Ein bruchloser Übergang von der Prozeßmodellierung zu kooperativen Software-Architekturen

Patrick Hupe Florian Matthes Holm Wegner

Arbeitsbereich Softwaresysteme (STS)
Technische Universität Hamburg-Harburg
D-21071 Hamburg
{pa.hupe,f.matthes,ho.wegner}@tu-harburg.de

Zusammenfassung

Ein bekanntes Problem beim OOAD-Prozeß ist der systematische Übergang von Prozeßbeschreibungen (Anwendungsfällen, *Use Cases*) der Analysephase zu objektorientierten Entwürfen und Systemrealisierungen. In diesem Artikel stellen wir zusammenfassend die Beiträge des Modells der *Business Conversations* vor, die in den vergangenen drei Jahren in unserer Gruppe zur Lösung dieses Problems entwickelt wurden. Das Modell der *Business Conversations* formalisiert die langandauernde und zielgerichtete Interaktion zwischen Kunden und Dienstleistern in verteilten Systemen. Kunden und Dienstleister sind hierbei entweder menschliche Benutzer oder Softwaresysteme. Das Modell ergänzt die Notationen der UML und unterstützt einen bruchlosen Übergang von der Prozeßmodellierung bis hin zur objektorientierten Realisierung von Softwaresystemen. Die Realisierung wird durch ein *framework* unterstützt, das eine Abstraktion von den Details der Interaktion mit kooperierenden Systemen und Benutzern erlaubt. Wir illustrieren das Modell und den Entwicklungsprozeß anhand eines Anwendungsbeispiels aus einem Kooperationsprojekt mit der SAP AG.

1 Motivation

Die Wandlung von abgeschlossenen, monolithischen und zentralisierten Informationssystemen hin zu offenen, integrativen und verteilten Lösungen erfordert Konzepte, Methoden und Modelle für den Entwurf solcher sich rapide entwickelnden Systeme, um die Auswirkungen der Änderungen in der realen Welt auf die Systeme beschreiben und ihnen folgen zu können [DDJ+98].

In dieser Arbeit stellen wir ausgehend von einer Untersuchung der gängigen objektorientierten Analyse- und Designmethodik [Boo94, JCJÖ92, RBP+91] im Umfeld von UML [FS97] Defizite bei der Modellierung von Geschäftsprozessen fest. Durch Kombination mit der systematischen Entwicklung von Konversationen für die *Business Conversations* gelangen wir zu einem bruchlosen Entwicklungsprozeß.

Vor dem Hintergrund der Analyse und Entwicklung von betrieblichen Informationssystemen und der Analyse und Umsetzung von Geschäftsprozessen haben wir das in [Mat98, Mat97] vorgestellte Modell der *Business Conversations* entwickelt. Es erlaubt die Abstraktion von Details der Kooperation über die *Zeit*, den *Raum* und in verschiedenen *Modalitäten*. Die Grundlage ist für die ersten beiden Belange die Nutzung von Persistenz- und Mobilitätsabstraktionen sowie für letzteres das medien- und aktorenunabhängige Kooperationsmodell.

Wir stellen zunächst die beobachteten Defizite in Abschnitt 2 vor. Um unsere Lösungen vorstellen zu können (Abschnitt 4), schildern wir in Abschnitt 3 das Modell der *Business Conversations*. In Abschnitt 5 schildern wir die praktische Umsetzung anhand eines realen Projekts, einem Ho-

telreservierungssystem, das in Zusammenarbeit mit der SAP AG prototypisch zum Untersuchen dieser Konzepte entwickelt wurde. Abschnitt 6 faßt zum Schluß die Erkenntnisse noch einmal zusammen.

2 Defizite gängiger Modellierungstechniken

Ausgangspunkt der Systemanalyse bilden Geschäftsprozeßmodelle zur Beschreibung der durch das System zu erbringenden Dienste. Ereignis-Prozeß-Ketten (EPK) und Anwendungsfälle in UML haben sich in der Praxis als ein wichtiges und erfolgreich eingesetztes Mittel zur Geschäftsprozeßmodellierung erwiesen.

Das größte Defizit der EPK ist die starke Trennung von Daten-, Funktions- und Prozeßsicht. Objektorientierte Notationen wie UML überwinden die Trennung von Funktionen und Daten, jedoch sind die Prozeßbeschreibungen nur informell mit dem Objektmodell verbunden [Boo96]. Es existiert kein formales Verfahren zur Entwicklung der Zustandsdiagramme und des Objektmodells aus den die Anforderungsanalyse ausmachenden Anwendungsfällen und den Akteuren. In der Regel muß hier „kochrezeptartig“ vorgegangen werden. Bei OMT wird zumindest die Ableitung der Klassen direkt aus den Akteuren vorgeschlagen [RBP+91]; eine Beschreibung der Interaktionen läßt sich so aber noch nicht gewinnen.

