Architektur für verteilte, agentenbasierte Workflows

Christine Reese, Sven Offermann, Daniel Moldt
Fachbereich Informatik
Universität Hamburg
{reese, offermann, moldt}@informatik.uni-hamburg.de

Zusammenfassung

Die Koordination der Zusammenarbeit zwischen räumlich und zeitlich getrennten Unternehmensabteilungen und Geschäftspartnern stellt einen wichtigen Aspekt bei der Realisierung von Informationssystemen für global agierende und kooperierende Unternehmen dar. Um entsprechende verteilte Anwendungen umzusetzen, werden verschiedene verbreitete Konzepte und technische Mittel genutzt. Die Koordination von auszuführenden Aktivitäten um Geschäftsziele zu erreichen ist die Anwendungsdomäne von Workflow-Management-Systemen (WFMS). Beim Einsatz von WFMS zur Koordinierung von Abläufen zwischen selbstständigen und wechselnden global verteilten Organisationseinheiten ergeben sich besondere Anforderungen im Bereich der Flexibilität, der Autonomie und der Koordination entsprechender Systeme.

Basierend auf der formalen Technik der höheren Petrinetze stellt diese Arbeit eine konzeptionelle, auf Agenten basierende Workflow-Architektur vor. Eine sich aus der Kombination dieser formalen Ansätze ergebende innovative Architektur wurde bereits in [ROO+05] vorgestellt. In dieser Arbeit wird zum einen eine Architekturerweiterung zur Integration existierender Informationssysteme auf Basis von WebServices vorgestellt. Zum anderen wird die Architektur im Hinblick auf interorganisationale Workflows präzisiert.

Keywords: Verteilte Workflows, WebServices, Agenten, verteilter Workflow-Enactment-Service, höhere Petrinetze, Capa, Renew

1 Einleitung

In Folge der Globalisierung sind länderübergreifend agierende Unternehmen zunehmend gezwungen, ihre Organisationseinheiten weltweit zu verteilen. Die Verteilung auf unterschiedliche Länder und Zeitzonen führt dabei zu einem räumlich und zeitlich verteilten Arbeiten. Die Erledigung von unternehmensweiten Aufgaben erfordert klare Konzepte zur Koordinierung dieser räumlich und zeitlich verteilten Tätigkeiten. Dies ist die typische Anwendungsdomäne von Workflow-Management-Systemen (WFMS). Eine weitere Anwendungsdomäne solcher

Systeme stellt die Koordinierung der interorganisationalen Arbeitsabläufe zur flexiblen Zusammenarbeit zwischen kooperierenden Unternehmen dar, die ein gemeinsames Geschäftsziel erreichen wollen. Solche interorganisationalen Geschäftsprozesse entstehen z.B. bei der ganzheitlichen Betrachtung von Zulieferketten oder bei der Steuerung von Kooperationsprozessen im Rahmen von virtuellen Unternehmen.

Die Koordinierung weltweit verteilter Aktivitäten zwischen wechselnden eigenständigen Partnern stellt neue Anforderungen an ein WFMS. Entsprechende Systeme müssen die hohen Anforderungen an Flexibilität, Autonomie und Koordinierung erfüllen. Sie müssen flexibel an die wechselnden Beziehungen zwischen Geschäftspartnern und die sich ständig ändernden Marktbedingungen anpassbar sein als auch die Autonomie der kooperierenden Geschäftspartner berücksichtigen sowie die Anforderungen aufgrund einer zeitlich und räumlich getrennten Zusammenarbeit erfüllen. Diese Anforderungen müssen im Rahmen einer Architektur für ein entsprechendes WFMS berücksichtigt werden.

Im Folgenden wird eine agentenbasierte Architektur für ein verteiltes Workflow-Management-System zur Koordinierung interorganisationaler Workflows vorgestellt. Dabei wird zwischen der Verteilung innerhalb eines Intranets und der Verteilung über Organisationsgrenzen hinweg unterschieden. Zur Modellierung von Workflows werden Petrinetze verwendet. Die Modellierung von Workflows mittels Petrinetzformalismen ermöglicht eine formale und trotzdem intuitive grafische Modellierung. Die klare Semantik der Petrinetze bietet den Vorteil der direkten Ausführbarkeit der Workflows sowie die Möglichkeit einer formalen Verifikation der Konsistenz von Abläufen. Die Verwendung von Agentenkonzepten unterstützt hingegen die Modellierung flexibler verteilter Systeme mit kommunizierenden autonomen Akteuren.

Die konzeptionellen Grundlagen der vorgestellten Architektur wurden bereits an anderer Stelle vorgestellt (siehe [ROO+05]). Die grundlegende Idee besteht darin, Workflows und Workflow-Fragmente innerhalb von Agenten zu kapseln. Andere typische Komponenten eines Workflow-Management-Systems werden konzeptionell ebenfalls als Agenten modelliert. Die Kapselung in Agenten unterstützt zum einen aufgrund des Mobilitätsaspekts von Agenten eine flexibele Verteilung von Workflow-Fragmenten. Zum anderen unterstützt die Autonomie von Agenten die Realisierung von Sicherheitsaspekten, indem Agenten konzeptionell flexibel über die Herausgabe von Daten entscheiden können. Die Synchronisation von Teilworkflows erfolgt durch Kommunikation zwischen den Agenten. Ein WFMS-Agent dient anderen Agenten des Systems als Plattform, kapselt die beteiligten Agenten und bildet so eine Workflow-Agentenplattform. Durch eine mehrfach hierarchische Kapselung können mehrere WFM-Systeme intra- oder interorganisational zu einem logischen WFMS zusammengefasst werden, wobei die Koordination von Workflows und der sichere Datenfluss auf mehreren Ebenen kontrolliert werden kann.

Zur Realisierung der Architektur existieren bereits Werkzeuge: Das Petrinetzwerkzeug Renew mit Editoren und Simulatoren für Referenznetze. Basierend auf Renew existiert weiterhin eine FIPA-kompatible Agentenplattform Capa, welche eine konzeptionelle Modellierung und Implementierung von Agen-

tensystemen mittels Petrinetzen ermöglicht. Im Zusammenhang mit Renew existiert außerdem bereits eine petrinetzbasierte Workflow-Umgebung inklusive Client-Anwendung, Rollenverwaltung, persistenter Workflow-Engine und Definitionswerkzeug. Abschließend existiert für die Agentenplattform Capa die Realisierung eines Gateway-Agenten, welcher eine Kommunikation zwischen WebServices und Agenten unterstützt und damit die Integration von WebServices in Agentensysteme ermöglicht.

