

Business-Objekte: Konzepte, Architekturen, Standards

Mathias Weske
Lehrstuhl für Informatik, Institut für Wirtschaftsinformatik
Westfälische Wilhelms-Universität Münster
E-mail: weske@helios.uni-muenster.de

Zusammenfassung

Betriebliche Anwendungssysteme können effektiver entwickelt und schneller an veränderte Umweltbedingungen angepasst werden, falls sie durch Kombination objektorientiert entwickelter fachlicher Komponenten erstellt wurden; diese Komponenten werden als Business-Objekte bezeichnet. Der vorliegende Beitrag diskutiert die Grundlagen dieser neuen Technologie sowie mit der OMG Business Object Facility und dem San Francisco Framework zwei zentrale Ansätze zur Entwicklung von Anwendungssystemen auf der Basis von Business-Objekten.

1 Motivation und Überblick

Aufgrund der Öffnung der Märkte in Europa und des Trends zur Globalisierung sehen sich Unternehmen gezwungen, schnell und effektiv auf veränderte Marktsituationen zu reagieren [FeSi97, Suth97, Bohr98a]. Da betrieblichen Informationssystemen dabei eine besondere Bedeutung zukommt, sind die Anforderungen an diese in den letzten Jahren in Bezug auf Flexibilität, Verlässlichkeit und Anpassbarkeit stark angewachsen. Es wird zunehmend klarer, dass monolithisch aufgebaute Systeme immer weniger zur Bewältigung der enormen Komplexität und der Forderung nach Flexibilität und effektiver Systementwicklung heutiger Anwendungen geeignet sind. Die Entwicklung betrieblicher Anwendungssysteme auf der Basis autonomer, objektorientiert modelliert und entwickelter fachlicher Komponenten stellt einen viel versprechenden Ausweg aus dieser Situation dar; diese Komponenten werden als Anwendungs- oder Business-Objekte (engl.: *Business Objects*) bezeichnet [Casa95, HuSR98]. Eine aktuelle Studie besagt, dass von Anwendern und Entwicklern in Business-Objekten ein großes Potential zur effektiven Entwicklung verlässlicher Informationssysteme gesehen wird, dass aber derzeit noch nicht deutlich ist, was genau unter Business-Objekten verstanden wird, welche Aktivitäten im Bereich der Standardisierung von Business-Objekten derzeit zu beobachten sind und wie diese Objekte zur Lösung konkreter betrieblicher Problemstellungen eingesetzt werden können [HuSR98]. In dem vorliegenden Beitrag werden die Grundlagen dieser neuen Technologie dargestellt, unterschiedliche Architekturen beschrieben, und aktuelle Standardisierungsaktivitäten dargestellt und im Hinblick auf ihre Entwicklungspotentiale bewertet.

Dieser Bericht gliedert sich wie folgt. In Abschnitt 2 werden die Grundlagen von Business-Objekten erläutert, und es wird erklärt, was unter Business-Objekten zu ver-

stehen ist. Da Unternehmensentscheidungen bezüglich der verwendeten IT-Infrastruktur oft strategische Bedeutung besitzen und nicht ohne die Kenntnis grundlegender Techniken getroffen werden sollten, arbeiten wir in Abschnitt 3 die technische Infrastruktur von Objektsystemen auf; dabei stehen nicht technische Einzelheiten im Vordergrund, sondern konzeptionelle Eigenschaften, die Auswirkungen auf deren Nutzung zur Unterstützung komponentenbasierter, objektorientierter Anwendungssysteme besitzen. In Abschnitt 4 werden Frameworks für Business-Objekte vorgestellt und dabei aktuelle Standardisierungsaktivitäten der Object Management Group (OMG) zur Spezifikation einer Business Object Facility diskutiert; mit San Francisco der Firma IBM wird ein Java-basiertes Framework zur Entwicklung von Business-Objekten präsentiert. Eine Bewertung der vorgestellten Ansätze und ein Ausblick auf weitere Entwicklungen schließen diesen Beitrag.

2 Business-Objekte

In diesem Abschnitt wird diskutiert, warum sich objektorientierte Ansätze zur Modellierung von komponentenbasierten Anwendungssystemen eignen, welche betriebswirtschaftlichen und technischen Gründe für ihre Verwendung sprechen und was unter Business-Objekten verstanden wird.

2.1 Motivation

Die vielleicht wichtigste Grundlage zur Entwicklung wiederverwendbarer und erweiterbarer fachlicher Komponenten besteht in dem Paradigma der Objektorientierung und seiner Anwendung im Kontext betrieblicher Aufgabenstellungen [FeSi90, Inma98, Suth97, NIIP98].