Weiterhin gilt es grundsätzlich festzuhalten, daß es keine über die Anwendungsfälle hinausgehende Modellierung der Interaktion zwischen Benutzer und System gibt: Zum Beispiel zerfällt ein für Kunde und Unternehmen zusammenhängender Geschäftsprozeß wie die Auftragsabwicklung in eine große Zahl von unabhängig voneinander modellierten aufeinanderfolgenden Anwendungsfälle (Anfrage, Angebot, Bestellung, Lieferung, Bezahlung, Mahnung, ...).

Als weitere Hürde stellt sich der Schritt innerhalb der Entwurfsphase beim Übergang vom idealen Objektmodell zum realen Objektmodell dar. Bereits an dieser Stelle ist der Entwickler gezwungen, stereotype „Hilfsklassen“ zur Visualisierung, Verwaltung von Sitzungen, Ereignisverwaltung etc. einzuführen, die keinerlei Anwendungssemantik tragen und den Entwurf insgesamt erschweren.

Durch beide Defizite führen selbst kleine Prozeßänderungen zu weitreichende Konsequenzen im Systementwurf. Falls zum Beispiel als Ergebnis einer Geschäftsprozeßoptimierung (*business process reeingineering*) Prozeßschritte in der Auftragsabwicklung reorganisiert werden, hat dies zur Folge, daß zahlreiche Hilfklassen angepaßt werden müssen. Unglücklicherweise unterstützen die Anwendungsfalldiagramme den Entwickler bei dieser Aufgabe in keiner Weise.

3 Prozeßmodellierung mit *Business Conversations*

Bevor wir in Abschnitt 4 die Lösungsansätze erläutern, stellen wir in diesem Abschnitt das Modell der *Business Conversations* knapp und im Zusammenhang vor. Ausführliche Darstellungen finden sich in [Mat98, Mat97] sowie [Joh97, Weg98].

Auf den linguistischen Arbeiten der Sprechakttheorie von AUSTIN und SEARLE [Aus62, Sea69] basierend versuchten Arbeiten im Umfeld der computerunterstützten Gruppenarbeit (*computer supported cooperative work*, CSCW), Sprechakte, die im Rahmen von Geschäftsprozessen auftreten, zu identifizieren und zu klassifizieren. In [Win87] und [FGHW88] wird im *language/action*-Ansatz dafür eine wesentliche Art des Sprechakts genannt: *Conversations for Actions* (CfA). Der zentrale Gedanke dabei ist, daß zwei Partner zum Zwecke der Erledigung ihres gemeinsamen Anliegens eine Reihe von Sprechakten vollführen; mit anderen Worten: eine *Konversation führen*. Dem Kooperationsmodell der *Business Conversations* liegt die CfA zugrunde. Dabei vereinbaren die Partner wechselseitig künftige Aktionen, die direkt an die geäußerten Sprechakte der jeweiligen Partner gekoppelt sind.

Das Interaktionsmuster der Sprechakte ist daher *universell* in dem Sinne, daß es unabhängig von der Art der Akteure (Software oder Menschen) und den von ihnen verwendeten Kommunikationsmedien und -protokollen ist (Medienunabhängigkeit). Die Interaktion zwischen zwei menschlichen Akteuren, zwischen einem Menschen und einem maschinellen Akteur und auch zwischen zwei maschinellen Akteuren ist innerhalb des Modells transparent (Aktorenunabhängigkeit).

Das Modell der *Business Conversations* geht von der Interaktion zwischen zwei Partnern aus, von denen der eine in der Rolle des *Kunden* und der andere in der des *Dienstleisters* handelt. Die Interaktion zwischen beiden Akteuren kann in einen Zyklus mit vier Phasen unterteilt werden (siehe Abbildung 1) und folgt stets diesem Verlauf:

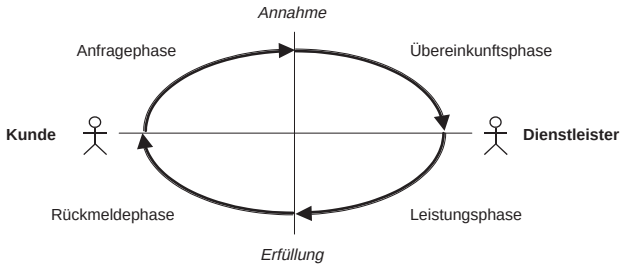

Abbildung 1: Struktur eines kundenorientierten Geschäftsprozesses

Im ersten Schritt, der *Anfragephase*, fragt der Kunde unverbindlich beim Dienstleister nach einer Dienstleistung an. Die Phase endet mit der Annahme der Anfrage. Die Initiative geht dabei immer vom Kunden aus. Mit Annahme der Anfrage beginnt die zweite, die *Übereinkunftsphase*. In ihr erfolgt die Verhandlung zwischen Kunde und Dienstleister über die genauen Modalitäten des zu erbringenden Dienstes. Am Ende steht die Übereinkunft. Nach der Übereinkunft beginnt die *Leistungsphase*, in der der Dienstleister die vereinbarte Leistung erbringen muß. Sie endet aus Sicht des Dienstleisters mit der Erfüllung der Leistung. Die vierte Phase, die *Rückmeldephase*, dient dem Kunden dazu, dem Dienstleister mitzuteilen, ob die zuvor erbrachte Leistung seinen Anforderungen und Erwartungen entspricht.