Der weitere Text ist wie folgt gegliedert. Kapitel 2 erläutert die grundlegenden Konzepte und technischen Basisbestandteile der WFMS-Architektur. Darauf aufbauend führt Kapitel 3 das Gesamtsystem aus verschiedenen Perspektiven ein: die Abhängigkeiten der verwendeten Software-Komponenten werden dargestellt, eine Verfeinerung in einzelne Agenten und deren Aufgaben erläutert. Schließlich werden einzelne wichtige Aspekte und Lösungsansätze eines laufenden Systems aufgegriffen. Kapitel 4 dient abschließend einer Zusammenfassung und einem Ausblick auf weiterzuführende Arbeiten.

2 Konzeptioneller und technischer Hintergrund

Abbildung 1 zeigt einen ersten Überblick über die Komponenten der vorgestellten Architektur. Eine einfache Variante wurde bereits in [ROO+05] veröffentlicht. Die Laufzeitumgebung besteht aus Java und dem Petrinetzsimulator Renew. Die Workflow-Engine, die Agentenplattform und das WebService-Gateway laufen in dieser Umgebung. Auf der Agentenumgebung und der petrinetzbasierten Workflow-Engine aufbauend werden, dem Workflow-Referenzmodell der WfMC (Workflow Management Coalition, siehe [Wor05]) folgend, Agenten entwickelt, welche die einzelnen funktionalen Aspekte eines WFMS abdecken und Dienste zur informationstechnischen Unterstützung von Geschäftsprozessen anbieten. Durch den WebService-Gateway-Agenten können Anwendungen konzeptionelle Eigenschaften aus der Agentendomäne (z.B. Architekturen für planende, intelligente Agenten) verwenden und trotzdem höhere, konzeptionelle Dienste über WebService-Schnittstellen anbieten.

Es folgt ein detaillierter Einblick in die Komponenten der WFMS-Architektur.

2.1 Referenznetze und Renew

Referenznetze sind höhere Petrinetze und bieten als solche einen Formalismus mit exakter Semantik, Methoden zur formalen Validierung sowie eine anschauliche grafische Darstellung. Insbesondere können nebenläufige Prozesse intuitiv mittels Petrinetzen dargestellt werden. Mittels Marken lassen sich Ressourcen und Daten abbilden. Diese liegen auf Stellen, welche über Kanten mit Transitionen verbunden sind. Transitionen entfernen Marken von Stellen und legen Marken auf Stellen ab und modellieren somit aktive Systemteile. Verschiedene Erweiterungen des ursprünglichen Petrinetzformalismus, wie z.B. gefärbte oder hierarchische Petrinetze, bieten Mechanismen zur kompakten Darstellung komplexer Systeme. Einige dieser Erweiterungen können mit (größeren) einfacheren

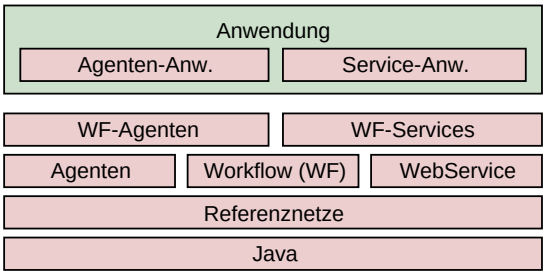

Abbildung 1: Übersicht: Komponenten der Architektur

Petrinetzen dargestellt werden, andere erweitern die Ausdruckskraft der Netze.

Referenznetze stellen einen erweiterten höheren Petrinetzformalismus dar. Sie basieren auf den gefärbten Petrinetzen und erweiten diese um die Konzepte der synchronen Kanäle und der Netze-in-Netzen. Synchrone Kanäle ermöglichen ein synchronisiertes Schalten von Transitionen und den Transfer von Daten zwischen Transitionen bzw. Netzen. Das Netze-in-Netzen Paradigma erlaubt die Verwendung von Netzinstanzen als Marken in anderen Netzinstanzen. Durch diese Erweiterungen unterstützen Referenznetze die Dekomposition komplexer Systemmodelle sowie insbesondere die Modellierung dynamischer Systeme. Als Werkzeug zur Modellierung und Simulation von Referenznetzen existiert das Petrinetzwerkzeug Renew. Renew bietet parallele Ausführung unabhängiger Netzteile durch internes Multithreading sowie umfangreiche Entwicklungsunterstützung durch Werkzeuge zur Inspektion eines laufenden Netzes und der Markierung. Die Verwendung einer Java-ähnlichen Anschriftensprache erlaubt die Ausführung beliebigen Java-Codes zum Zeitpunkt des Schaltens von Transitionen.[1]

2.2 Workflow-Netze

Die Verwendung von Petrinetzen zur Beschreibung von Workflows wurde gründlich untersucht (siehe [Aal97]). Die in [AHKB03] definierten typischen Workflow-Muster zur Steuerung des Kontroll- und Datenflusses (z.B. Sequenz, Split oder Join) können mit Petrinetzen dargestellt werden, wobei Transitionen als aktive Netzelemente Aktivitäten modellieren. Über Transitionsanschriften können Anwendungen aufgerufen werden oder Benutzer können aufgefordert werden Aufgaben zu bearbeiten. Das existierende Renew-Plugin für Workflows ([Jac02]) stellt neben den allgemeinen Möglichkeiten eines Workflow-Enactment-Dienstes auch eine Rollen- und Rechteverwaltung bereit. Es basiert auf einem Vorschlag

[1] Eine ausführliche Behandlung von Referenznetzen befindet sich in [Kum02] und [Kum01] gibt eine praktische Einführung. Gefärbte Petrinetze werden u.a. in [Jen92], Netze-in-Netzen in [Val98] und synchrone Kanäle in [CH92] dargestellt. Die Renew-Software, Handbuch und Literatur sind unter [Ren05] frei verfügbar.

für eine nebenläufige, petrinetzbasierte Workflow-Engine aus ([AMVW99]) und auf einer persistent arbeitenden Engine, die in [JKM01] vorgestellt wurde.

