Dynamische Generierung von Protokollen zur Steuerung automatisierter Verhandlungen

M.T. Tu, C. Langmann, F. Griffel, W. Lamersdorf *
Arbeitsgruppe Verteilte Systeme, Fachbereich Informatik, Universität Hamburg
Vogt–Kölln–Str. 30, D–22527 Hamburg
[tu|2langman|griffel|lamersd]@informatik.uni-hamburg.de

Zusammenfassung

In elektronischen Handelssystemen (E-Commerce) bekommt die Unterstützung der *Verhandlungsphase* von Geschäftstransaktionen eine wachsende Bedeutung. Jedoch sind die meisten existierenden Systeme mit einem bestimten Protokoll fest verbunden und daher eingeschränkt in ihrer *Offenheit* bzgl. der jeweiligen Anforderungen an eine Verhandlung. Um dem entgegenzuwirken, wird in diesem Beitrag eine Petrinetz-basierte Sprache zur Spezifikation von *Verhandlungsprotokollen* vorgestellt, welche dynamisch für jeweils eine einzelne Verhandlung generiert und aktiviert werden können. Solche Protokolle und das entsprechende Verhandlungssystem sind darüber hinaus dafür konzipiert, vollständig automatisierte Verhandlungsprozeße mittels Anbindung von Softwareagenten zu steuern.

Schlüsselwörter: Elektronischer Handel, Kommunikation und Kooperation, Verteilte Systeme

1 Einleitung

In den letzten Jahren hat der Bereich E-Commerce (elektronischer Handel) immer mehr an Bedeutung gewonnen, und der Trend scheint sich noch zu verstärken. Kennzeichnend für die bisherige Entwicklung des E-Commerce ist, daß sich sowohl die Informationsgewinnung als auch die Bestellung und Bezahlung ins Internet verlagert. Selbst die Lieferung von geeigneten Produkten, wie etwa Lizenzen, Multimediadokumenten und Software erfolgt zunehmend online. Einen wichtigen Teil sowohl des elektronischen wie auch des herkömmlichen Handels nimmt das *Verhandeln* ein. Dieser Bereich umfaßt einfache Preisabsprachen und das Aushandeln von Rabatten und reicht bis hin zu komplexen Verhandlungen mit einer großen Anzahl von Verhandlungspartnern über eine Vielzahl von Verhandlungszielen (siehe [1]). Deshalb gibt es z.Z. verstärkt Bemühungen von einer Reihe von Online-Systemen und Firmen, das Verhandeln zu ermöglichen bzw. zu unterstützen. Dies kann am besten durch die rasche Verbreitung von elektronischen Handelssystemen, die allgemein unter dem Begriff *Auktionssystemen* zusammengefaßt werden, beobachtet werden, wie etwa Onsale [3], Lufthansa [4] und Priceline [5], um nur einige zu nennen. Allerdings ist all den existierenden Systemen gemeinsam, daß sie ein festes Protokoll vorgeben, das die Regeln der Verhandlung festlegt – in den meisten Fällen ist es in der Tat eines der Auktionsprotokolle (siehe [2]) oder ein noch einfacheres (Such-)Verfahren wie im Falle von Priceline. Damit sind solche Systeme ungeeignet für alle Handelstransaktionen, die eine andere Art von Verhandlung und damit ein anderes Verhandlungsprotokoll voraussetzen.

Um derartigen Einschränkungen entgegenzuwirken, wird in diesem Papier ein generisches System zur Steuerung elektronischer Verhandlungen vorgestellt, das dafür konzipiert ist, möglichst viele und insbesondere *beliebig spezifizierbare* Verhandlungsprotokolle zu unterstützen. Dieses im Rahmen des DynamiCS[1]-Projektes

*Diese Arbeit ist gefördert von der Deutschen Forschungsgemeinschaft (DFG) im Rahmen des Projektes La 1061/1-2

[1] *Dynami*cally *C*onfigurable *S*oftware

an der Universität Hamburg entwickelte Verhandlungssystem ist darüber hinaus darauf ausgelegt, nicht nur den herkömmlichen Verhandlungsprozeß mit menschlichen Teilnehmern elektronisch zu unterstützen, sondern diesen auch mittels Anbindung von (mobilen) Softwareagenten, die jeweils eine bestimmte Verhandlungsstrategie implementieren, weitestgehend zu automatisieren. Um die Zielsetzung und den Anwendungskontext der in diesem Papier behandelten Protokolle zu beleuchten, wird daher im folgenden zunächst in Abschnitt 2 die Architektur des DynamiCS-Verhandlungssystems umrissen. In Abschnitt 3 werden die konkreten Anforderungen an eine generische Spezifikation von Verhandlungsprotokollen erörtert. Es folgt dann in Abschnitt 4 eine Darstellung von OOPAMELA, einer Petrinetz-basierten Sprache zur Beschreibung von Protokollen, die dynamisch von einer entsprechenden Steuerungseinheit interpretiert werden können. In Abschnitt 5 wird ein Tool zur Unterstützung des Prozeßes der Generierung von OOPAMELA-konformen Protokollen präsentiert. Im letzten Abschnitt 6 wird ein Ausblick auf weitere Aktivitäten im Zusammenhang mit dem vorgestellten Verhandlungssystem gegeben.