Um eine Brücke zwischen den abstrakten Modellen und den softwaretechnischen Realisierungsmöglichkeiten schlagen zu können, ist eine Formalisierung des eigentlichen Inhalts der Konversationen und deren genauen Ablaufs nötig. Wir verwenden dazu im folgenden zwei getrennte Beschreibungsebenen für Konversationen: Die Ebene der *Konversationsspezifikationen*, auf der die Struktur von möglichen Konversationen beschrieben wird und die der Typebene einer Programmiersprache entspricht, sowie die Ebene der *Konversationsinstanzen*. Dies ist die der konkreten Konversationen, die zwischen zwei konkreten Akteuren stattfinden. Analog zu Programmiersprachen ist jede Konversationsinstanz Ausprägung ihres Typs, der Konversationsspezifikation.

Das Modell sieht die Gliederung jeder Konversation in einzelne *Dialogschritte* vor, wobei nach der Eröffnung der Konversation durch die initiale Anfrage des Kunden in jedem Schritt ein formal spezifizierter *Dialog*, den man als Abstraktion eines Formulars sehen kann, vom Dienstleister an den Kunden geschickt wird, der Kunde diesen inspiziert und den Inhalt ggf. verändert und ihn schließlich unter Angabe einer aus einer Menge von für diesen Dialog angegebenen *Anfragen* zurückschickt. Dies geschieht solange, bis beide Partner übereinkommen, daß die Konversation beendet ist.

Eine Konversationsspezifikation definiert neben ihren Namen alle in einer Konversation möglichen Dialoge; dazu erhält jeder Dialog einen innerhalb der Konversationsspezifikation eindeutigen Namen sowie eine formale, strukturierte Beschreibung des Inhalts durch ein simples Typsystem, ferner eine Menge von Anfragen für jeden Dialog, aus der eine als Antwort des Kunden gefordert wird; jede Anfrage hat dazu einen innerhalb des Dialoges eindeutigen Namen.

Schließlich enthält die Spezifikation zu jedem Dialog die Menge aller möglichen Folgedialoge (dazu werden zu jeder Anfrage, die der Dialog enthält, die Namen der erlaubten Folgedialoge angegeben).

Abgesehen von dem Typsystem für die Dialoginhalte läßt sich eine Konversationsspezifikation als Spezifikation eines nichtdeterministischen endlichen Automaten (NFA) und der Ablauf einer Konversation als Interpretation desselben betrachten. Dabei bilden die Dialoge die Zustände und die Anfragen des Kunden die Eingaben des Automaten, wobei der Nichtdeterminismus des Automaten durch die Möglichkeit mehrerer Folgedialoge eines Dialoges auf eine Anfrage zum Ausdruck kommt. Die Zustandsübergänge werden also im Rahmen der Spezifikation durch die Auswahl der Anfrage durch den Kunden und die Auswahl des Folgedialoges durch den Dienstleister festgelegt.

Die Spezifikation des Inhalts eines Dialoges wird durch ein einfaches Typsystem geleistet. Es sieht neben Basisdatentypen wie `boolean`, `integer`, `real`, `string`, `date`, `time` und `currency` zusammengesetzte Typen wie `records`, `variants`, `single`- und `multiple-choice` sowie `sequences` vor. Dabei werden alle Komponenten des Inhalts sowie alle `record`- und `variant`-Komponenten durch Namen gekennzeichnet, um den Zugriff zu ermöglichen. Die `record`- und `variant`-Typen dienen zum Aufbau von komplexeren, strukturierten Typen. Dabei sind alle Typen orthogonal kombinierbar. Die `choice`-Typen, deren Instanzen sich wertmäßig nicht von den entsprechenden Typen unterscheiden, dienen dazu, die *mögliche* Wertemenge einzuschränken. Die Wertemengen sind nicht Teil der Spezifikation, sondern werden erst zur Laufzeit den Instanzen zugewiesen. Durch die Bereitstellung solcher Typen wird die Visualisierung von Datenstrukturen unterstützt, etwa (je nach Kardinalität) durch *checkboxes*, *radiobuttons* oder geeignete Auswahllisten. Der `sequence`-Typ dient zur Definition und Benutzung von Massendaten.

Der Aufbau einer vollständigen Konversationsspezifikation läßt sich durch ein Klassendiagramm veranschaulichen (vgl. Abbildung 2), das auch die Grundlage für ein *framework* zur agentenbasierten Realisierung kooperativer Informationssysteme bildet [Weg98, Mat98].