2.3 Fragmentierung der Workflownetze

In der vorangegangenen Veröffentlichung [ROO+05] wurde eine Fragmentierung für Workflownetze vorgestellt, die es erlaubt, unabhängige Fragmente zu erzeugen. Die Grenze zwischen Fragmenten wird an Stellen im definierenden Workflownetz gezogen. Die Fragmente synchronisieren sich an den Randstellen mit einem zentralen Steuerungsnetz und lösen auf diese Weise auch evtl. vorhandene verteilte Konflikte. Die Fragmentierung ist dabei nicht konzeptionell beschränkt, so kann ein Fragment beliebig viele Eingangs- und Ausgangsstellen haben. Die zusammengefügten Fragmente ergeben ein vollständiges Workflownetz mit den typischen Eigenschaften, wie z.B. einer eindeutigen Start- bzw. Endstelle. Es wird außerdem argumentiert, dass zu jedem generierten Fragment automatisch verfeinerte Randstellen hinzugefügt werden, welche die Funktionalität zur Synchronisation umsetzen.

2.4 Agenten

Das hier verwendete Agentenrahmenwerk Capa (Concurrent Agent Platform Architecture, siehe [DMR03]) basiert auf dem konzeptuellen Modell „Mulan“ (Multi-agent nets) für komplexe Multiagentensysteme. Capa erweitert dieses Modell um eine konkrete FIPA-konforme Kommunikationsschicht und bietet somit eine Plattform zur Realisierung petrinetzbasierter, FIPA konformer Agenten.

Capa stellt einen abstrakten Agenten als erweiterbaren Rahmen für die Agentenprogrammierung zur Verfügung. Damit sind Basisfunktionen wie Senden und Empfangen von Nachrichten und Zugriff auf die Wissensbasis bereitgestellt. Das Verhalten wird mit Protokollnetzen definiert, welche eine Workflownetz-artige Struktur aufweisen. Die Schnittstelle der Protokollnetze zum umschließenden Agenten ermöglicht die Definition von expliziten Anfangs- und Endpunkten, ein- und ausgehende Nachrichten sowie Zugriff auf die Wissensbasis des Agenten. Capa unterstützt weiterhin Mobilität [KMR03].

Die Spezifikation von Interaktionen zwischen Agenten kann mittels AUML-Interaktionsdiagramme dargestellt werden. Renew und seine Erweiterungen bieten Werkzeuge zum Zeichnen solcher Diagramme und zur automatischen Generierung skelettartiger Protokollnetze für Capa-Agenten, welche dann um konkrete Funktionen ergänzt werden (siehe hierzu [CMR03]).

2.5 Anbindung an WebServices

Es existieren verschiedene Ideen zur Verbindung von WebService-, Agenten- und Workflowtechnologien. Konzeptionell bietet Agententechnologie ein Begriffsgebäude für verteilte Systeme mit autonomen, intelligenten und flexiblen Komponenten. Auch im Bereich der WebServices und Workflows bemüht man sich um

mehr Flexiblität und Kapselung und erweitert die ursprünglichen für diese Gebiete typischen Funktionalitäten, so dass WebService- und Workflow-Systeme schließlich nur noch durch die verwendete Basis- und Kommunikationstechnik von Agentensystemen unterscheidbar sind. Ein analoger Gedankengang lässt sich von der WebService-Technolgie aus verfolgen: Basis von WebServices ist eine technische Infrastruktur zum Bekanntgeben, Auffinden und Nutzen von Diensten über Netzwerke. Analoge Technologien bestehen im Rahmen von Agenten z.B. mit den FIPA-Standards, welche ebenfalls eine Sammlung von Protokollen, Kommunikationssprachen und Verzeichnisdiensten spezifizieren. Mittels Übersetzungsdiensten sind die Standards weitestgehend konzeptionell austauschbar und kombinierbar.

Grundsätzlich gibt es verschiedene Ansätze, die auf jeweils ihre Weise Agenten und WebServices *technisch* integrieren. Zunächst gibt es in dieser Arbeitsgruppe eine Modellierung von WebServices und ihren Behältern mit Referenznetzen (siehe [MOO05]), welche die konzeptionellen Systemkomponenten aus Mulan (s.o.) auf WebServices überträgt. Die parallelen Ansätze sind in Abbildung 1 dargestellt. Weiter kann z.B. ein Agent, der einen Task ausführt, so programmiert sein, dass er mit Hilfe einer Java-API für WebServices (WSIF) einen WebService mittels Java-Anschriften benutzt. Ein dritter in dieser Arbeitsgruppe benutzter Ansatz ist die Realisierung eines Gateway-Agenten für Capa, der sowohl WebService- als auch Agentennachrichten versteht und diese übersetzt und weiterleiten kann. Wie in Abbildung 4 dargestellt ist, stellt der Gateway-Agent eine generische Plattformerweiterung dar, welche nicht nur im Rahmen der WFMS-Architektur verwendet werden kann.

Konzeptionell entspricht der erste Ansatz dem Gedankengang, WebServices mit einer flexiblen Architektur zu modellieren und der letztgenannte Ansatz dem Gedankengang, Agenten die Syntax von WebService-Aufrufen zugänglich zu machen.

2.6 Zur Beziehung zwischen Workflows und Agenten

In der vorgestellten Architektur werden Workflows von Agenten gekapselt. Die konzeptionellen Vorteile von Petrinetzen und Agententechnologie sollen kombiniert werden. Indem man eine Anwendung als Workflowsystem betrachtet, werden Aspekte der Steuerung und des Monitoring betont. Betrachtet man dieselbe Anwendung als Multiagentensystem, so werden dadurch Eigenschaften wie Autonomie, Kapselung und Flexibilität betont.

Indem (höhere) Petrinetze als Definitionssprache für zentrale Teile einer Anwendung verwendet werden (also für Workflow- und Agentenverhaltensdefinition), wird der Aspekt der Verifikation und der grafischen Modellierung hinzugefügt. Das bedeutet, dass Protokollnetze für Workflow-Agenten sowohl die Schnittstelle für Protokollnetze als auch Eigenschaften eines Workflows aufweisen müssen, ohne jedoch durch technische Einzelheiten vom Inhalt des Workflows oder Workflow-Fragments abzulenken.

