Der Richter und die Staatsgewalt (Cop) sind bei einer solchen Verhaftung ebenfalls involviert. Die nächste zu erfolgende Transition ist die Anhörung.
Abbildung 4.6 stellt die Situation bei der Verkündung des Urteils dar.

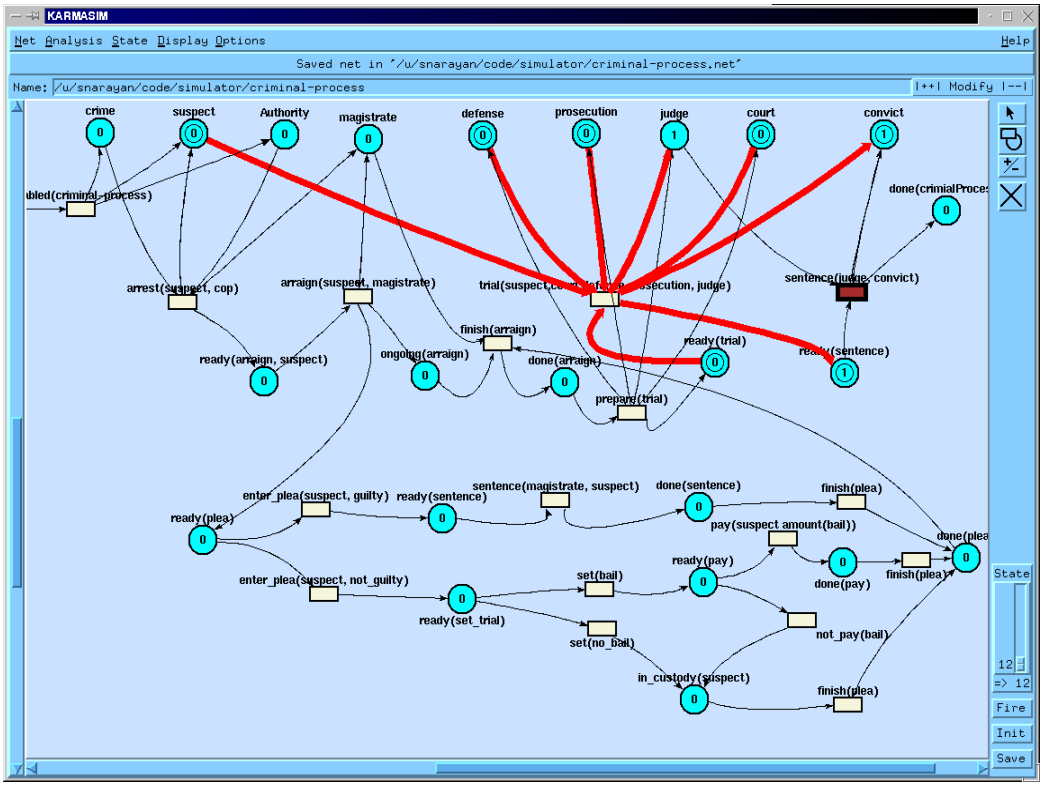

Abbildung 4.6: Gerichtsprozess 2

Kapitel 5

Ausblick

Zusammenfassend kann man zum Schluss kommen, dass Simulationen helfen, Kosten zu sparen. Es gibt allerdings nur sehr wenige, dafür aber teure, Softwarepakete welche alle Teilaspekte der QoS-Anforderungen abdecken. Im Bereich der Webservice Entwicklung haben sich einige Simulations- und Testprogramme bewährt und konnten so eine schnelle Entwicklung von Webservices, und somit auch niedrige Entwicklungskosten, ermöglichen. Durch den Einsatz von XML und ihrer Untersprachen, ist es dem Menschen nur sehr schwer möglich, auf einen Blick den Inhalt eines Webservices zu erkennen, währen Maschinen ihn problemlos bei korrekter Implentierung ausführen können. Daher stellen die meisten Simulatoren eine grafische Visualisierung der ausgetauschten Nachrichten oder des ganzen Webservices zur Verfügung. Bedingt durch die Sprachvielfalt für die Programmierung von Webservices können die meisten Simulatoren nur mit einer speziellen Sprache umgehen. Allerdings existieren Tools um zum Beispiel aus einer WSDL-Datei eine DAML-S konforme Datei zu erstellen. Viele kommerzielle Anbieter scheuen aber momentan noch das Risiko der Entwicklung umfassender Simulationsprogramme, da sich, außer XML als Grundlage, noch keine einheitliche und standardisierte Sprache herauskristallisiert hat, zumal bei einigen Sprachen das Standardisierungsverfahren noch nicht abgeschlossen ist. Allerdings lassen sich schon heute dynamische Prozesse von Web-Services simulieren, und auch für Interoperabilitätstests stehen Simulationsprogramme zur Verfügung. Ein Hauptproblem beim Testen der Interoperabilität ist die mangelnde Veröffentlichung von Webservices mittels UDDI. Viele große Firmen verwenden innerhalb des Intranetzes Webservices, stellen diese jedoch nicht immer öffentlich zur Verfügung.
Ein bisher noch nicht betrachteter Blickwinkel zur Simulation von Webservices ist die Simulation durch Webservices selbst. Erste Ansätze sind in einigen Forschungsarbeiten erarbeitet worden, Simulationen von Anwendungen

durch Webservices syelbst zu realisieren. Dieser Ansatz erscheint insofern lobenswert, da bei Veröffentlichung solcher Simulatoren viele Entwickler mit ähnlichen Problemen auf schnellsten Wege zu ausreichend getesteten Simulatoren kommen können. Da die Vielzahl der Entwickler von Software auch einen Zugang zum Internet haben, kann so unter realistischen Bedingungen die eigene Software getestet werden, ehe sie zum, eventuell kommerziellen, Einsatz freigegeben wird.

Literaturverzeichnis

[1] Alvandi, A. Simulation von Web Services. Universiät Mannheim, 2004.

[2] Arun Nagarajan, A. M. Understanding quality of service for Web services. *IBM developer Works* (Jan. 2002). http://www-106.ibm.com/developerworks/library/ws-quality.html.

[3] Cardoso, J., Sheth, A., and Miller, J. Workflow Quality of Service. Tech. rep., LSDIS Lab, Department of Computer Science, University of Georgia, März 2002.

[4] Chandrasekaran, S., Silver, G., Miller, J. A., Cardoso, J., and Sheth, A. P. Web service technologies and their synergy with Simulation. In *Proceedings of the 2002 Winter Simulation Conference* (2002), E. Yucesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, Eds.

[5] DAML-S and related Technologies. DAML-Homepage, http://www.daml.org/services, 2004.

[6] Herrtwich, R. G., and Hommel, G. *Kooperation und Konkurrenz: nebenläufige, verteilte und echtzeitabhängige Programmsysteme.* Springer, Berlin; Heidelberg; New York; London; Paris; Tokyo; Hong Kong, 1989.

[7] Homepage von Karmasim. http://www.ai.sri.com/daml/services.

[8] Narayanan, S., Fillmore, C., Baker, C., and Petruck, M. Framenet meets the semantic web. http://www.icsi.berkeley.edu/framenet.

[9] Narayanan, S., and McIlraith, S. Analysis and Simulation of Web Services, 2002.

[10] Narayanan, S., and McIlraith, S. Simulation, verification and automated composition of web services. In *Proceedings of the Eleventh International World Wide Web Conference (WWW-11)* (May 2002).

[11] the OWL Services Coalition. OWL-S: semantic Markup for Web-Services. http://www.daml.org/services, 2004.