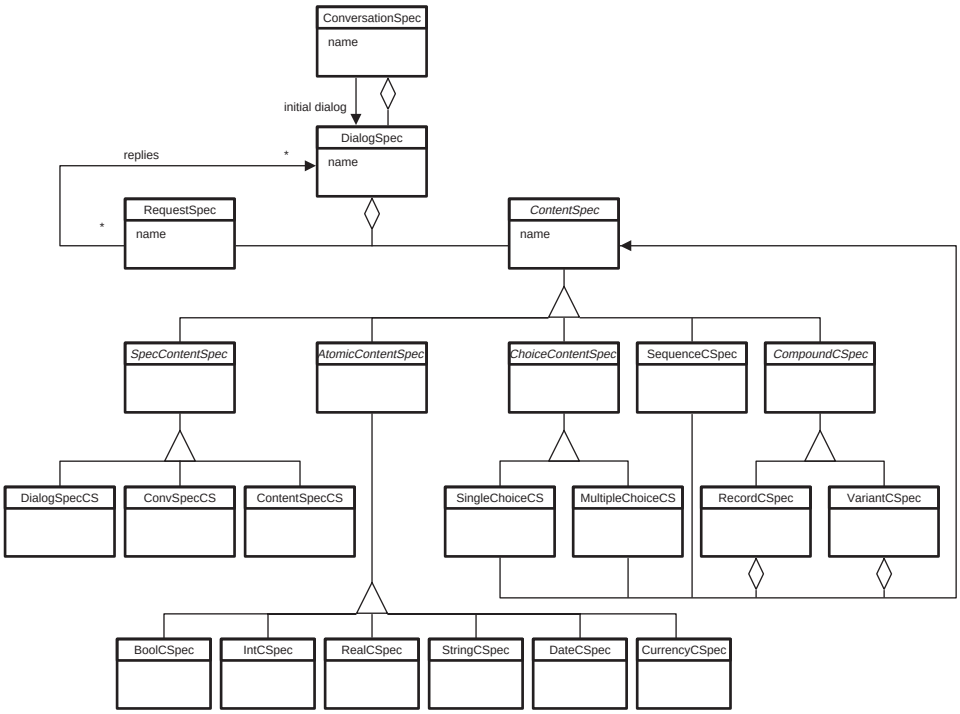

Abbildung 2: Objektorientierte Modellierung von Konversationsspezifikationen

Durch die Einführung der Typen `convspec`, `dialogspec` und `contentspec` sind alle Spezifikationen selbst Objekte erster Klasse in ihrem eigenen Typsystem, d.h. Spezifikationen können wiederum Inhalte sein, die innerhalb von Konversationen verwendet werden können. So wird es besonders einfach, Meta-Dienste zu realisieren, wenn diese selbst *Business Conversations* zur Interaktion verwenden.

4 Lösungsbeitrag zu einem bruchlosen Softwareentwicklungsprozeß

4.1 Einbettung von Anwendungsfällen in kundenorientierte Geschäftsprozesse

Um die zuvor beschriebenen Probleme zu lösen, ist es hilfreich, einen Schritt zurückzutreten und zunächst das Gesamtbild der Anwendungsfälle (*use cases*) zu betrachten.

In der UML-Modellierung der Anwendungsfälle sind die darunterliegenden Geschäftsprozesse nicht mehr zu erkennen. Sie werden vielmehr durch nicht zusammenhängende Fälle beschrieben (beispielsweise „Anmelden beim System“, „Buchen eines Zimmers“, „Abfrage der Kreditkartennummer“, „Bestätigung der Buchung“ etc.). Durch Zusammenführen der inhaltlich zu einem Prozeß gehörenden Fälle zu einer *Konversationsspezifikation* stellen wir zunächst die Bezüge untereinander (wieder) her. Eine semantische Strukturierung wird zusätzlich durch eine Zuordnung zu den einzelnen Phasen (vgl. Abbildung 1) vorgenommen. Komplexe Szenarien mit mehr als zwei Akteuren lassen sich gemäß der in [Mat98, Joh97, Weg98] genannten Strukturierungsprinzipien (Koordination, *Broker*, Subkonversationen, sekundäre Konversationen) durch mehrere Konversationsspezifikationen beschreiben.

Das Erstellen einer Konversationsspezifikation umfaßt dann die Definition der einzelnen Dialoge und der dazwischen möglichen Übergänge (Anfragen). Einzelne Anwendungsfälle lassen sich dabei auf einen oder mehrere Dialoge abbilden. An dieser Stelle muß das bei der Anforderungsermittlung gesammelte Wissen (Domänenwissen) über die Geschäftsprozesse verwendet werden, um die einzelnen Dialoge, ihre Bedeutung und die enthaltenen Informationen festzulegen [Rip98]. Zu jedem Dialog werden die Inhalte in Form von strukturierten, hierarchischen Dokumenten spezifiziert [Joh97]. Dies gewährleistet eine aktoren- und medienunabhängige (und so auch implementierungsunabhängige) Modellierung der semantischen Inhalte der Interaktion.[1]

Um eine Trennung der Anwendungslogik von dem die Kooperation in Raum, Zeit und verschiedenen Modalitäten unterstützenden *framework* zu erreichen, wird sie in *Regeln* isoliert, die – gemäß der Sichtweise als NFA – die Zustandsübergänge entscheiden [Weg98]. Dabei ist zwischen Kunden- und Dienstleisterregel zu unterscheiden.