Normalerweise existieren Workflows als Datenstruktur im System. Indem ein Workflow von einem Agenten gekapselt wird, ist er konzeptionell nur über

die Nachrichtenschnittstelle des Agenten zugreifbar (wobei hier nicht die allgemeinen Probleme und Herausforderungen im Bereich der Agentensicherheit diskutiert werden). So wird ein gewisser Grad an Autonomie und Mobilität *in der Architektur* bereitgestellt. Für normale Workflow-Anwendungen brauchen diese Workflow-Agenten keine Autonomie, aber diese hochflexible Architektur ermöglicht wesentlich komplexere Anwendungen, wie in der Einleitung skizziert.

2.7 Die Komponenten eines WFMS

Das Strukturmodell für Workflow-Management-Systeme, wie es von der WfMC spezifiziert wurde, definiert sechs grundlegende Komponenten eines WFMS, die in Abbildung 2 bildlich dargestellt sind. Im Folgenden wird beschrieben, wie diese im hier vorgestellten System realisiert werden.

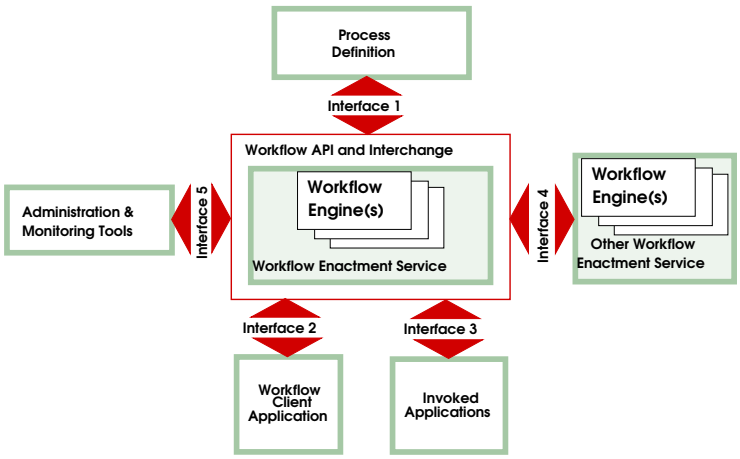

Abbildung 2: Referenzmodell der WfMC: Komponenten eines WFMS. Nach: [Wor05]

Ein *Process Definition Tool* ist Teil des bereits oben erwähnten Workflow-Plugins für Renew. Dieses wurde wie in [ROO+05] beschrieben erweitert, so dass nun Randstellen in einem Workflownetz definiert werden können, mit deren Hilfe das Workflownetz automatisch in Fragmente zerlegt wird, wodurch eine Verteilung möglich wird.

Eine *Workflow Client Application* ist ebenfalls im existierenden Plugin vorhanden und wird von einem Agenten gekapselt.

Invoked Applications werden von Agenten gekapselt. *Task-Agenten* können entweder eine Interaktion mit einem Anwender anstoßen, eine Aufgabe direkt erledigen oder eine Anwendung oder einen Dienst aufrufen.

Der *Workflow Enactment Service* wird in diesem System von dem existierenden Workflow-Plugin erbracht. Dieses wird hier von einem Agenten gekapselt

und funktioniert als eine Agentenplattform analog zur Agentenplattform Capa. Die Workflow-Agenten sind in diesem *enthalten*.

Die *Workflow Engines* sind ebenfalls durch Agenten gekapselt, die sich auf der Plattform des Workflow-Enactment-Service befinden. Sie koordinieren die Erledigung der Tasks, wie sie in einem Workflow oder einem Workflow-Fragment definiert werden, durch Delegation an den jeweils passenden Task-Agenten.

Monitoring wird von Agenten erledigt, die Informationen über laufende und beendete Tasks oder über problematische Situationen sammeln. Hier wird eine Sicht auf den Systemzustand entsprechend den Berechtigungen geboten, soweit dies in einem verteilten System möglich ist.

Die *Administration*, also Rollen-, Rechte- und Benutzerverwaltung wird von einem Agenten gekapselt.

3 Architektur und Infrastruktur

Die folgenden Abschnitte beschreiben unseren Entwurf von Workflow-Agenten, die wie in Abbildung 1 auf Workflow- und Agententechnologie aufbauen, sowie die Integration in existierende Implementierungen. Dazu werden zunächst die verwendeten Renew-Plugins mit ihren Beziehungen vorgestellt.

3.1 Abhängigkeiten zwischen Plugins

Die Abhängigkeiten der erwähnten Plugins für Renew sind in Abbildung 3 dargestellt. Renew ist vollständig „pluggable“, d.h. dass sogar die Kernfunktionen von Renew als austauschbare Plugins implementiert sind. Die Laufzeitumgebung des vorgeschlagenen Systems wird durch das Simulator-Plugin bereitgestellt. Das Workflow-Agenten-Plugin, das hier entworfen wird, baut auf dem Capa-Plugin und auf dem Workflow-Plugin auf. Die direkte Abhängigkeit zum Simulator von Renew resultiert aus der Fragmentierung der Workflow-Netze, da das Netzelement Stelle erweitert wurde.

Das Renew-Pluginsystem ermöglicht die Definition von *bedingten Abhänigkeiten*, also Plugins, die genutzt werden können, obwohl das Grundsystem auch ohne sie läuft. Dieses Prinzip wird für die GUI-Anbindung benutzt. Es gibt auch ein WebService-Plugin für Capa, das einen Gateway-Agenten hinzufügt. Dieser kann Nachrichten übersetzen und weiterleiten, so dass WebServices und Agenten kommunizieren können. Dieses kann vom Workflow-Agenten-Plugin benutzt werden.

Eine Anwendung, die auf dieses System aufbaut, nutzt das Workflow-Agenten-Plugin und Capa, evtl. auch direkt das WebService-Plugin. Die GUI einer solchen Anwendung kann auf dem GUI-Plugin von Renew aufbauen oder kann ihre eigene GUI erzeugen.