2 Die DynamiCS Architektur zur Unterstützung verhandlungsfähiger Agenten

In dem DynamiCS-Projekt wird eine Umgebung entwickelt, in der intelligente, mobile Agenten miteinander verhandeln können. Dafür sind zwei Grundlagen zu legen, nämlich eine Umgebung für Verhandlungsstrategien und eine für die Umsetzung verschiedener Verhandlungen durch die Bearbeitung von Verhandlungsprotokollen. Dabei wird insbesondere Wert darauf gelegt, sowohl das Zustandekommen (Vermittlung der Teilnehmer und Initialisierung) einer Verhandlung als auch den Verhandlungsprozeß selbst möglichst *dynamisch* und somit flexibel zu gestalten. Zum einen sollen deshalb alle Komponenten, die einen mobilen Agenten in die Lage versetzen, an einer Verhandlung teilzunehmen, zur Laufzeit von ihm erst erworben oder ausgetauscht werden können. Diese Eigenschaft wird bereitgestellt durch sogenannte *Plug-in*-Mechanismen, die in [6] beschrieben werden. Zum anderen soll auch das Protokoll, das jeweils von den Teilnehmern erwünscht ist, dynamisch für eine bestimmte Verhandlung generiert und aktiviert werden können. In Abbildung 1 wird gezeigt, wie die einzelnen Bestandteile der DynamiCS-Architektur zusammenarbeiten.

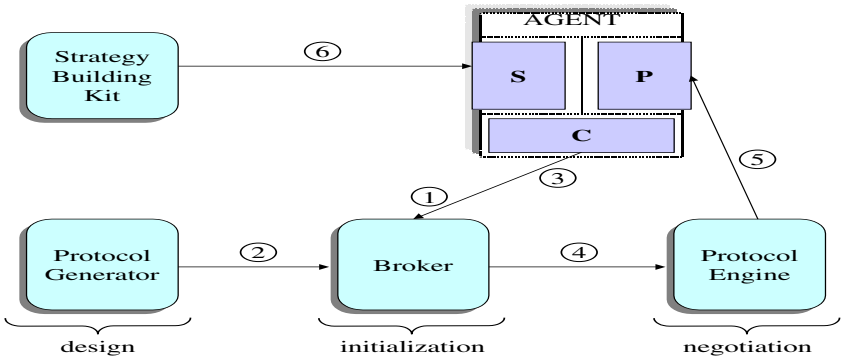

Abbildung 1: Struktur verhandlungsfähiger Agenten mit zugehörigen Unterstützungskomponenten

Demnach besteht ein verhandlungsfähiger Agent aus drei Hauptkomponenten, die dynamisch aggregiert werden können, nämlich einem *Kommunikationsmodul*

(C), einem *Protokollmodul* (P) und einem *Strategiemodul* (S). Das C-Modul ist zuständig für das Senden und Empfangen von Verhandlungsnachrichten in einer be-stimmten Kommunikationssprache für Agenten. Das P-Modul sorgt dafür, daß der Agent konform zu einem Verhandlungsprotokoll handelt, d.h. jede ein- und ausgehende Verhandlungsnachricht wird von diesem Modul auf Protokollkonformität geprüft. (Das P-Modul kann entweder eine eigenständige Komponente oder ein *Frontend* zu einer (logisch) zentralen *Protokoll-Engine* sein.) Das S-Modul realisiert eine konkrete Strategie, die für die Erzeugung von Verhandlungsaktionen und Angeboten zuständig ist. (Siehe [6] für eine ausführliche Beschreibung der Funktionalität der Module.)

Das Grundablaufschema zur Initialisierung einer Verhandlung mit Agenten erfolgt in folgenden Schritten: Ein Agent meldet sich beim *Broker*, um die Verhandlung zu einem gewissen Gegenstand zu initiieren (1). Dazu muß er ein *Protokoll-Template* auswählen – das zuvor mit dem *Protokoll-Generator* (siehe Abschnitt 5) erstellt wurde und bei diesem abgelegt ist – und sich für eine bestimmte *Rolle* im Template registrieren, z.B. die eines Auktionators (2). Weitere Agenten registrieren sich als Interessenten für jeweils eine der (noch besetzbaren) Rollen des Templates, z.B. für die Rolle eines Auktionsteilnehmers (3). Wenn die Startbedingungen für die Verhandlung, beispielsweise wenn die Mindestanzahl von Teilnehmern für eine Auktion erreicht ist, erfüllt sind, werden alle Beteiligten informiert und das ausgefüllte Template an die *Protokoll-Engine* übergeben (4). Die Protokoll-Engine erzeugt für jeden Verhandlungsagenten ein entsprechendes Protokoll-Modul, das dynamisch in das Agentengerüst eingebunden wird (5). Es wird dann ein zu dem Protokoll-Modul passendes Strategie-Modul, das aus einem Strategie-Baukasten (*Strategy Building Kit*) ausgewählt werden kann, hinzugefügt und die Verhandlung kann nun beginnen (6).