Mit dem Zusammenfassen von Anwendungsfällen zu Konversationsspezifikationen und dem anschließende Herunterbrechen der Bearbeitung der einzelnen Anfragen und Dialoge auf Regeln entsteht so ein sehr fein granulares Modell für die im System vorkommenden Ereignisse.[2] Die fein granularen Dialoge, Anfragen und Regeln sowie die Akteure bilden jetzt den Ausgangspunkt für die Identifizierung und den späteren Entwurf der eigentlichen Geschäftsobjekte (*business objects*), auf denen die Systeme (z.B. ein Buchungssystem als Dienstleister) arbeiten [Rip98]. Ab hier werden die Geschäftsobjekte mit der bekannten Vorgehensweise weiter spezifiziert (Objektmodell) und die Modellierung durch weitere Diagramme (Zustandsdiagramme etc.) ergänzt.

Abbildung 3 verdeutlicht das Ineinandergreifen der Entwicklungsschritte von UML und *Business Conversations*.

[1] In Zukunft könnte sich XML als geeignete Austauschform etablieren.

[2] Diese entsprechen im übrigen genau den in den EPK modellierten Ereignissen [Rip98].

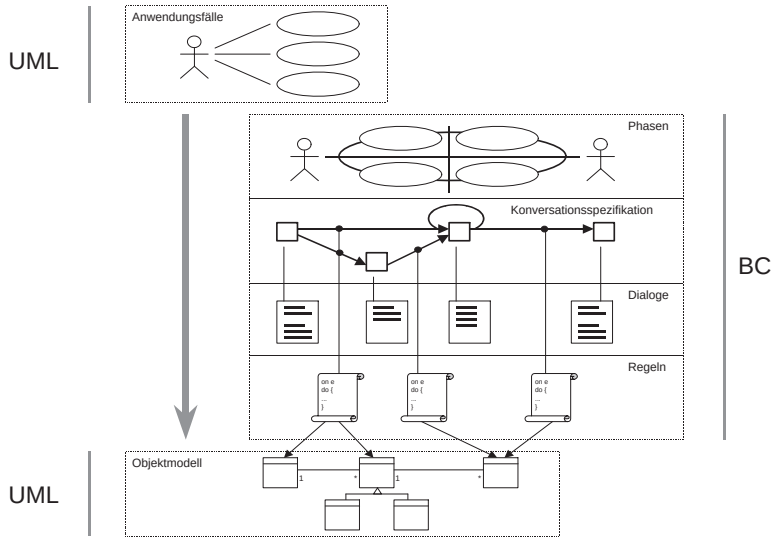

Abbildung 3: Die Brücke zwischen Anwendungsfällen und Objektmodell

4.2 Abstraktion von Interaktionsdetails durch ein *framework*

Dem zweiten in Abschnitt 2 angesprochenen Problem beim Übergang von idealen zum realen Objektmodell läßt sich auf einer mehr technischen Ebene begegnen: Durch Verwendung eines geeignet entworfenen und implementierten *frameworks* zur Repräsentation von Konversationsspezifikationen und Konversationen (Dialogen, Anfragen und Inhalten), zur Abwicklung der Kommunikation zwischen Akteuren gemäß der Spezifikation sowie zur Verwaltung verschiedener, paralleler Konversationen (Sitzungen etc.) und der geordneten Regelausführung entfällt im typischen dialogorientierten Anwendungsfall betrieblicher Prozesse der Entwurf und die Implementierung von Hilfsklassen (vgl. Abschnitt 2) häufig ganz oder reduziert sich zumindest drastisch. Konkret erlaubt die Benutzung der *frameworks*, die in [Joh97, Weg98] entwickelt und beschrieben wurden, die Abstraktion von *Persistenz*, *Mobilität*, *Sitzungsverwaltung*, *Medien-* und *Aktorenabhängigkeiten* sowie die *orts-* und *migrationstransparente* Nutzung von beliebigen Diensten. Durch die Entwicklung eines *generischen Kunden*, der in der Lage ist, Konversationen zu visualisieren [Weg98, Rip98, Zad98], entfällt ferner der ansonsten nicht unerhebliche Aufwand der Realisierung einer graphischen Benutzungsoberfläche. Durch diese Abstraktionen bzw. Entkopplung reduziert sich der Aufwand der Entwicklung erheblich und ermöglicht die volle Konzentration auf die Anwendungslogik.