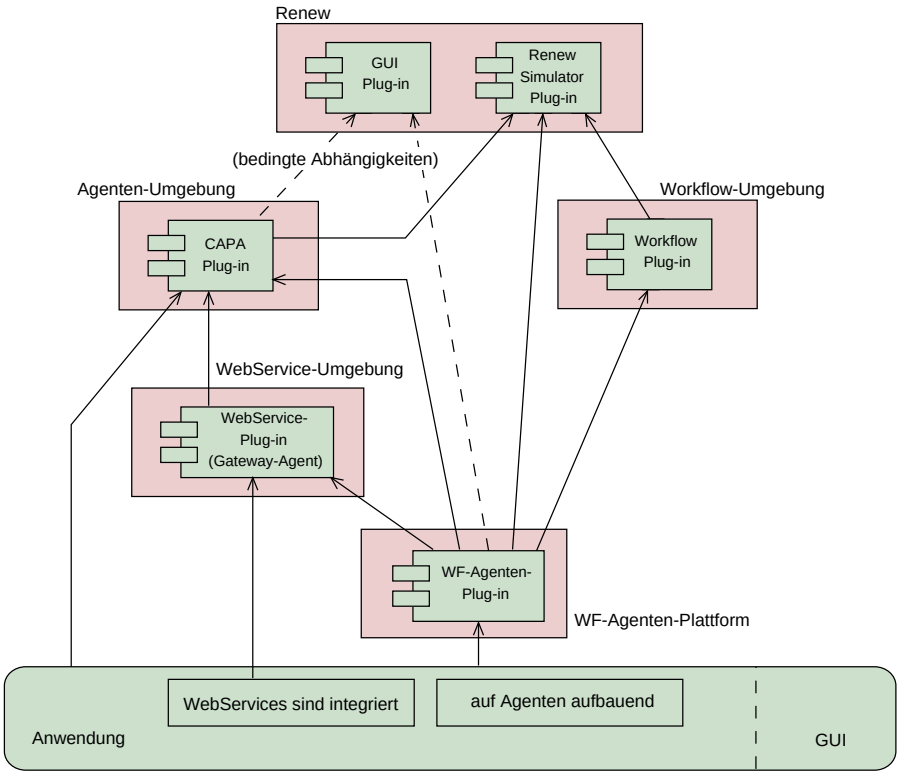

Abbildung 3: Abhängigkeiten zwischen RENEW-Plugins (Benutzt-Perspektive der Infrastruktur)

3.2 Infrastruktur

Die meisten der hier definierten Agententypen stellen Teile der Workflow-Plattform dar und implementieren deren Dienste (vergleiche Abbildung 4 mit folgendem Text).

WFMS-Agent Der *Workflow-Management*-Agent kapselt das WFMS auf einem Rechner. Er bietet nach außen die Funktionalität eines lokalen WFMS an. Dieser Agent funktioniert als Plattform für alle anderen Agenten des Workflow-Agenten-Plugins.

WF-Cl-Agent Der *Workflow-Client*-Agent stellt die Verbindung zwischen Benutzern und dem System dar, unter Kenntnis der Benutzer, Gruppen, Rollen und Rechte für das WFMS, indem er mit dem Administrations-Agent kommuniziert. Er stellt die aktuellen Tasks je nach Rolle für den Benutzer dar.

Monitoring-Agent Dieser Agent sammelt Informationen über den Zustand der laufenden Workflow-Anwendung, die von den anderen Agenten explizit zur Verfügung gestellt werden. Er kann diese Daten zusammenfassen und im Sonderfall autonom (gemäß einem definierten Vorgehen, also wieder Workflow-gesteuert) agieren.

Task-Agent Diese Agenten entsprechen den vom WFMS aufgerufenen externen Anwendungen im Modell der WfMC. Sie bieten aus dem Anwendungskontext heraus ihre Dienste an und werden beauftragt, wenn ein Workflow bei der Ausführung dies vorsieht.

Verteilungs-Agent Dieser Agent ist für die Konfiguration der Workflowfragment-Agenten (WFF-Agenten) und der Workflow-Agenten (WF-Agenten) zuständig. Wenn ein Benutzer mit dem Workflow-Definitionswerkzeug von RENEW einen Workflow definiert, wird dieser über den WF-Cl-Agent (also entsprechend der Rechte des Benutzers), mit Hilfe des Verteilungsagenten in Form von WF- und WFF-Agenten in das System eingegliedert.

WF-Definition Dieser Agent kapselt die Schnittstelle zum Workflow-Definitionswerkzeug von RENEW, so dass ein instanziierter Workflow nur von autorisierten Benutzern (oder Agenten) verändert werden kann.

Administration Mit dem *Administrations*-Agenten können Rollen, Rechte und Benutzer verwaltet werden. Der WF-Cl-Agent wendet sich an den Administrations-Agenten, um einen Benutzer anzumelden und um Operationen gemäß dessen Rechten bereitzustellen (z.B. um neue Benutzer, Rollen oder Workflows ins System zu integrieren).

Remote-Agent Dieser Agent stellt die Kommunikation zu anderen WFMS-Agenten im lokalen System bereit, so dass ein verteilt laufendes WFMS realisiert ist. Zusammen mit dem Administration-Agent realisiert er auch sichere Kommunikationskanäle zwischen verteilten WFE-Agenten.

WFES-Agent Der *Workflow Enactment Service*-Agent bildet eine Agentenplattform, welche die WFE-Agenten in sich laufen hat. Dieser Agent kann von einem ankommenden WF-Agenten Informationen erhalten, aufgrund derer er die passende Workflow-Engine auswählt oder instanziiert.

WFE-Agent Eine *Workflow Engine* führt einen gegebenen Workflow-Agenten oder Workflow-Fragment-Agenten aus. Im Falle eines gewöhnlichen Workflows kann der WFE-Agent für einen aktuellen Task entscheiden, welcher Task-Agent geeignet ist, diesen Task auszuführen. Im Falle eines komplexeren, autonomen WF-Agenten interagiert der WFE-Agent mit dem WF-Agent.

WF- und WFF-Agenten Die Workflows selbst existieren als *Workflow-Agenten* und *Workflow-Fragment-Agenten* auf der WFES-Plattform oder in einer WFE-Plattform. Ein WF-Agent koordiniert die WFF-Agenten, welche lokale oder entfernte Teile des Workflows enthalten. Dabei sollen verteilte

Konflikte innerhalb eines Unternehmens gelöst werden, einerseits, weil in einem eng verbundenen Intranet die Synchronisationsmöglichkeiten und damit die Konfliktlösungsmöglichkeiten besser sind, andererseits weil dies konzeptionell erwünscht ist.

