Architekturelle Unterstützung von Electronic Commerce Anwendungen

Ludger Fiege
fiege@rbg.informatik.tu-darmstadt.de
Graduiertenkolleg "Infrastruktur für den elektronischen Markt"
TU Darmstadt

Zusammenfassung

Die IT-Infrastruktur muß im Bereich der elektronischen Märkte die Kernkompetenzen effizient unterstützen und in einem dynamischen Umfeld immer wieder neu in verschiedenen Konfigurationen anbieten. Die vorgestellten abstrakten Kooperationsmodelle erlauben Invarianten zu identifizieren und fördern die Entwicklung einer ereignis-basierten Kommunikations-Infrastruktur für EC-Anwendungen. Diese darf nicht auf eine Abstraktionsebene beschränkt sein und muß anwendungsunabhängig verschiedene QoS-Parameter in komplexen Systemen umsetzen können.

1 Electronic Commerce

Electronic Commerce (EC) ist einer der am schnellsten wachsenden Industriezweige. Sowohl das Volumen der umgesetzten Güter als auch die Ausgaben für die Infrastruktur übersteigen bereits andere große Industriezweige wie die Gesundheits- oder die Automobilindustrie. Allgemein wird erwartet, daß diese Entwicklung bedeutende Auswirkungen auf alle Bereiche des Geschäftslebens haben wird.

Die IT-Einsatz wird, wie heute schon ersichtlich ist, die Struktur und die Organisationsformen von Firmen und Industriezweigen verändern [5]. Die Kosten von Firmenkooperationen verringern sich durch die computergestützte Zusammenarbeit und folglich können immer mehr Bereiche ausgelagert werden. Firmen können sich auf Kernkompetenzen konzentrieren und komplexe Produkte und Dienstleistungen werden nur noch in Verbünden von kooperierenden Partnern erstellt. Aus dieser Entwicklung ergeben sich zahlreiche neuen Szenarien. Das Wort der „Coopetition" beschreibt die Zusammenarbeit konkurrierender Firmen, die ein gemeinsames Ziel erreichen wollen [1].

Kurzfristige Kooperationen sind ein charakterisierendes Merkmal von virtuellen Unternehmen. In einem *Virtual Enterprise* (VE) sind verschiedene, autonom arbeitende und weltweit verteilte Unternehmen kurzfristig vereint, um eine konkrete Geschäftsmöglichkeit auszunutzen. Einzelne Unternehmen können gleichzeitig an mehreren virtuellen Unternehmen beteiligt sein und zu späteren Zeitpunkten mit ehemaligen Partnern in unterschiedlichen VEs konkurrieren. Die Dienstleistungen sind auf Kernkompetenzen reduziert und sie unterliegen einem ständigem Wandel, um sie optimal in verschiedene VEs einzubringen. IT-Standards und die dadurch reduzierten Kooperationskosten erlauben zunehmend auch kurzfristig Kooperationen aufzubauen und aufzulösen.

Diese Eigenschaften charakterisieren ein Extrem in der Entwicklung von Organisationsformen, aber einzelne Aspekte davon sind in allen EC-Anwendungen wiederzufinden. Vom Gesichtspunkt der technischen Infrastruktur können VEs zunächst auf kooperierende (IT-)Dienste (Services) bzw. die Dienstintegration reduziert werden. Die bereitgestellten Dienste müssen die oben genannten Eigenschaften von VEs unterstützen. Die IT-Infrastruktur muß also verstärkt mit sehr dynamischen Konfigurationen umgehen können; diese Anforderung wird durch die mangelnden Einflußmöglichkeiten auf die kooperierenden Partner noch verschärft.

Der Erfolg von (virtuellen) Unternehmen wird daher maßgeblich von der Möglichkeit abhängen die IT-Services effizient anpassen und integrieren zu können. Die wiederholte (Dis-, Re-)Integration von Diensten ist die wichtigste Anforderung an die IT-Infrastruktur. Die zukünftige Entwicklung muß darauf konzentriert sein, die IT-Dienste sauber zu trennen und die Abhängigkeiten zwischen ihnen zu reduzieren.

2 Kooperationsmodelle

Wie die obige Diskussion belegt, wird im Bereich elektronischer Märkte die Architektur von Software Systemen, d.h. die Unterteilung in Subsysteme und die Art ihrer Zusammenarbeit, zunehmend wichtig. Die Anwendungen sind von inhärent vernetzter Natur und unterliegen ständigen Veränderungen. Der

Aufwand für Rekonfigurationen kann nur klein gehalten werden, wenn wesentliche Teile unverändert bleiben. Die Auswahl der "richtigen" Architektur ist sicherlich ein entscheidendes Kriterium.

Architekturelle Entscheidungen werden häufig von Fragen der Implementierungstechniken beeinflußt oder sogar bestimmt, ohne dem eigentlichen Kooperationmodell die nötige Beachtung zu schenken. Im folgenden werden vier abstrakte Kooperationsmodelle unterschieden, die unabhängig von Implementierungstechniken und -randbedingungen Kooperationen charakterisieren können. Sie repräsentieren invariante architekturelle Eigenschaften.