4.3 Unterstützung für kooperierende verteilte Informationssysteme

Bei einer weitergehenden Betrachtung stellt sich heraus, daß die Verschiebung der Sichtweise von Anwendungsfällen als asymmetrische Sicht auf eine Benutzer-System-Beziehung hin zu einer symmetrischen Sicht auf eine Akteur-Akteur-Beziehung (bzw. Kunde-Dienstleister) zu einer weitaus besseren Modellierungsfähigkeit für den Entwurf von kooperativen Informationssystemen führt [Mat97]. Die symmetrische Sicht, das abstrakte Interaktionsprotokoll und die weitgehende Entkoppelung von technischen Implementierungsdetails verbessert die Möglichkeiten zur unabhängigen Evolution der beteiligten Systeme. Da nach [DDJ+98, DDJ+97] kooperative Informationssysteme als verteilte Systeme betrachtet werden können, in denen autonome Akteure ihre langandauernden Aktivitäten koordinieren, um ein gemeinsames Ziel zu erreichen, stellt unser Modell eine dafür gut passende Entwicklungsmethodik bereit.

5 Beispiel: *Hotel-Front-Office-System*

Um die Vorteile des oben geschilderten Vorgehens zu verdeutlichen, schildern wir anhand eines tatsächlich realisierten Projektes die Umsetzung. Die Aufgabe bestand darin, ein umfangreiches *Hotel-Front-Office-System* zur Hotelzimmerreservierung prototypisch zu entwerfen und zu implementieren und wurde als Kooperationsprojekt mit der SAP AG durchgeführt [Rip98].

Die Analyse besteht nach der Anforderungsermittlung aus der Aufstellung von Anwendungsfällen, die durch die üblichen Dokumente wie Aktorenkataloge etc. vervollständigt werden. Typische Anwendungsfälle etwa sind „Suche Geschäftspartner“, „Suche verfügbare Produkte“ oder „Erstelle Angebot im Anfragekontext“. Ausgehend davon werden für alle Anwendungsfälle die zur Abwicklung des Geschäftsprozesses nötigen Interaktionen zwischen Benutzer und System durch eine Folge von HTML-Formularen beschrieben. Veknüpft wurden diese jeweils durch eine Reihe von möglich Folgeformularen, die durch *submit-buttons* erreichbar sind. Diese Formulare dienen zunächst lediglich zur Spezifikation der Interaktionen.[3] Durch informelle Zuordnung zu den einzelnen Phasen des Interaktionsmodells der *Business Convesations* und die o.g. Verknüfungen der Formulare ergibt sich so automatisch eine *Konversationsspezifikation*. Dabei wird der Benutzer als Kunde und das System als Dienstleister betrachtet. Jeder Anwendungsfall kann so auf eine Konversationsspezifikation abgebildet werden; dabei stellt jedes HTML-Formular einen Dialog und jeder *submit-button* eine Anfrage im Sinne der *Business Conversations* dar. Die Formularfelder eines Formulars bilden die Inhaltsspezifikation.

Die zentrale Idee ist nun, aus diesen HTML-Formulare einerseits maschinell Konversationsspezifikationen zu gewinnen, die von einem wie oben beschriebenen *framework* genutzt werden können, um die Konversation abzuwickeln, und andererseits, die Formulare zur generischen Visualisierung der ablaufenden Konversationen auf der Seite des Benutzers zu verwenden. Das Vorgehen ist in Abbildung 4 illustriert.

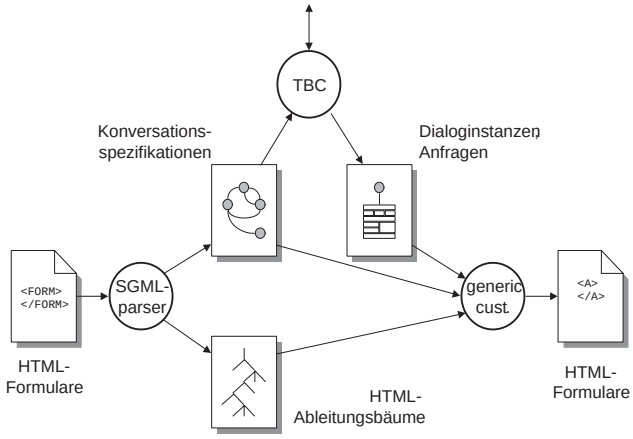

Abbildung 4: Vorgehen bei der Wiederverwendung der HTML-Formulare

Aufbauend auf dem im Rahmen von [Weg98] entwickelten *Tycoon Business Conversations*-System (TBC) wurde dazu ein *generischer Kunde* entwickelt, der unter Verwendung eines SGML-Parsers aus den HTML-Formularen eine Konversationsspezifikation generiert und ferner als Kunde im Sinne des TBC-Systems unter Benutzung der Spezifikation und der HTML-Formulare als HTTP-Server agiert, um die Konversation mit einem menschlichen Benutzer zu führen. Durch die Generizität dieses Ansatzes sind auf der Kundenseite auch keine Regeln zu implemetieren, da alle Entscheidungen vielmehr von dem menschlichen Benutzer getroffen werden. In Abbildung 5 ist