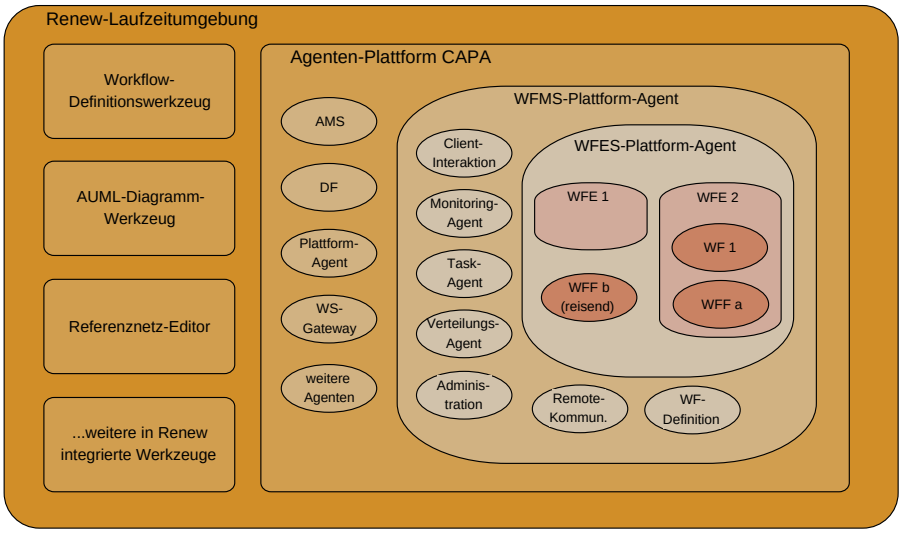

Abbildung 4: Infrastruktur in der Enthalten-Perspektive für einen Rechner

Abbildung 5 zeigt eine beispielhafte Infrastruktur einer Workflow-Agenten-Plattform. Die Ebenen, aus denen sie besteht, sind: Die Kommunikationsschicht über HTTP, die von CAPA zur Verfügung gestellt wird. Darüber liegen die Plattform-Agenten von CAPA, welche die grundlegenden Dienste einer FIPA-kompatiblen Plattform bereitstellen (Der *Directory Facilitator* (DF) und das *Agent Management System* (AMS)), sowie der Plattform-Agent von CAPA selbst. Darüber ist der WFMS-Agent dargestellt, der in Abbildung 4 differenziert wird. Über ihre Remote-Schnittstelle bilden die WFMS-Agenten innerhalb eines Intranets ein verteiltes WFMS. Auf diesem kann nun ein weiterer WFMS-Agent (mit allen enthaltenen Agenten) gestartet werden, der mit seiner Remote-Schnittstelle Verbindung zu einem anderen Intranet aufnehmen kann. Zusammen bilden sie dann ein Multilevel-WFMS, auf dem eine Anwendung aufsetzen kann (für die Anwendung zeigt die Abbildung nicht die auf konkreten Rechnern laufenden Teile).

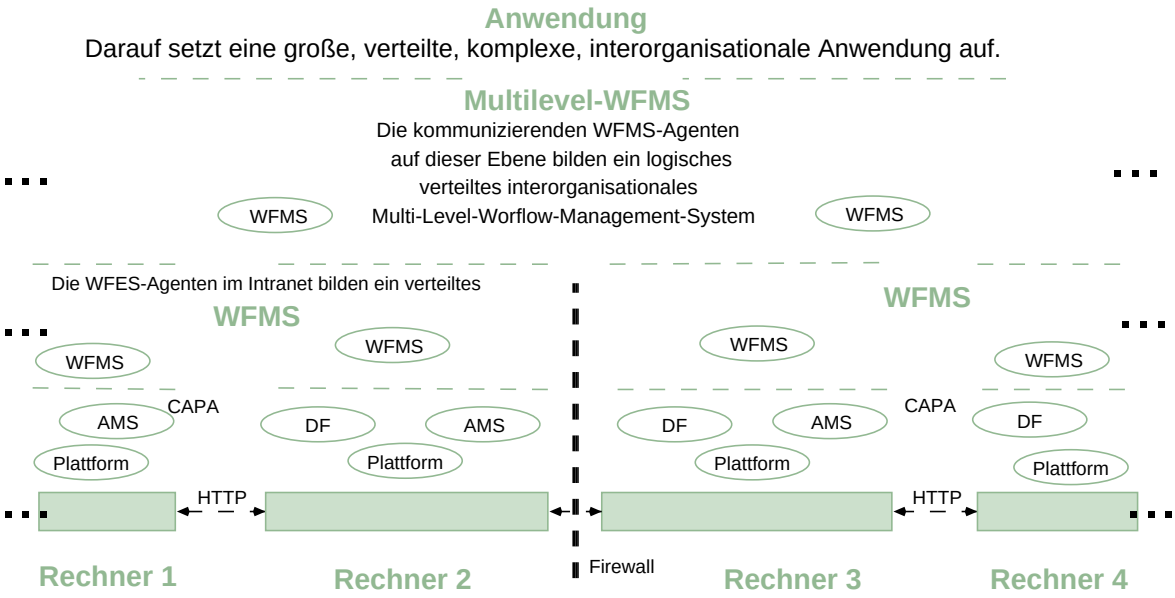

Abbildung 5: Beispielhafte Infrastruktur mehrerer Workflow-Agentenplattformen. Zum inneren Aufbau des WFMS-Agenten siehe Abbildung 4.

3.3 Erweiterungen des Prototyps

3.3.1 Verteilung und Ausführung

Fragmente von Workflows werden in WFF-Agenten gekapselt und verteilt. Die Fragmente werden von einem WF-Agent über einen Verzeichnisdienst aufgefunden. Die WFF-Agenten bieten ihre Dienste jeweils nur dem zugehörigen WF-Agenten an. Um die gegenseitige Erkennung sicherzustellen, werden die WF- und WFF-Agenten registriert und erhalten eine Signatur.