3 Anforderungen an Verhandlungsprotokolle

An Verhandlungsprotokolle können eine große Zahl von Anforderungen gestellt werden. Das liegt daran, daß Verhandlungen in unterschiedlichsten Situationen unter verschiedenen Bedingungen stattfinden und das ausführende System mittels Verhandlungsprotokolle an die unterschiedlichen Situationen angepaßt wird. Beispielsweise können Verhandlungen zwischen kooperativen Teilnehmern stattfinden, die das gleiche Ziel anstreben (wie bspw. Aufgabenverteilung) oder zwischen Verhandlungspartnern, die gegenteilige Ziele erreichen wollen. Bei dem Entwurf eines Verhandlungsprotokolls ist zu beachten, daß es wesentlichen Einfluß auf die Taktik bzw. die Strategie der Teilnehmer ausüben kann (siehe hierzu die Diskussion in [8]).

Teilnehmerbezogen: Eine der Anforderungen besteht darin, die Teilnehmerzahl zu nach oben oder unten beschränken, bzw. neue Teilnehmer nur zu bestimmten Zeitpunkten zuzulassen. Zusätzlich muß eingeschränkt werden können, ob sich die Teilnehmer gegenseitig kennen. Zusammengefaßt kann es also verschiedene Arten von Zulassungsbeschränkungen geben. Dies ist aus Gründen der Ressourcenknappheit oder Aufgrund von Teilnehmerforderungen sinnvoll.

Heterogen: Verhandlungsprotokolle müssen plattform- und sprachunabhängig sowie ortstransparent beschrieben werden können. Die Verhandlungsteilnehmer dürfen vom Verhandlungsprotokoll aus in ihrer Implementation weder auf eine bestimmte Hardware, ein bestimmtes Betriebssystem oder eine Implementationssprache festgelegt werden. Da vor mobile Agenten als Teilnehmer zugelassen werden sollen, müssen Verhandlungen auch Ortstransparent beschrieben und ausgeführt werden. Als Beispiel dienen Teilnehmer, die über Handy und Modem von einer Segelyacht aus an einer Verhandlung in München teilnehmen, oder einen mobilen Agenten während der Verhandlung zu den jeweiligen Verhandlungspartnern schicken.

Aktionsbegrenzung: Jeder Verhandlungspartner darf nur eine bestimmte Anzahl bestimmter Aktionen durchführen. Als eine Aktion wird jede Handlung im Rahmen einer Verhandlung bezeichnet, wie z.B. Angebot unterbreiten oder ablehnen. Dies kann notwendig sein, um einen schnellen Vertragsabschluß zu sichern oder um unerwünschte Verhandlungsstrategien zu unterbinden. Zusätzlich sollen Reihenfolgen für die Teilnehmer festgelegt werden können, um rundenbasierte Verhandlungen zu ermöglichen.

Validierung: Jeder eingereichte Vertrag muß hinsichtlich verschiedener Kriterien validiert werden können. Dies kann "gedächtnislos" oder unter Einbeziehung der Historie geschehen. Mögliche Kritereien sind:

- Hinzufügen neuer Vertragsinhalte
- Entfernen alter Vertragsinhalte
- Einschränkung von Werten (Intervalle, Ober-/Untergrenzen, Auswahl) von Vertragsinhalten
- Einschränkung von möglichen Relationen zwischen Attributen eines Angebotes (z.B.: wenn Autofarbe rot, dann Sitzfarbe schwarz oder rot) oder verschiedener Angebote (bei der sog. *open-cry* Auktion (siehe z.B. [2]) muß jedes Angebot höher als das letzte sein)

Verständlichkeit und Überprüfbarkeit: Obwohl die Beschreibungssprache in erster Linie dem Zweck dient, Verhandlungsprotokolle formal zu beschreiben, um Verhandlungen ausführen und kontrollieren zu können, ist es wichtig, daß Menschen die Protokolle erstellen und auch im nachhinein analysieren und verstehen können. Zwar können diese Vorgänge durch Werkzeuge unterstützt werden, es ist aber zur Bildung des Vertrauens in die gesamte Verhandlungsumgebung wichtig, daß ein Benutzer das Protokoll, welches benutzt wird, vollständig und möglichst ohne Filter betrachten und analysieren kann. Ein Verhandlungsprotokoll sollte formal auf Schwachstellen überprüft werden können.

Zusätzlich zu diesen formalen Forderungen müssen Verhandlungsprotokolle realen Verhandlungssituationen entsprechen. Nur so kann eine breite Akzeptanz eines Verhandlungssystems erreicht werden, da potentielle Benutzer von Verhandlungen bestimmte Vorstellungen haben. Zusätzlich müssen bekannte Verhandlungstypen mittels Verhandlungsprotokollen beschrieben werden können. Daher werden im folgenden Aspekte realer Verhandlungen aufgegriffen, die mittels Verhandlungsprotokollen nachgebaut werden können sollen.

Rollen: Werden die Teilnehmer von Verhandlungen betrachtet, so kristallisieren sich bestimmte funktionale Rollen heraus. Es ist möglich, daß reale Verhandlungsteilnehmer die Aufgaben mehrerer Rollen übernehmen, aber auch, daß mehrere Verhandlungsteilnehmer die gleiche funktionale Rolle innehaben. Rollen beschreiben die Funktion verschiedener Teilnehmer in einer Verhandlung. Die in vielen Verhandlungen vorhandenen Rollen lassen sich in drei Gruppen aufteilen:

- Die erste Gruppe von Rollen hat die Funktion, übliche Verhandlungsteilnehmer zu beschreiben. Diese Rollen werden von den Agenten der jeweiligen Anbieter, Interessenten, etc. ausgefüllt.
- Eine weitere Gruppe von Rollen bilden Notare, Validierer, Protokollanten, usw. Solche Kontrollrollen haben eher die Eigenschaft von Komponenten mit bestimmter Funktionalität und beeinflussen nicht das Verhandlungsprotokoll in seiner deskriptiven Struktur.