[3] Nebenbei bilden sie durch ihre „Anfaßbarkeit“ auch eine gute Grundlage für die Diskussion mit den künftigen Anwendern des Systems.

der Ablauf der Umsetzung auf technischer Ebene dargestellt. Hier ist gut zu erkennen, wie nach dem Aufbau einer Sitzung die Dienstleisterseite – unabhängig von der Realisierung der Kundenseite – Dialoge und Anfragen austauscht.[4] Die *generische Kundenregel* sorgt in Verbindung mit dem parallel laufenden HTTP-Server für die Übersetzung von Dialogen in HTML-Formulare und umgekehrt von Formularen zurück in Dialoge und Anfragen.

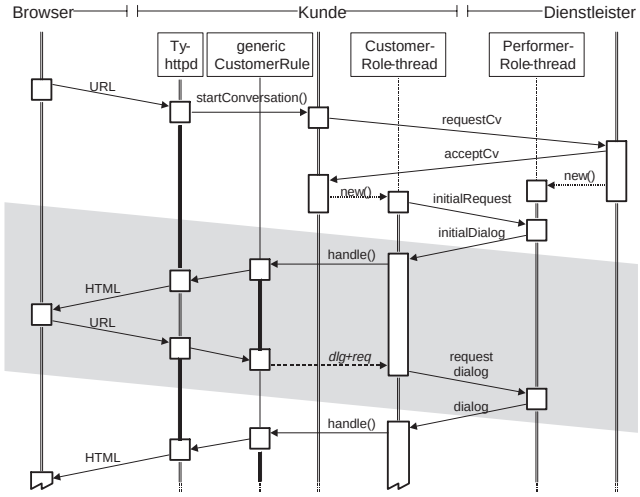

Abbildung 5: Technische Umsetzung der Interaktion über das HTTP

Auf der anderen Seite, der Dienstleisterseite, die das Zimmer-Resevierungssystem darstellt, geben die durch die Konversationsspezifikation vorgegebenen Regeln einen fein strukturierten Rahmen für die Entwicklung der eigentlichen Anwendungslogik. Entwurf und Implementierung des Objektmodells für die Funktionalität des Hotel-Zimmer-Reservierungssystems können gänzlich von sekundären Aspekten wie Visualisierung, Sitzungen, Nebenläufigkeit etc. abstrahieren. Die Regeln stellen sehr „kleine“ Transitionen dar, die sich mit wenig Aufwand implementieren lassen (z.B. „ein Zimmer aus der Liste der angebotenen Zimmer im Detail darstellen“, „die Buchung genau des angezeigten Zimmers bestätigen“ etc.).

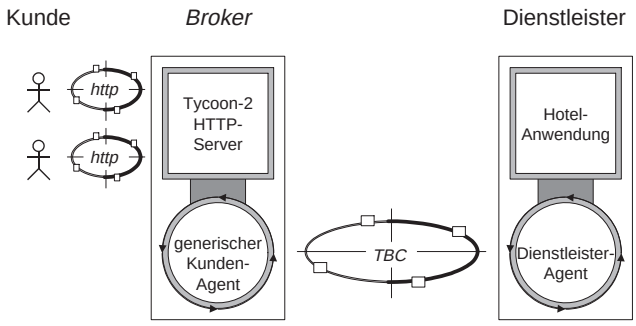

Abbildung 6: Kooperierende Informationssysteme

Die Gesamtarchitektur des entwickelten Systems ist abschließend in Abbildung 6 skizziert.

[4] Die auf der Dienstleiterseite vorhandenen Regel-Aufrufe sind der Übersichtlichkeit halber nicht mit dargestellt.

6 Conclusio

Wir haben eine zielgerichtete Herangehensweise zur Modellierung von Geschäftsprozessen auf Basis von UML und den *Business Conversations* aufgezeigt. Die genannten Defizite bei der Modellierung mit UML, namentlich der Bruch zwischen Anwendungsfällen und Objektmodell in der Analysephase einerseits und der Übergang vom idealen zum realen Objektmodell in der Entwurfsphase andererseits, haben wir durch Verwendung des Modells und der zugehörigen Methodik der *Business Conversations* verringert. Somit sind wir zu einem Entwicklungsprozeß gelangt, der die Modellierung von Geschäftsprozessen durchgängig von der Analyse bis zur Implementation beschreibt.