Ein WF-Agent sucht nach Anbietern für die notwendigen Fragmente, dabei kann mehr als ein Anbieter gefunden werden. Die WFF-Agenten werden erst dann aktiviert, wenn sie „an der Reihe sind“. Solange ein Fragment nicht aktiviert ist, kann zwischen alternativen Dienstanbietern gewählt werden.

Die Fragmente synchronisieren sich an ihren Randstellen über ein Steuerungsnetz. Dies wird durch eine verteilte Sperre realisiert, also ein Mechanismus, der einen konsistenten Zustand für gemeinsam genutzte Ressourcen sicherstellt. Nur der Inhaber der verteilten Sperre darf Änderungen vornehmen, und diese stellt sicher, dass es stets nur einen Inhaber zur Zeit geben kann.

Die Koordinierung und Konfliktlösung geschieht innerhalb eines Intranets über das Steuerungsnetz, in interorganisationalen Workflows sollen zur Vereinfachung keine verteilten Konflikte vorkommen. Die resultierende Topologie ist

sternförmig für je einen WF-Agent, die Fragmente kennen sich also nicht untereinander. Weitere Einzelheiten sind in [Car04] beschrieben.

3.3.2 Redundanz und Lastverteilung

Eine Architektur nach der FIPA ist nützlich zur Realisierung von Redundanz, indem mehrere Agenten einen Dienst anbieten. Sie können auf verschiedenen Agentenplattformen existieren. Der Mechanismus, nach dem von mehreren Dienstanbietern einer ausgewählt wird, realisiert dabei den gewünschten Effekt, wie z.B. Lastverteilung. Die Agenten, die einen Dienst aufrufen, müssen dabei flexibel implementiert sein, d.h. sie müssen autonom nach alternativen Dienstanbietern suchen. Falls auch die Plattform-Agenten auf den verschiedenen Ebenen mehrfach vorhanden sein sollen, müssen sie ihren Zustand sorgfältig synchronisieren.

3.3.3 Verzeichnisdienst

Das vorgeschlagene System braucht einen Verzeichnisdienst, um die Komponenten im schwach gekoppelten System aufzufinden und sie zu verkuppeln. Die Einträge eines solchen Verzeichnisdienstes brauchen eine Gültigkeitsdauer und sollten den Dienstanbieter und den angebotenen Dienst global eindeutig beschreiben. Die Einträge sollten über Plattformgrenzen hinweg aufgefunden werden können und sie müssen unbedingt zuverlässig sein: Registrierung und Manipulation von Einträgen dürfen nur authorisiert durchgeführt werden, die Dienstbeschreibungen müssen eindeutig sein und die Beziehung zum Dienstanbieter muss zuverlässig sein. Der FIPA Directory Facilitator (DF) erfüllt diese Kriterien zum guten Teil. Die fehlenden Sicherheitsfragen werden in diesem Beitrag nicht betrachtet. Eine andere Möglichkeit ist der Entwurf eines speziellen Verzeichnisdienstes, der z.B. von dem WFMS-Agenten bereitgestellt wird. In beiden Fällen müssen alle beteiligten Agenten die bereitgestellten Sicherheitsmerkmale auch nutzen.

Mit dem Vernetzungs-Plugin von Capa, Ace (siehe [RDK+03]), können Agenten ihre Dienste weltweit veröffentlichen und suchen, z.B. indem ein Netz von Agenten wie Agentcities oder openNet [opn05] verwendet wird.

4 Schlussbemerkung

In diesem Beitrag wurde die in [ROO+05] vorgeschlagene Architektur erweitert und konkretisiert.

Insgesamt entsteht eine konzeptionelle Architektur für komplexe Anwendungen, die all die in Abschnitt 2.6 erwähnten Eigenschaften, wie Steuerung, Monitoring, Autonomie, Kapselung, Flexibilität, Verifikation und graphische Modellierung, auf geordnete Weise vereint. Im Anwendungskontext zusammenhängende Sachverhalte werden auf zusammenhängende Systemkomponenten abgebildet, die durch lokale Änderungen an neue Anforderungen angepasst werden können.

Dadurch, dass Workflows nicht als Datenstrukturen, sondern als Agenten im System enthalten sind, ermöglicht diese Architektur auch komplexeres Verhalten als von „normalen“ Workflows bekannt ist.

Indem die Systemkomponenten über die Kommunikationsschicht von Capa miteinander verbunden sind, entsteht ein lose gekoppeltes System. Dies hat den Vorteil, dass einzelne Komponenten aktualisiert und neu gestartet werden können, ohne das gesamte System zu betreffen.

Da für diese Architektur Referenznetze, also höhere Petrinetze genutzt werden, steht eine präzise Modellierungstechnik für Workflows zur Verfügung. Die Workflows werden innerhalb eines Unternehmens beliebig verteilt; über Unternehmensgrenzen hinweg werden sie dergestalt verteilt, dass keine verteilten Konflikte mehr auftreten. Auf diese Weise können die Mitarbeiter eines Unternehmens untereinander kooperieren und die Unternehmen können auf einer Ebene darüber ebenfalls kooperieren, und zwar auf einem präzisen Prozessmodell aufbauend. Der Nachteil eines höheren Kommunikationsaufwandes liegt in der Natur einer mächtigeren Infrastruktur, wie sie für verteilte Workflows notwendig ist.

Unsere Hauptmotivation für diese verteilte Workflow-Engine liegt in der Unterstützung von kooperativer Arbeit, die durch verteilte Workflows unterstützt wird (wie in [LM04] formuliert). Andere Ansätze sind für Lastverteilung entworfen, wie z.B. in [BRD03], indem Workflows entsprechend der aktuellen Last auf passende Server zur Ausführung verteilt werden.

Es existieren zwar bereits petrinetzbasierte Workflow-Systeme (wie z.B. von [PSP04]), aber nach unserem Wissensstand ist dies der einzige Vorschlag, der die bereits realisierten komplexen Systeme auf eine durchgängige Architektur mit Petrinetzen abbildet, damit ein formales Fundament für solche Systeme liefert und außerdem ein lauffähiges System bietet.

