Der CORBA Notification Service als Basis für die Vermittlung von Gruppenbewußtsein

Thomas Paradies
tran*SIT* GmbH
Langewiesener Str. 32
98693 Ilmenau
paradies@transit-online.de

Zusammenfassung

Die natürliche Vermittlung von Gruppenbewußtsein oder Mutual Awareness im Rahmen von CSCW-Applikationen ist eine sehr komplexe und vielschichtige Aufgabe, der man softwaretechnisch häufig durch die Verwendung von eventbasierten Kommunikationsmechanismen begegnet. Dem Gruppenbewußtsein liegt die Transparenz von Nutzeraktivitäten zugrunde. Aus Gründen der Akzeptanz muß die Transparenz rollenbasiert hinsichtlich der Kriterien Offenheit und Interesse durch den Nutzer kontrollierbar sein. Praktische Arbeiten mit einem prototypisch in Java implementierten Notification Service zeigen, daß dieser Dienst für die Entwicklung von CORBA-basierten CSCW-Anwendungen zur Vermittlung von Gruppenbewußtsein sinnvoll zum Einsatz gebracht werden kann.

1 Vermittlung von Gruppenbewußtsein durch Software

Eine Spezifik von CSCW-Applikationen gegenüber traditionellen Softwareapplikationen ist das Vorhandensein einer mehrfachen Mensch-Maschine-Schnittstelle. Zur Abgrenzung von einfachen Mensch-Maschine-Schnittstellen in Multi-User-Systemen, wie z.B. bei Datenbankanwendungen sehr verbreitet, wird die mehrfache Mensch-Maschine-Schnittstelle im CSCW-Sprachgebrauch gern zu einer Mensch-Maschine-Mensch-Schnittstelle qualifiziert.

Die Mensch-Maschine-Mensch-Schnittstelle ist erforderlich, um eine dem Arbeitskontext angemessene Zusammenarbeit zwischen Mitgliedern eines Arbeitsteams zu ermöglichen. Ein wichtiger Aspekt ist dabei die gegenseitige Transparenz der Aktivitäten eines einzelnen am System für alle Teammitglieder in einer Art und Weise, wie sie auch in Arbeitsumgebungen ohne Computerunterstützung der Zusammenarbeit ausgeprägt ist.

Unabhängig vom Grad der Computerunterstützung ist die Gegenseitigkeit der Aktivitätentransparenz ein entscheidender Faktor für die Ausprägung des in der Literatur vieldiskutierten Gruppenbewußtseins (*engl*. awareness, mutual awareness). Aus der Forderung nach einer möglichst natürlichen Unterstützung leitet sich die Unabdingbarkeit der Transparenzkontrolle durch den Nutzer ab.

Aus der Spezifik der Nutzerschnittstelle muß die Steuerbarkeit der Transparenz sowohl in der Rolle des Nutzers als Initiator bzw. Ausführer oder als Beobachter von Aktivitäten gegeben sein. Der Grad der Transparenz beschreibt in der ersten Rolle die Offenheit und in der zweiten Rolle das Interesse.

Mit dieser knappen Darstellung einiger Wesenszüge des Gruppenbewußtseins soll verdeutlicht werden, welche Anforderungen an CSCW-Applikationen zu stellen sind, um dem Anspruch der möglichst natürlichen Vermittlung von Gruppenbewußtsein gerecht zu werden. Softwaretechnische Umsetzungen beschreiben Aktivitäten typischerweise über Ereignisse, was die Verwendung von eventbasierten Kommunikationsmechanismen nahelegt.

Entsprechend der Vielschichtigkeit und Komplexität der Problemstellung gibt es bereits eine enorme Anzahl von mehr oder weniger speziellen Softwarelösungen ohne erkennbare

Wiederverwendungsansätze. Hinsichtlich der Architektur von CSCW-Applikationen ist klar, daß sie bedingt durch die Aufgabenstellung immer verteilte Softwaresysteme sind und potentiell auch aufsetzend auf objektorientierten Middlewareplattformen wie CORBA implementiert werden können.