- Eine dritte Gruppe von Rollen sind die Vermittlungsrollen. Diese sind dafür zuständig, festgefahrene Verhandlungen auf verschiedene Arten zu einem erneuten Anlaufen bzw. zu einem Abschluß zu bringen. Wie bei einer Kontrollkomponente ist es bei vielen nötig, unabhängig von den Verhandlungsteilnehmern zu sein.

Vertragsakzeptanz: Unter Vertragsakzeptanz soll festgelegt werden, wie eine Einigung zustande kommt. Für jedes Akzeptanzverfahren sind bestimmte Rollen (s.o.) nötig. In Verhandlungsprotokollen sollen verschiedene Akzeptanzverfahren verwendet werden können. Die verschiedenen Möglichkeiten sind kombinierbar. Wird beispielsweise nicht innerhalb einer bestimmten Frist auf normale Weise ("Bestätigung") eine Einigung erzielt, kann ein anderes Verfahren eingesetzt werden ("Abstimmung über Auswahl"). Dies entspricht dem "Final-Offer"-Verfahren. Als Akzeptanzverfahren sollen genannt werden:

- Bestätigung: Jeder Teilnehmer stimmt einem Entwurf zu.
- Auswahl: aus einer Menge von Entwürfen wird einer von einer zuständigen Instanz ausgewählt.
- Vermittlung: Aus mehreren Entwürfen wird ein neuer generiert.
- Wahl: Auf mehrere Entwürfe verteilen die Teilnehmer ihre Stimmen.

4 OOPAMELA: Eine Petrinetz-basierte Sprache zur Spezifikation von Verhandlungsprotokollen

Um der dargestellten Vielfalt an Anforderungen an Verhandlungsprotokolle gerecht zu werden, wurde für deren Spezifikation im DynamiCS-Projekt OOPAMELA entwickelt – eine Sprache, die den Aufbau von gefärbten High-Level-Petrinetzen (siehe [9]) beschreibt. Das Petrinetz beschreibt dabei die statische Struktur einer Verhandlung. Die dynamische Struktur ergibt sich durch das Schalten von Transitionen. Zusätzlich kann sich das Netz zur Laufzeit im Rahmen der festgelegten Struktur durch Hinzunahme oder Wegfall von Teilnehmern, die durch entsprechende Rollen modelliert werden, verändern. Das Schalten des Petrinetzes wird als ein Prozeß angesehen. Daher wird die statische Struktur als Prozeßbeschreibung bezeichnet.

Eine *Rolle* besitzt ein Interface und wird bei der Instanziierung des Protokolls durch ein oder mehrere Objekte besetzt. Von jeder Transition, an die eine Rolle angeschlossen ist, existieren zur Laufzeit genauso viele Exemplare, wie Objekte die Rolle besetzen. Andere Transitionen oder Stellen können zusätzlich mit der Rolle verknüpft werden. Auch diese Netzelemente werden entsprechend vervielfacht. Man kann daher von der Transition- und Stellenbeschreibung auch als Transitions- und Stellen*typen* sprechen.

Die *multiple* Instanziierung von Transitionen und Stellen bezogen auf die zugehörigen Rollenexemplare erfordert eine zusätzliche Verwaltungsbeschreibung. Diese ist das *Metaprotokoll*. Im Metaprotokoll wird bspw. beschrieben, wie viele Teilnehmer es von jeder Rolle geben darf und ob sich die Teilnehmer kennen. Die Zusammenfassung der Prozeßbeschreibung und der Metabeschreibung wird in OOPAMELA Protokoll genannt. In OOPAMELA werden alle Protokollkomponenten als CORBA-Objekte angesehen und dementsprechend über ein IDL-Interface (siehe [10]) angesprochen. Dazu zählen die Marken, Stellen, Transition und Rollen. Alle diese Komponenten sind Objekte mit Identitäten und Eigenschaften.

Jedes Protokoll ist systematisch aus mehreren Teilen zusammengesetzt. Eine graphische Darstellung ist in Abbildung 2 zu sehen. Der erste Teil ist eine Beschrei-

Abbildung 2: Struktureller Aufbau von OOPAMELA-Protokollen

bung des Protokolls. Der zweite beschreibt die eigentliche Protokolldefinition. Die Protokolldefinition besteht selbst wieder aus drei Bestandteilen:

- Einer Interfacedefinition, in der alle benutzen Interfaces importiert oder deklariert werden.
- Einer Prozeßdefinition, in der der strukturelle Aufbau eines Protokolls beschrieben wird. Die Prozeßdefinition enthält eine Liste von Stellendeklarationen sowie die Beschreibung der Transitionen.
- Einem Metaprotokoll, in dem Verwaltungs- und Instantiierungsinformationen zu dem beschriebenen Prozeß aufgezählt werden.