Während objektorientierte Methoden und Techniken bereits seit geraumer Zeit erfolgreich auf systemtechnischer Ebene (u.a. zur Modellierung und Entwicklung graphischer Benutzeroberflächen) verwendet werden, rückt ihr Einsatz auf höheren, anwendungsnahen Abstraktionsebenen erst in jüngere Zeit in den Vordergrund. Diese Entwicklung wird zum einen durch die Weiterentwicklung objektorientierter Programmiersprachen und -modelle sowie durch die Entwicklung stabiler und performanter objektorientierter Datenbanksysteme [Banc88, Heue97] forciert. Zum anderen wird diese Entwicklung durch die offenkundigen Probleme gefördert, die bei dem Einsatz monolithischer Systeme zur Bewältigung komplexer betrieblicher Aufgaben aufgetreten sind und oft nur durch den Einsatz erheblicher Ressourcen behoben werden können.

Nun sollen kurz die zentralen anwendungsspezifischen und technischen Gründe für den Einsatz objektorientierter Techniken zur Entwicklung von Anwendungssystemen diskutiert werden. Zunächst kann die Modellierung komplexer Anwendungen von den mächtigen Abstraktionskonstrukten objektorientierter Techniken [Banc88] profitieren. Bei der Entwicklung betriebswirtschaftlicher Anwendungssysteme sind etwa Personen, Adressen und Konten, aber auch beispielsweise LKWs eines Fuhrunternehmens Objekte der Realwelt, die durch softwaretechnische Objekte abgebildet werden. Mit dem Semantischen Objektmodell (SOM) wird in [FeSi90] ein Ansatz zur Objektmodellierung betrieblicher Informationssysteme und in [FeSi91] ein entsprechendes Vorgehensmodell eingeführt. Diese Arbeiten stellen eine wichtige Grundlage zur Anwendung ob-

jektorientierter Methoden im betrieblichen Umfeld dar; neben einem Objektmodell, in dem Struktur und Verhalten betrieblicher Objekte spezifiziert wird, wird auch die Vorgangsebene untersucht. Damit werden objektorientierte Methoden zur Beschreibung statischer und dynamischer Aspekte verwendet; letztere können systemtechnisch durch Vorgangsteuerungs- bzw. Workflow-Management-Systeme unterstützt werden [GeHS95, LeA194]; in Abschnitt 5 werden aktuelle Entwicklungen bei der Standardisierung von Workflow-Funktionalität unter Verwendung objektorientierter Techniken und ihre Beziehungen zu Business-Objekten diskutiert.

Neben diesen anwendungsspezifischen Gründen für die Verwendung objektorientierter Techniken spielen auch technische Gründe eine wichtige Rolle, die sich aus der Heterogenität heute typischerweise anzutreffender IT-Landschaften ergeben. Unter Heterogenität versteht man in diesem Zusammenhang die Eigenschaft, Rechner unterschiedlicher Hersteller zu verwenden, auf denen unterschiedliche Betriebssysteme, Datenbanksysteme und Anwendungssysteme installiert sind. Während man beim Aufkommen vernetzter Anwendungen zunächst den Weg suchte, Heterogenität zu vermeiden und gleiche, d.h. homogene Systeme einzusetzen, so hat man mittlerweile erkannt, dass Heterogenität ein Faktum in großen Organisationen ist, und dass Techniken zu entwickeln sind, um heterogene Daten und Funktionen zu integrieren, d.h. die Interoperabilität zwischen heterogenen Systemen zu unterstützen.

Bei der Integration heterogener Systeme können objektorientierte Techniken einen wichtigen Beitrag leisten, etwa bei der Integration von Datenbeständen, die von unterschiedlichen Anwendungssystemen verwaltet werden. Unter Integration verstehen wir die Notwendigkeit, von unterschiedlichen Systemen (die unabhängig voneinander entwickelt wurden) verwaltete Daten und Funktionen einheitlich zur Verfügung zu stellen. Durch Verwendung von Wrapper-Techniken (engl.: wrapper = Hülle) können Alt-Systeme von außen als Objekte erscheinen, und andere Objekte können auf diese (d.h. auf deren Daten und auf deren Funktionalität) unter Verwendung der üblichen Objekt-Schnittstellen zugreifen. Dies ist eine Folge der Objektkapselung, wobei Objekte lediglich die Schnittstellen anderer Objekte kennen müssen, um mit diesen kooperieren zu können.

Objektorientierte Methoden und Techniken können daher einen wichtigen Beitrag zur Modellierung komplexer betrieblicher Anwendungen sowie zur Entwicklung der entsprechenden Anwendungssysteme in heterogenen IT-Landschaften leisten.

2.2 Definition

Vor dem Hintergrund anwendungsspezifischer und technischer Gründe für die Verwendung objektorientierter Ansätze zur Modellierung und Entwicklung von Anwendungssystemen werden nun die zentralen Eigenschaften von Business-Objekten untersucht.