Die Vorteile dieser Modellierungstechnik sind: die Verwendung der verbreiteten Modellierungssprache UML, die für alle relevanten Modellierungsaspekte adäquate Darstellungsmittel (Diagramme) bereitstellt; die Verwendung der Konversationsspezifikationen der *Business Conversations*, die im Gegensatz zu EPK *(Ereignis-Prozeß-Ketten)* zusammengehörige Prozesse von Kunde und Dienstleister in einer gemeinsamen Ablaufspezifikation beschreiben, was bei der Evolution von Geschäftsprozessen unnötigen Abstimmungsaufwand vermeidet; die symmetrische Sicht der *Business Conversations* auf Akteur-Akteur-Beziehungen statt durch Anwendungsfälle implizierten Akteur-System-Beziehungen. Diese ermöglicht eine elegante und zielgerichtete Modellierung beim Entwurf von *kooperativen Informationssystemen*, da solche Kooperationsbeziehung besser als Akteur-Akteur denn als Akteur-System Beziehungen verstanden werden; die Verwendung eines *frameworks* zur Unterstützung *Business Conversations*, das den Übergang vom idealen zur realen Objektmodell in der Entwurfsphase stark vereinfacht, da es die meisten zur Kommunikation und Kooperation nötigen Basismechanismen bereits enthält und generische Visiualisierungskomponenten bereitstellt. Die Abstraktion des *frameworks* von Aktoren und Modalitäten und deren technischer Realisierung schafft zudem die nötige Flexibiltät für die unabhängige Weiterentwicklung (Evolution) der den kooperativen Informationssystemen zugrundeliegenden Kunde- und Dienstleistersystemen.

Das bisherige in [Weg98] beschriebene System ist in TYCOON-2 implementiert. Für den Einsatz in weiteren Anwendungsprojekten wird es zur Zeit in JAVA re-implementiert. Neben dem im Beispiel in Abschnitt 5 illustrierten *broker*-Muster bestehend aus Kunde, *broker* und Dienstleister untersuchen wir in anderen Anwendungsprojekten weitere Konversationsmuster (z.B. Koordinator) zwischen kooperierenden Informationssystemen.

Literatur

[Aus62] J. Austin. How to do things with words. Technical report, Oxford University Press, Oxford, 1962.

[Boo94] G. Booch. *Object-Oriented Design with Applications*. Benjamin / Cummings Publishing Company, Inc., second edition, 1994.

[Boo96] Grady Booch. *Object Solutions: Managing the Object-Oriented Project*. Addison-Wesley Publishing Company, 1996.

[DDJ+97] Giorgio De Michelis, Eric Dubois, Matthias Jarke, Florian Matthes, John Mylopoulos, Mike Papazoglou, Klaus Pohl, Joachim Schmidt, Carson Woo, and Eric Yu. Cooperative information systems: A manifesto. In Mike P. Papazoglou and Gunther Schlageter, editors, *Cooperative Information System: Trends and Directions*. Academic Press, 1997.

[DDJ+98] Giorgio De Michelis, Eric Dubois, Matthias Jarke, Florian Matthes, John Mylopoulos, Joachim Schmidt, Carson Woo, and Eric Yu. A three-faceted view of information systems. *Communications of the ACM*, Dezember 1998.

[FGHW88] F. Flores, M. Graves, B. Hartfield, and T. Winograd. Computer systems and the design of organizational interaction. *ACM Transactions on Office Information Systems*, 6(2):153–172, 1988.

[FS97] Martin Fowler and Kendall Scott. *UML Distilled: Applying the Standard Object Modelling Language*. Addison-Wesley Publishing Company, 1997.

[JCJÖ92] I. Jacobson, M. Christerson, P. Jonson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Publishing Company, 1992.

[Joh97] Nico Johannisson. Eine Umgebung für mobile Agenten: Agentenbasierte verteilte Datenbanken am Beispiel der Kopplung autonomer „internet web site profiler“. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany, April 1997.

[Mat97] Florian Matthes. Mobile processes in cooperative information systems. In *Proceedings STJA'97 (Smalltalk und Java in Industrie und Ausbildung)*, Erfurt, Germany, September 1997. Springer-Verlag.

[Mat98] F. Matthes. Business conversations: A high-level system model for agent coordination. In *Database Programming Languages: Proceeding of the 6th International workshop; proceedings / DBPL-6, Estes Park, Colorado, USA, August 18 - 20, 1997*. Springer-Verlag, 1998.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Rip98] Volker Ripp. Verbesserung der Lokalität und Wiederverwendbarkeit von Geschäftsprozeßspezifikationen: Probleme und Lösungsansätze am Beispiel kundenorientierter Hotelgeschäftsprozesse. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany, March 1998.

[Sea69] J. Searle. Speech acts. Technical report, Cambridge University Press, Cambridge, 1969.

[Weg98] Holm Wegner. Objektorientierter Entwurf und Realisierung eines Agentensystems für kooperative Internet-Informationssysteme. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany, May 1998.

[Win87] T.A. Winograd. A language / action perspective on the design of cooperative work. Technical Report No. STAN-CS-87-1158, Stanford University, Mai 1987.

[Zad98] Heiko Zade. Visualisierungsunterstützung für kooperative Aktivitäten im Internet. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Germany, March 1998.