Ein Kooperationsmodell [6, 3] beschreibt die Interaktionsmuster die ein Dienst unterstützt. Es ist also eine charakteristische Eigenschaft des Dienstes und früh im Design festzulegen. Die Modelle werden danach unterschieden, welcher der kooperierenden Dienste aus Sicht des Initiators von Interesse ist und ob die einzelnen Adressaten den Teilnehmern bekannt sind. Das erste Merkmal, *Service of Interest* (SoI), beschreibt, ob Leistungen/Daten des Initiators (SoI = Initiator) einer Aktion oder eines anderen Dienstes (SoI = Responder) von Interesse sind. Im ersten Fall übermittelt der aktive Dienst (Initiator) den kooperierenden Partnerdiensten Informationen über sich selbst. Im zweiten Fall erfragt der aktive Dienst eine Leistung von anderen Diensten. Das zweite Merkmal unterscheidet, ob die Adressaten dem Initiator bekannt sind (*Addressee* = Specific) oder ob sie nur der zugrundeliegenden Kooperationsplattform bekannt sind (*Addressee* = Anonymous).

Das *Request/Reply* Kooperationsmodell (SoI=Responder, Addressee=Specific) ist aus dem Einsatz in Client/Server Umgebungen und als Aufruf in prozeduralen Programmiersprachen bekannt (auch zum Beispiel RPC und CORBA). Beim *Anonymous Request/Reply* ist der Adressat dem Initiator (Client) nicht bekannt und die Übermittlung eines Requests an den/die tatsächlichen Server obliegt der Kooperationsplattform. Im *Callback* Modell sind die Adressaten dem Initiator, d.h. dem zurückrufenden Dienst, bekannt und es werden nur Informationen über seinen Status übermittelt. Im *ereignis-basierten* Modell ist schließlich der Adressat wiederum unbekannt. In den beiden letztgenannten Modellen (SoI = Initiator) sind die Initiatoren nicht von den passiven Diensten (Empfängern) abhängig und sind daher nur lose gekoppelt.

Die Kooperation zwischen einem Paar oder in einer Menge von Diensten wird über die abstrakten Interaktionsmuster hinaus durch Quality of Service (QoS) Parameter weiter bestimmt [6]. Aspekte der Synchronisation, Fehlertoleranz, Transaktionsfähigkeit, Sicherheit usw. können hier einfließen. Eine Kooperationsplattform stellt die Primitive zur Verfügung, die ein Kooperationsmodell mit den entsprechenden QoS-Parametern realisieren.

Ähnliche Unterscheidungen von Kooperationen findet man im Bereich der Design Patterns [2] und des Software Engineerings (Architectural Styles [7]). Die Unterscheidungsmerkmale konnten hier aber explizit und unabhängig von einer bestimmten Abstraktionsebene oder Funktionalität angegeben werden. Darüberhinaus bildet die Definition nur einen Ausgangspunkt für weitere Untersuchungen: Der Entwurf dieser Plattformen und Primitiven ist von zentraler Bedeutung in verteilten Systemen, weil die architekturelle Unterstützung die Struktur der darüberliegenden Dienste und deren Änderungsmöglichkeiten beeinflußt. Welche Leistung können die Primitive bieten?

Heute werden allzuoft nur Techniken bereitgestellt mit denen verschiedene Kooperationsmodelle realisierbar sind. Ursprüngliche Entwurfsentscheidungen hinsichtlich der Kooperation sind nach einigen Änderungen nicht mehr sichtbar. Auf das Kooperationsmodell bezogene Primitive bewahren diese Entscheidungen und vereinfachen nachträgliche Änderungen. Dies gilt vor allem auch, wenn sich die darunterliegende Technik verändert.

Das Request/Reply Modell wurde ausführlich untersucht und es existiert mit CORBA ein umfangreicher und akzeptierter Standard. Callback und Anonymous Request/Reply wurden als Kooperation bisher nicht ausführlich betrachtet; es existieren hierfür höchstens Techniken (Message Queues u.ä.). Die ereignis-basierte Kooperation ist nicht neu, aber komplexe Systeme werden nur unzureichend unterstützt.

3 Ereignis-basierte Systeme

Lose gekoppelte Systeme sind in zweifacher Hinsicht von immer größer werdender Bedeutung [3, 6]. Zunächst beschränken sie die gegenseitigen Abhängigkeiten und eignen sich somit besonders für dynamische Umgebungen, in denen Rekonfigurationen häufig vorkommen — wie zum Beispiel EC-Anwendungen. In einer Umgebung, in der es um Dienstintegration geht, sollten sich die Dienste möglichst nur auf die eigene Funktionalität und die eigenen Daten beschränken. Andernfalls ist ein

Einsatz außerhalb ihres aktuellen Kontextes ohne weiteres nicht möglich. Eine ereignis-basierte Kooperation kommt diesen Anforderungen entgegen. Das zweite Argument zugunsten von lose gekoppelten Systemen liefert die zunehmende Automatisierung. Im Electronic Commerce sind Informationsflüsse von größerer Bedeutung als die alleinige Abfrage von Informationen. Der Konkurrenzdruck fördert personalisierte und kundenbezogene Dienstleistungen, die dem Dienstnutzer automatisch "geeignete" Informationen anbieten.