Jede *Stelle* ist typisiert. Sie kann nur Marken enthalten, die dem Typ der Stelle entsprechen (siehe auch [7]). Dabei ist zu beachten, daß für die Typen die aus der Objektorientierung bekannte Subtyppolymorphie gilt: Jeder Typ kann anstelle seiner Obertypen benutzt werden. Zusätzlich kann sich jede Stelle auf einen Rollennamen beziehen ("relates to"). Wenn sie das tut, werden von ihr zur Laufzeit so viele Exemplare erzeugt, wie es Objekte gibt, die die Rolle besetzen.

Eine *Transitionsbeschreibung* besteht aus einer Bezeichnung, einer Rollenzuordnung, der Angabe der ein- und ausgehenden Kanten sowie dem Aktionsteil. Die *Bezeichnung* jeder Transition muß eindeutig sein. Jede Transition kann wie eine Stelle an eine Rolle gebunden sein. Deren Interface steht der Transition im Aktionsteil zur Verfügung. Auch eine Transition wird bei der Netzerstellung so häufig erzeugt, wie es Exemplare der zugehörigen Rolle gibt.

Eine Transition kann an mehrere Rollen gebunden sein. Dabei ist die Einschränkung zu beachten, daß jede zusätzliche Rolle entweder gemeinsam für alle Transitionsexemplare besetzt wird oder einzeln für jede Transition. Die Anzahl der Transitionsexemplare wird durch die erste angegebene Rolle vorgegeben (Abbildung 3). Die Anbindung der Transition "transX" laut Struktur in Abbildung 3a wird beschrieben durch:

```
trans transX uses RolleA[M], RolleB[1]
```

Abbildung 3b zeigt die entsprechende Struktur, in der jedes Transitionsexemplar an jeweils ein eigenes Rollenexemplar der Rolle B angebunden ist. Die OOPAMELA-Beschreibung dafür lautet:

```
trans transX uses RolleA[M], RolleB[M]
```

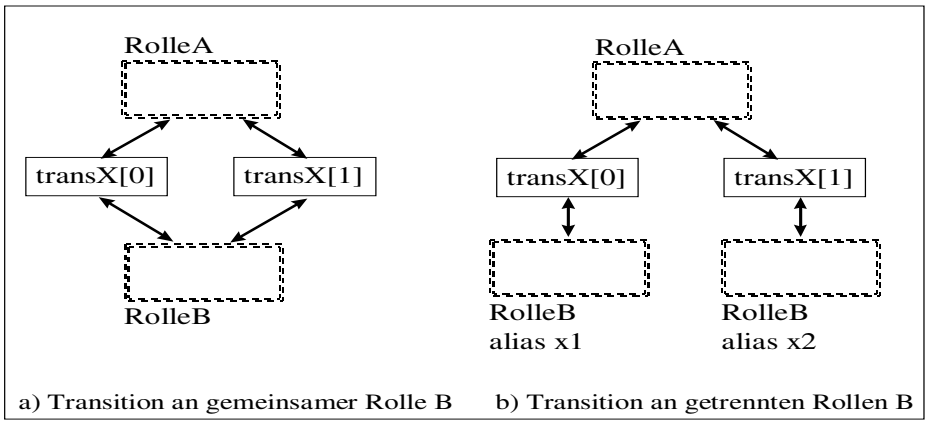

Abbildung 3: Anbindung einer Transition an mehrere Rollen

In diesem Fall wird, um die Eindeutigkeit der Rollennamen zu bewahren, intern automatisch ein Alias vergeben und den Rollenexemplaren zugeordnet.

Ein Netzelement (Stelle oder Transition) wird mit einem anderen Netzelement über die "get"- und "put"-Anweisung der beteiligten Transition verbunden. Besondere Betrachtung erfordert der Umstand, daß die in einem get/put-Statement angegebenen Eingangs- und Ausgangsstellen jeweils eine Menge von Stellen sein können. Das ist automatisch der Fall, wenn die Transition ohne, die Stelle aber mit Multiplizität angegeben ist.

Beispielhaft werden Eingangsstellen betrachtet: Es wird angenommen, daß eine an eine multiple Rolle gebundene Stelle die betrachtete Eingangsstelle für eine singuläre Transition ist (Abbildung 4). Die Anbindung geschieht über eine "get"-Anweisung der Transition. An diese kann ein "Guard" (Schutzfunktion) angehängt werden, so daß eine Transition in abhängigkeit der Markeneigenschaften in den Eingangsstellen schaltet. Ausgangsstellen werden durch entsprechende "put"-Anweisungen an Transitionen angebunden.

Abbildung 4: Stellenexemplare als Eingangsstellen

Nach den "get"- und "put"-Anweisungen folgt der Aktionsteil. Jede Aktion weist einer Variablen einen berechneten Ausdruck zu oder besteht aus einem Methodenaufruf. Ein Ausdruck besteht aus einem oder der Verknüpfung von mehreren Werten.

Auf den Aktionsteil der Transitionsbeschreibung folgt optional die *Timeout*-Beschreibung. Sie wird benutzt, um das Verhalten von Transitionen zu beschreiben,

die für das Schalten zu lange brauchen. Die Zeitmessung beginnt in dem Moment, in dem die Marken aus den Eingangsstellen entfernt werden. Für Verhandlungen ist das Timeout von Bedeutung, um zeitkritische Verhandlungsformen ausführen zu können oder ein Fehlverhalten bei Teilnehmern zu bemerken.