Nimmt man an, daß Awareness-Unterstützung, so wie oben geschildert, auf der Basis von event-orientierten Kommunikationsmechanismen auf relativ niedrigem Niveau möglich ist, stellt sich die Frage nach der Eignung der von der OMG spezifizierten Dienste für das Handling der Bewußtseinsvermittlung.

2 CORBA Notification Service

In CORBA-basierten Applikationen beruht die Kommunikation zwischen Objekten typischerweise auf Events, was in großen Systemen einen beträchtlichen Eventverkehr zur Folge haben kann. Eine gezielte Selektion bestimmter Events aus dem Eventstrom über den von der OMG definierten Event Service, der auf einem einfachen Level über Channeldefinitionen eine eventbasierte Kommunikation zwischen verteilten Objekten realisiert, ist nur über Filterkriterien und korrespondierende Filtermechanismen zu erreichen. Der Allgemeingültigkeit des Problems und dem dafür naheliegenden Lösungsansatz folgend (siehe [1],[2],[3]), initiierte die OMG mit einem RFP (*Request For Proposal*) [4] die Diskussion um die Spezifikation des Notification Services, die aktuell noch nicht abgeschlossen ist.

Der Notification Service ist eine Erweiterung des Event Services [5], die neben dem erweiterten Filtermechanismus die Einstellung von Filterkriterien – zur Anmeldung des Interesses an bestimmten Ereignissen – sowie das Zustellen von filterbaren Events umfaßt. Damit obliegt die Aufgabe der Eventfilterung nicht mehr den einzelnen Event Consumern, sondern kann einem zentralen Dienst überantwortet werden. Die Consumer haben dem Notification Service über ein geeignetes Interface lediglich das Interesse an bestimmten Events mitzuteilen. Für die Anmeldung unterstützt der Event Service sogenannte Subscriptions.

In Analogie zum Event Service sind Quellen von Notifikationen Notification Producers bzw. auch Notification Supplier, Senken werden als Notification Consumers bezeichnet. Beide über sind über Notification Channels (die wiederum Spezialisierungen von Event Channels sind) voneinander entkoppelt und sollten wie beim Event Service gegenüber dem Notification Service sowohl im Push- als auch im Pull-Modus arbeiten können. Zur deutlicheren Abgrenzung zum Event Service werden die durch den Notification Service behandelten Events als Notifications bezeichnet.

Typische Applikationsfelder, für die der Notification Service potentiell vorgesehen ist, sind die Administration und das Monitoring großer verteilter Systeme, die dem Anwender bei genauerer Betrachtung auch ein Bewußtsein vermitteln – allerdings von System- und nicht von Nutzeraktivitäten. Grundsätzlich ist der Dienst durch die benutzerdefinierbare Notifikationen für die Anwendung in beliebigen Domänen offen.

Der Einsatz des Dienstes zur Awareness-Vermittlung im Sinne von CSCW - wie oben beschreiben - ist allerdings nur eingeschränkt möglich, da lediglich der Aspekt der Beobachterrolle eine geeignete Umsetzung vorfindet. Durch die Verlagerung der Filterfunktionalität von den Konsumenten hin zum Dienst kann das Interesse der Konsumenten an bestimmten Events über Subscriptions gesteuert werden.

Der Aspekt der Initiatorrolle, über die die Offenheit gesteuert wird, findet in der Spezifikation des Dienstes keine klare Umsetzung. Damit ist diese Aufgabe von den Produzenten wahrzunehmen. Prinzipiell eröffnet die Kopplung über einen Event Channel die Varianten „offen“ oder „nicht offen“ eine Lösung, die zwar umsetzbar ist, aber eigentlich nicht über das hinausgeht, was mit dem Event Service an Funktionalität zur Verfügung steht.

3 Prototypische Implementierung eines Notification Service

Da für die geplanten Untersuchungen keine Implementierung der Notification Service Spezifikation verfügbar war, wurde auf der Grundlage des Notification Service RFP eine prototypische Eigenimplementierung vorgenommen. Diese Implementierung setzt auf dem VisiBroker for Java sowie auf dem Event Service der Firme Inprise (ehemals Visigenic) auf. Da sowohl der Object Request Broker als auch die Services in Java implementiert sind, wurde Java als Implementierungssprache gewählt.