Ein Automatismus erzeugt die Kundenorientierung und reagiert auf Statusänderungen, die im allgemeinen Fall mit Hilfe einer ereignis-basierten Kooperation verbreitet werden. Da jeder computergestütze Automatismus durch eine Interaktion mit der physikalischen Welt ausgelöst wird (Zeit, Sensoren, Tastatur,...), besitzt jedes Computersystem nach außen ereignis-basierte Schnittstellen. An einem Punkt innerhalb dieses Systems wird bisher typischerweise die ereignis-basierte Kooperation auf Request/Reply abgebildet.

In statischen Systemen sind die Übergänge zwischen ereignis-basierter und Request/Reply Kooperation einfach zu identifizieren und bleiben unverändert. Bei der Dis- und Reintegration von Diensten stellt sich aber die Frage nach der geeigneten Größe der Dienste, d.h. auf welcher Abstraktionsebene liegen die Übergänge. Diese Frage wird sich bei jeder Rekonfiguration erneut stellen und ist nicht eindeutig zu beantworten, wenn ein Unternehmen IT-Dienste in mehrere VEs einbringt. Daneben erfordern nebenläufige Aktionen mit zunehmender Skalierung zwangsläufig auch wieder ereignis-basierte Kooperation; Polling ist an dieser Stelle eine schlecht skalierbare Implementierungstechnik [4].

Software Systeme zum Aufbau virtueller Unternehmen, wie zum Beispiel die Produkte von I2 und Manugistics, belegen den Trend zu lose gekoppelten Systemen. Sie nutzen Schnittstellen einer Message Oriented Middleware (MOM) um die Funktionalität von Systemteilen zu kombinieren; zusammen mit existierenden MOM-Produkten eine Lösung für eher statische Konfigurationen.

4 Ausblick

Die bei der Realisierung von virtuellen Unternehmen auftretenden Probleme sind in vielen Electronic Commerce Anwendungen vorhanden. Die wiederkehrende (Re-)Konfiguration von Diensten mit wechselnden Anforderungen ist eine zentrale Aufgabe der IT-Infrastruktur. Um den Aufwand hierfür klein zu halten, muß die grundlegende Architektur klar definiert sein und unverändert bleiben. Die vorgestellten Kooperationsmodelle erlauben zusammen mit diversen QoS-Parametern eine anwendungs- und technikunabhängige Untersuchung der möglichen architekturellen Unterstützung.

Dynamische, komplexe Systeme profitieren von der losen Kopplung ereignis-basierter Kooperation. Die bisherigen Lösungen erlauben keine effiziente Umsetzung umfangreicher Systeme, weil die angebotenen Funktionalitäten das ereignis-basierte Kooperationsmodell meist nur unvollständig unterstützen und immer nur in einer Abstraktionsebene betrachten. Von entscheidender Bedeutung ist daher ein System, daß die möglichen (umfangreichen, aber auch adäquaten) Funktionalitäten, wie sie etwa durch die QoS-Parameter Transaktionfähigkeit, Sicherheit, Fehlertoleranz beschrieben werden, effizient auf verschiedenen Abstraktionsebenen realisiert. Die Administration und Rekonfiguration zur Laufzeit, sowie das Messen realer Systeme sind zusätzliche Anforderungen die für einen praktischen Einsatz zu unterstützen sind.

Die Themen die wir aktuell weiter verfolgen umfassen den Begriff der *Aktion* in ereignis-basierten Systemen, sowohl im Hinblick auf transaktionelle Eigenschaften als auch im Hinblick auf die Sichtbarkeit von Ereignissen und die Ausdehnung von Aktionen. Letzteres ist besonders wichtig um die Administrierbarkeit sicherzustellen. Die Festlegung der Sichtbarkeitsbereiche berührt Fragen der Sicherheit und der Abrechnung, die gerade auch bei Dienstleistungen in virtuellen Unternehmen sehr wesentlich sind.

Literatur

[1] Adam M. Brandenburger and Barry J. Nalebuff. *Coopetition.* Doubleday, 1996.

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons, 1996.

[3] L. Fiege, G. Mühl, and A. Buchmann. An architectural framework for electronic commerce applications. submitted for publication, 2000.

[4] Michael J. Franklin and Stanley B. Zdonik. "Data In Your Face": Push Technology in Perspective. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 516–519. ACM Press, 1998.

[5] Thomas W. Malone and Robert J. Laubacher. The dawn of the e-lance economy. *Harvard Business Review*, pages 145–152, September 1998.

[6] G. Mühl, L. Fiege, and A. Buchmann. Realizing advanced electronic commerce applications. In *Information Systems for E-Commerce*, 2000.

[7] Elisabetta Di Nitto and David Rosenblum. Exploiting ADLs to specify architectural styles induced by middleware infrastructures. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 13–22. IEEE Computer Society Press / ACM Press, 1999.