Ausblick

Es gibt verschiedene Möglichkeiten, diese Arbeit weiterzuführen. Dabei stehen die Implementierung verschiedener, oben angeführter Erweiterungen des Prototyps und die Einbettung in einen Anwendungskontext im Vordergrund. Die zukünftig weiterentwickelten Teile werden auf eine verteilte Software-Entwicklungsumgebung hinzielen. Wir werden Renew, Mulan, Capa, Ace und das Workflow-Agenten-Plugin integrieren, um eine CWFE (Collaborative Workflow Engine) zu entwickeln. Die Konzepte aus der Agententechnologie werden von Bedeutung sein, da Flexibilität, Offenheit, Autonomie und Mobilität zunehmend zentrale Eigenschaften der heute entwickelten Systeme sind. Weiterhin werden Sicherheitsaspekte einen wichtigen Teil der Erweiterung des Systems darstellen, da diese gerade im Bereich interorganisationaler Workflows einen wichtigen Aspekt für die Akzeptanz eines WFMS darstellen.

Literatur

[Aal97] Wil van der Aalst. Verification of Workflow Nets. In Pierre Azéma und Gianfranco Balbo, Hrsg., *Application and Theory of Petri Nets 1997*, number 1248 in LNCS, Seiten 407–426, Berlin, 1997. Springer.

[AHKB03] Wil van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski und A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, Juli 2003.

[AMVW99] Wil van der Aalst, Daniel Moldt, Rüdiger Valk und Frank Wienberg. Enacting Interorganizational Workflows Using Nets in Nets. In *Proceedings der Workflow Management-Konferenz 1999*, Jgg. 70, Seiten 117–136. Universität Münster, 1999.

[BRD03] Thomas Bauer, Manfred Reichert und Peter Dadam. Intra-Subnet Load Balancing in Distributed Workflow Management Systems. *Int. J. Cooperative Inf. Syst.*, 12(3):295–324, 2003.

[Car04] Timo Carl. Entwicklung eines agentenbasierten verteilten Workflow-Management-Systems mit Referenznetzen. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 2004.

[CH92] Søren Christensen und Niels Damgaard Hansen. Coloured Petri Nets Extended with Channels for Synchronous communication. Bericht DAIMI PB–390, Computer Science Department, Aarhus University, April 1992.

[CMR03] Lawrence Cabac, Daniel Moldt und Heiko Rölke. A Proposal for Structuring Petri Net-Based Agent Interaction Protocols. In Wil van der Aalst und Eike Best, Hrsg., *24nd ICATPN 2003, Eindhoven, NL*, Jgg. 2679, Seiten 102 – 120, Berlin, 2003. Springer.

[DMR03] Michael Duvigneau, Daniel Moldt und Heiko Rölke. Concurrent Architecture for a Multi-agent Platform. In Fausto Giunchiglia, James Odell und Gerhard Weiß, Hrsg., *AOSE 2002, Revised Papers and Invited Contributions*, Jgg. 2585 of *LNCS*, Berlin, 2003. Springer.

[Jac02] Thomas Jacob. Implementierung einer sicheren und rollenbasierten Workflowmanagement-Komponente für ein Petrinetzwerkzeug. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 2002.

[Jen92] Kurt Jensen. *Coloured Petri Nets: Volume 1; Basic Concepts, Analysis Methods and Practical Use.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1992.

[JKM01] Thomas Jacob, Olaf Kummer und Daniel Moldt. Persistent Petri Net Execution. *Petri Net Newsletter*, 61:18–26, Oktober 2001.

[KMR03] Michael Köhler, Daniel Moldt und Heiko Rölke. Modelling mobility and mobile agents using nets within nets. In Wil van der Aalst und Eike Best, Hrsg., *24th ICATPN*, Jgg. 2679 of *LNCS*, Seiten 121–139. Springer, 2003.

[Kum01] Olaf Kummer. Introduction to Petri Nets and Reference Nets. *Sozionik Aktuell*, 1:1–9, 2001. ISSN 1617-2477.

[Kum02] Olaf Kummer. *Referenznetze*. Logos, Berlin, 2002.

[LM04] Kolja Lehmann und Vanessa Markwardt. Proposal of an Agent-based System for Distributed Software Development. In Daniel Moldt, Hrsg., *Proc. of MOCA 2004*, Seiten 65–70, Aarhus, Dänemark, Oktober 2004.

[MOO05] Daniel Moldt, Sven Offermann und Jan Ortmann. A Petri Net-Based Architekture for Web Services. In Lawrence Cavedon, Ryszard Kowalczyk, Zakaria Maamar, David Martin und Ingo Müller, Hrsg., *Workshop on Service-Oriented Computing and Agent-Based Engineering, SOCABE 2005, Utrecht, Niederlande, July 26, 2005. Proceedings*, Seiten 33–40, 2005.

[opn05] OpenNet Project. `http://www.x-opennet.org/`, 2005.

[PSP04] Maryam Purvis, Bastin Tony Roy Savarimuthu und Martin K. Purvis. Evaluation of a Multi-agent Based Workflow Management System Modeled Using Coloured Petri Nets. In Mike Barley und Nikola K. Kasabov, Hrsg., *PRIMA*, Jgg. 3371 of *LNCS*, Seiten 206–216. Springer, 2004.

[RDK+03] Christine Reese, Michael Duvigneau, Michael Köhler, Daniel Moldt und Heiko Rölke. Agent–based Settler Game. In *Agentcities Agent Technology Competition, Barcelona, Spain*, Februar 2003.

[Ren05] Renew – The Reference Net Workshop homepage. URL `http://www.renew.de/`, 2005.

[ROO+05] Christine Reese, Jan Ortmann, Sven Offermann, Daniel Moldt, Kolja Lehmann und Timo Carl. Architecture for Distributed Agent-Based Workflows. In Brian Henderson-Sellers und Michael Winikoff, Hrsg., *Workshop on Agent-Oriented Information Systems, AOIS 2005, Utrecht, Netherlands, July 26, 2005. Proceedings*, Seiten 42–49, Utrecht, Niederlande, Juli 2005. Verfügbar unter http://www.aois.org/2005/aamas-proceedings.pdf.

[Val98] Rüdiger Valk. Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In Jörg Desel, Hrsg., *19th ICATPN*, number 1420 in LNCS, Berlin, 1998. Springer.

[Wor05] Workflow Management Coalition. WfMC Workflow Reference Model. URL `http://www.wfmc.org/standards/model.htm`, 2005.