Die Interfaces des Notification Service sind in Anlehnung an die Struktur des Event Services in zwei IDL-Modulen zusammengefaßt:

- *CosNotifyChannelAdmin*
  - *ProxyNotifyPushSupplier*
  - *ProxyNotifyPushConsumer*
  - *NotifyConsumerAdmin*
  - *NotifySupplierAdmin*
  - *NotificationChannel*
- *CosNotifyComm*
  - *EventInfo*
  - *FilterExpr*
  - *Subscription*
  - *SubscriptionAdmin*
  - *NotifyPushConsumer*
  - *NotifyPushSupplier*

Das Modul *CosNotifyChannelAdmin* beinhaltet alle Interfaces, die in Verbindung mit dem Notification Channel stehen. Das Modul *CosNotifyComm* stellt Interfaces für Supplier- und Consumer-Objekte zur Verfügung.

Notifikationen gehen immer von Supplier-Objekten (den Event-Quellen) aus und werden an das assozierte Consumer-Proxy-Objekt gesendet. Vom Consumer-Proxy-Objekt wird die Notifikation an den Channel weitergeleitet, der sie wiederum an Proxy-Supplier-Objekte weitergibt, die die Filterung des EventInfo-Objekts veranlassen. Wird die Notifikation vom Filter bestätigt, wird sie an das Consumer-Objekt durchgereicht.

Die Implementierung des Notification Services ist im Sinne des Notification Service RFPs zwar nicht vollständig, reflektiert aber die grundsätzlich vorgesehenen Eigenschaften bzw. Funktionalität des Dienstes und wird damit den Ansprüchen des Themas gerecht. Grundsätzlich akzeptiert der Dienst Notifications von Producern/Suppliern und leitet diese an Consumer weiter, wobei beide Parteien vollständig voneinander entkoppelt sind.

Die Skalierbarkeit des Notification Channels durch die Kopplung mehrerer Channels und die Verteilung von Notification Servern ist derzeit nicht gegeben. Wie aus der obigen Aufstellung zu ersehen ist, wird das Pull-Modell nicht unterstützt. Unberücksichtig blieben auch getypte Notifications sowie das dynamische Hinzufügen von benutzerdefinierten Typen.

Da die endgültige Spezifikation des Notification Service noch nicht verabschiedet wurde, fällt die Unvollständigkeit der Implementierung in keiner Form ins Gewicht. Für die Zukunft ist die Implementierung des Pull-Modells interessant, da dann Misch-Modelle zur Anwendung kommen können die in Abhängigkeit von den Eventquellen gegenüber den Producer-Objecten vorteilhaft zum Einsatz gebracht werden können.

4 Ergebnisse

Anhand einiger ausgewählter Applikationsbeispiele (simulierte EMail-Notifikation, Text-Messaging-System) konnte mit dem prototypisch entwickelten Notification Service auf einem sehr einfachen Level die Tragfähigkeit des Ansatzes – allerdings bei vergleichsweise geringem Verkehr - gezeigt werden.

Inwiefern der Ansatz auch zur Lösung anspruchsvollerer Aufgaben, z.B. im Rahmen eines Intranets mit vielen unterschiedlichen Eventquellen, zur Anwendung kommen kann, werden weitere Untersuchungen ergeben. Wünschenswert ist die Erschließung weiterer Eventquellen mit einem Mehrwert für den Anwender.

Literatur

[1] K. S. DiBella, N. Bhandaru. Pilgrim Event Notifier Version 1.0. Techn. Report, University of Massachusetts, Nov. 1992

[2] ITU-T. X.721, X.733 und X.734. X.700 series, International Telecommunication Union

[3] J. Gough, G. Smith. Efficient recognition of events in a distributed system. Proc. 18th Australian Computer Science Conference, 1994

[4] Object Management Group. Notification Service Request For Proposal. RFP, 03. Juni 1997, OMG Document telecom/97-01-03

[5] Object Management Group. Event Service Specification. *CORBAservices: Common Object Services Specification*, Chap. 4, Nov. 1997, OMG