Bei einem Timeout werden bestimmte Timeoutmarken in die angegebenen Stellen gelegt. Wenn eine Transition eine Timeoutkante besitzt, so passiert es, daß entgegen der einfachen Petrinetzdefinition nicht alle ausgehenden Kanten verwendet werden. Dieses Verhalten wird als "nichtdeterministisches oder" bezeichnet, da aufgrund des Netzes nicht entschieden werden kann, welche Kante keine Verwendung findet (Abbildung 5a). Dieses Verhalten kann mit Hilfe von Verfeinerung auf einfache Petrinetze abgebildet werden (Abbildung 5b).

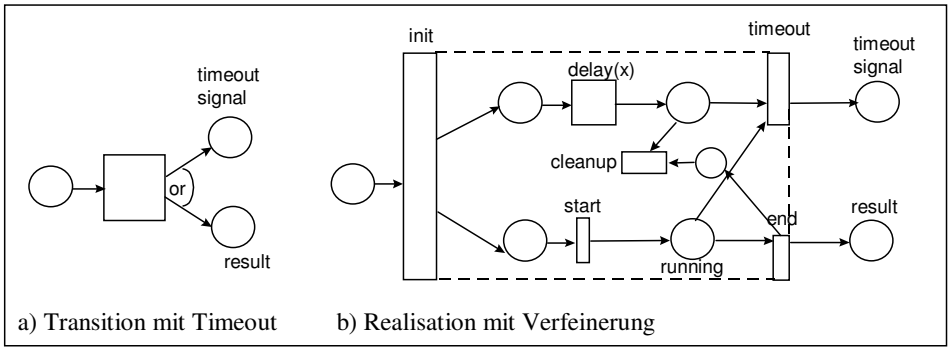

Abbildung 5: Realisation des Timeout-Konstrukts

In dem Metaprotokoll werden die Informationen abgelegt, die das Verhandlungssystem benötigt, um ein konkretes Exemplar des mit dem Verhandlungsprotokoll beschriebenen Verhandlungstyps erzeugen und verwalten zu können. Das Protokoll dient dem Zweck, eine komplexe Kommunikationsbeziehung zwischen Kommunikationspartnern zu beschreiben und im Endeffekt auch darin, ein System bei der Verwaltung des Kommunikationsprozesses zu unterstützen. Daher sind in dem Metaprotokoll hauptsächlich Informationen über die Zusammensetzung und das dynamische Verhalten der Kommunikationspartner festgelegt.

Ein Metaprotokoll besteht aus Listen von Einträgen. Diese Listen können dreierlei Art sein. Zunächst werden die globalen Einträge aufgezählt. Diese sind, solange keine speziellen Angaben in den später erläuterten Listen gemacht werden, für die gesamte Verhandlung und für alle Rollen gültig. Weitere Listen enthalten Informationen über Rollen und an Rollen angeschlossene Transitionen.

Für alle Metaprotokolleinträge gelten fünf Aussagen:

- Sie sind optional.
- Für jeden mehrfach in einer Liste vorkommenden Eintrag gilt der letzte.
- Jeder Eintrag ist einschränkend.
- Parameterbehaftete Einträge werden groß geschrieben. Eintragsnamen und Parameter werden wie in der Sprache Smalltalk durch Doppelpunkte getrennt.
- Parameterlose Einträge werden klein geschrieben und können genauer spezifiziert werden, indem ein Block in geschweiften Klammern folgt. Im allgemeinen können sie ersetzt werden durch parameterbehaftete Einträge mit dem Parametertyp "bool".

Diese Aussagen haben folgende Konsequenzen: Zunächst kann jeder Eintrag im Metaprotokoll gar nicht, einmal oder mehrfach vorkommen. Dadurch, daß jeder Eintrag eine einschränkende Eigenschaft besitzt, ist es möglich, in späteren Sprachversionen neue Eigenschaften hinzuzufügen, ohne daß alte Protokolle ein neues Verhalten bekommen.

5 Generierung von Verhandlungsprotokollen

Im folgenden wird die Konzeption und Funktionalität des im Rahmen des Dynami-CS-Verhandlungssystems entwickelten *Protokollgenerators* vorgestellt. Diese komplett in der Programmiersprache Java implementierte Komponente unterstützt sowohl die graphische Erstellung von OOPAMELA-konformen Protokollen als auch die Generierung von konkreten Protokoll-Instanzen aus vorgefertigten Schablonen (*Templates*) auf einer abstrakten Meta-Ebene, d.h. ohne daß der Benutzer genaue Kenntnisse von OOPAMELA sowie Petrinetzen im allgemeinen haben muß.

Verhandlungsprotokolle in OOPAMELA können auf verschiedenen Abstraktionsstufen betrachtet werden. Auf niedrigster Ebene befindet sich das in OOPAMELA beschriebene Protokoll in reiner Textform. Diese Ebene ist z.B. für Korrektheitsanalysen vonnöten und dient daher der Vertrauensgewinnung der Benutzer des Verhandlungssystems, auch wenn diese selbst den OOPAMELA-Code nicht lesen oder verstehen können. Aus dem OOPAMELA-Code läßt sich eine graphische Abbildung erzeugen, die vor allem die Struktur der Verhandlung verdeutlicht. Aus Übersichtsgründen können bestimmte Bestandteile, wie z.B. die Beschränkung der Teilnehmerzahl oder der Validierungsmechanismus ausgeblendet werden. Das Wissen über die Semantik von Petrinetzen ist jedoch immer noch Voraussetzung, um solche Abbildungen zu verstehen. Wird von dem zugrundeliegenden Petrinetz abstrahiert, so erhält man eine Abstraktion, in der die Semantik des Verhandlungsprotokolls in den Vordergrund rückt. Das zugrundeliegende Petrinetz darf nicht mehr Bestandteil des Verstehens auf dieser Ebene sein. Dies ist die Ebene, die für die Mehrheit der Benutzer eines Verhandlungssystems geeignet erscheint.

Viele Verhandlungsprotokolle sind sich in ihrer Grundstruktur ähnlich und unterscheiden sich in einzelnen Details häufig erst auf Metaprotokollebene. Diese Erkenntnis greift der Protokollgenerator auf, indem er die Gemeinsamkeiten verschiedener Verhandlungsprotokolle zu *Strukturprotokollen* zusammenfaßt und ein konkretes Protokoll durch Parametrisierung und Anpassung aus dem Strukturprotokoll gewinnt. Die möglichen Parameter hängen vom Strukturprotokoll ab und werden durch dessen Designer festgelegt. Ohne zusätzliche Angaben von diesem sind die Parameter, die Einträge des Metaprotokolls und die Beschränkung einzelner Transition bezüglich der Schaltvorgänge. Ansonsten wird jede Veränderung des Strukturprotokolls als Parametrisierung oder Anpassung bezeichnet.

Die Aufteilung in Struktur- und konkretes Protokoll entspricht der Aufteilung des Generators in zwei Module. Mit dem ersten wird das Strukturprotokoll in Form eines graphischen Petrinetzes erstellt und unter einem semantisch sinnvollen Namen gespeichert. Die Erstellung eines Auktionsprotokolls beispielsweise ist in Abbildung 6 illustriert. Zusätzlich wird eine Beschreibung erfaßt und gespeichert. Das Metaprotokoll findet keine Beachtung. Für jedes Strukturprotokoll kann ein spezieller *Wizard*, dem zweiten Modul, angegeben werden. Zunächst wird ein Strukturprotokoll anhand des Namens und einer zusätzlichen Beschreibung ausgewählt. Der Wizard stellt eine Oberfläche zur Verfügung, mit der der Benutzer aus dem Strukturprotokoll durch Angabe von Parametern ein konkretes vollständiges Verhandlungsprotokoll erzeugen lassen kann. Das resultierende Protokoll kann einerseits mit einer graphischen Petrinetzrepräsentation gespeichert werden, so daß es sich beispielsweise mit dem Design-Tool ansehen läßt, auf jeden Fall wird es aber

als OOPAMELA-Code im Textformat ausgegeben. Ein *Standardwizard* steht zur Verfügung und wird benutzt, wenn der Designer des Strukturprotokolls keinen anderen Wizard angibt. Da das Standardtool keine semantischen Informationen über das zu bearbeitende Protokoll besitzt, arbeitet es auf einer sehr niedrigen Abstraktionsebene und erfordert ein großes Maß an Systemwissen über OOPAMELA von dem Benutzer.

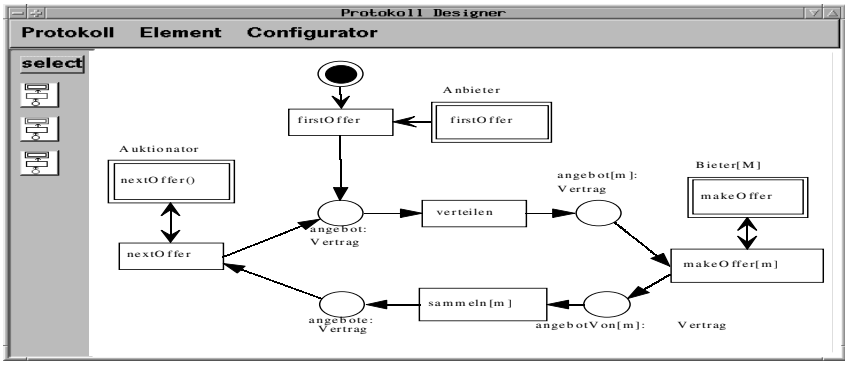

Abbildung 6: Graphische Erstellung eines Auktionsprotokolls

Der Designer kann zu dem von ihm erzeugten Strukturprotokoll einen Wizard angeben, der zum Konfigurieren und Erzeugen des konkreten Protokolls benutzt werden soll. Diesen Wizard muß er gegebenenfalls selbst erstellen. Dadurch ist es dem Designer möglich, das semantische Wissen über die von ihm erzeugte und beschriebene Klasse von Verhandlungsprotokollen in den Wizard aufzunehmen. Dies ermöglicht eine Oberflächengestaltung und Benutzerführung auf einem maximal hohen Abstraktionsniveau. Jeder Wizard wird von dem Wizard-Loader geladen. In dem Wizard-Loader werden alle verfügbaren Struktur-Protokolle angezeigt, und nach Auswahl eines von ihnen wird der zugehörige Wizard gestartet. Die Wizard-Architektur ist prinzipiell nicht festgelegt. Der Entwickler eines Wizards muß sich lediglich an die Anforderung halten, daß der Wizard mindestens das Interface `configurator.WizardInterface` implementiert (siehe Abbildung 7).

```
public interface WizardInterface {
          public String createOOPamelaCode();
          public void saveConcreteProtocol(ProtocolStorage aStorage);
          public void setStructureProtocol(StructureProtocol aProtocol);
          public void show();
};
```

Abbildung 7: Erfordertes Wizard-Interface

Sofort nachdem der Wizard geladen wurde, wird dessen Methode `setStructureProtocol` aufgerufen, um ihm sein Protokoll zu übergeben, welches er konfigurieren soll. Danach wird seine Methode `show` gerufen, um den Wizard anzeigen zu lassen. Die Methode `createOOPamelaCode` wird aufgerufen, um den Wizard aufzufordern, zu dem von ihm erstellten OOPAMELA-Netz den OOPAMELA-Code zu erzeugen. Dieser wird als ein langer String zurückgegeben. Die Methode `saveConcreteProtocol` wird benutzt, um das fertige Protokoll im ProtocolStorage abzulegen.

Wenn der Wizard das sichtbare Netz bearbeiten oder darstellen soll, muß das In-

terface `configurator.WizardOptionalInterface` implementiert werden. Die einzige Methode `setGraphicalProtocol` wird nach dem Start aufgerufen, nachdem `setStructureProtocol` aufgerufen wurde, und sie dient dazu, dem Wizard die graphische Darstellung des OOPAMELA-Netzes mitzuteilen. Da der Benutzer vom Wissen über OOPAMELA befreit sein sollte, empfiehlt es sich, dieses Interface nur zu Testzwecken zu implementieren.

6 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein generischer und dynamischer Ansatz zur Spezifikation und Generierung von Verhandlungsprotokollen dargestellt. Zunächst wurde als Anwendungskontext für die hier behandelten Protokolle das im DynamiCS-Projekt entwickelte Verhandlungssystem für intelligente, mobile Agenten vorgestellt. Nach einer Erörterung der Anforderungen an Verhandlungsprotokolle wurde dann OOPAMELA als eine Petrinetz-basierte Sprache zur deren Spezifikation dargestellt. Es folgte dann die Beschreibung des Protokollgenerators, eines Werkzeuges zur graphischen Erstellung von Protokoll-Schablonen sowie zur dynamischen Generierung von OOPAMELA-konformen Protokollen, die zur Steuerung automatisierter Verhandlungen eingesetzt werden können.

Wegen des Umfangs dieses Beitrags haben wir allerdings die Konzeption und Implementierung der entsprechenden Steuerungseinheit – die zur Laufzeit ein solches Protokoll interpretiert und die Verhandlungsteilnehmer entsprechend auf Protokollkonformität prüft – ausgelassen. Um bestimmte Verhandlungstypen vollständig automatisieren zu können, wird darüber hinaus z.Z. im DynamiCS-Projekt ein Framework zur Entwicklung von verteilten Strategiemodulen, die jeweils zu einem Protokolltyp konform sind, realisiert.

Literatur

[1] M. Merz, F. Griffel, T. Tu, S. Müller-Wilken, H. Weinreich, M. Boger, and W. Lamersdorf. Supporting Electronic Commerce Transactions with Contracting Services. In *International Journal on Cooperative Information Systems*, Vol. 7, No. 4. World Scientific Publishing, 1998.

[2] M. Kumar and S. Feldman. Business Negotiations on the Internet. IAC Reports, IBM Research Division, T.J. Watson Research Center, 1998. `www.ibm.com/iac/reports-technical/reports-bus-neginternet.html`.

[3] Onsale. Onsale atAuction, 1999. `www.onsale.com/atauction.htm`.

[4] Lufthansa. Lufthansa Live Auction, 1999. `virtualairport.lufthansa.com/deutsch/lh_auction/auction.htm`.

[5] Priceline.com, 1999. `tickets.priceline.com`.

[6] M.T. Tu, F. Griffel, M. Merz, and W. Lamersdorf. A Plug-in Architecture Providing Dynamic Negotiation Capabilities for Mobile Agents. In *Proceedings of the 2. Intl. Workshop on Mobile Agents (MA'98)*, Stuttgart. Springer-Verlag (LNCS), 1998.

[7] K. Müller–Jones, M. Merz, and W. Lamersdorf. Realisierung von Kooperationsanwendungen auf der Basis erweiterter Diensttypbeschreibungen. In *Entwicklung und Management verteilter Anwendungssysteme – Tagungsband des 2. Arbeitstreffens der GI/ITG Fachgruppe 'Kommunikation und verteilte Systeme' und der GI Fachgruppe 'Betriebssysteme'*. Springer-Verlag, Berlin, 1995.

[8] J. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiations among Computers.* MIT Press, 1994.

[9] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, EATCS Monographs on Theoretical Computer Science (3 Bände). Springer-Verlag, Berlin, 1997.

[10] Object Management Group. *CORBA 2.2/IIOP Specification.* OMG Dokument formal/98-07-01, 1998.