Zunächst wird das traditionelle Vorgehen bei der Entwicklung von Anwendungssystemen charakterisiert. In einer frühen Phase des Entwicklungsprozesses werden betriebliche Aufgabenstellungen im Rahmen der Entwicklung eines Fachkonzepts konkretisiert, um daraus anschließend ein implementierungsnahes DV-Konzept zu erstellen, das als Grundlage für die Entwicklung eines Informationssystems dient. Aufgrund der unterschiedlichen Ausrichtungen der beiden Modellierungsaktivitäten werden im Fachkonzept und im DV-Konzept oft unterschiedliche Entitäten behandelt. Auf fachkonzeptioneller Seite werden betriebliche Entitäten wie etwa Lieferanten oder Rechnungen betrachtet; im DV-Konzept sind Programmstrukturen wie etwa Records oder Listen der Betracht-

tungsgegenstand. Weil beide Konzepte darüber hinaus von unterschiedlichen Personen oder Personengruppen erstellt werden, kommt es oft zu Missverständnissen, was zu unvollständigen oder fehlerhaften Lösungsansätzen führen kann.

Der Grundgedanke bei der Verwendung von Business-Objekten zur Entwicklung von Anwendungssystemen besteht darin, betriebliche Entitäten und systemtechnische Entitäten durch gemeinsame Objekte zu modellieren. Dabei werden zwei bislang häufig getrennt voneinander bearbeitete Teilbereiche gemeinsam durch Business-Objekte repräsentiert: Die Modellierung betrieblicher Objekte und Abläufe sowie die Modellierung von Softwaresystemen. Mit der Entsprechung zwischen betrieblichen Objekten der Anwendungsumgebung und softwaretechnischen Objekten des Informationssystems ist eine einheitliche Basis für die Kommunikation zwischen Fachexperten auf der einen und Systementwicklern auf der anderen Seite gegeben und damit eine wichtige Voraussetzung zur Entwicklung adäquater, verlässlicher und effizienter Informationssysteme erfüllt. In [OMG96] werden Business-Objekte wie folgt definiert:

A business object is defined as a representation of a thing active in the business domain, including at least its business name and definition, attributes, behavior, relationships, rules, policies and constraints. A business object may represent, for example, a person, place, event, business process or concept. Typical examples of business objects are: employee, product, invoice and payment. The business object [...] is implemented by one or more objects in the information system.

Business-Objekte sind auf einer hohen Abstraktionsebene angesiedelt, und sie verwenden Dienste darunter liegender, technischer Ebenen; dies wird in der folgenden Definition deutlich [Suth97]:

Technically, business objects encapsulate traditional lower-level objects that implement a business process (i.e., they are a collection of lower-level objects that behave as single, reusable units).

Weil betriebliche Anwendungssysteme ein hohes Maß an Komplexität aufweisen und weil jede Anwendung spezielle, nur für sie geltende Anforderungen besitzt, ist es notwendig, Business-Objekte erweitern und zu komplexen Anwendungssystemen kombinieren zu können. Miteinander kooperierende Business-Objekte können einer Unternehmung zugeordnet sein, sie können aber auch zu verschiedenen Unternehmungen gehören, d.h., Business-Objekte können auch zur Entwicklung unternehmensübergreifender Anwendungssysteme verwendet werden. Die technische Interoperabilität von Business-Objekten wird durch Schnittstellen ermöglicht, die es erlauben, Struktur und Verhalten dieser Objekte unabhängig von Rechnerplattformen und Programmiersprachen zu spezifizieren. Dabei spielt die Standardisierung von Business-Objekten eine wichtige Rolle. Sie bildet die Voraussetzung für die Kombination von Business-Objekten unterschiedlicher Hersteller zu komplexen Anwendungssystemen. Konkrete Standardisierungsbemühungen für Business-Objekte werden in Abschnitt 4 mit der OMG Business Object Facility und dem San Francisco Framework der Firma IBM diskutiert; zuvor werden die Grundlagen verteilter Objektsysteme überblickartig dargestellt, wobei insbesondere auf den CORBA-Standard eingegangen wird.

3 Verteilte Objektsysteme

In diesem Abschnitt werden die Grundlagen verteilter Objektsysteme sowie mit Corba [Sieg96] eine objektorientierte Middleware dargestellt, die von einer Vielzahl von Herstellern unterstützt wird und mittlerweile als Industriestandard für objektorientierte, verteilte Middleware angesehen wird [Lewa98].

3.1 Verteilte Objekte und Middleware

Wie im vorhergehenden Abschnitt diskutiert, ist der objektorientierte Ansatz unter anderem aufgrund der Kapselung von Struktur und Verhalten von Objekten zur Implementierung komponentenbasierter Softwaresysteme angemessen. Diese Systeme sind oft nach dem Client/Server-Prinzip organisiert, d.h. es gibt eine logische Separierung von Dienst-Anforderern (Clients) und Dienst-Erbringern (Servern). Im allgemeinen sind Clients und Server durch Prozesse realisiert, die unabhängig voneinander ablaufen und lediglich durch Anfragen und Antworten miteinander kommunizieren, was oft durch request/reply-Paare charakterisiert wird [Lewa98].

Beim objektorientierten Ansatz sind Clients und Server als Objekte modelliert, die autonom sind und durch Nachrichten miteinander kommunizieren können. Durch die Objektkapselung ist eine natürliche Möglichkeit zur standardisierten Kommunikation zwischen Clients und Servern gegeben: Durch Objektschnittstellen wird festgelegt, welche Nachrichten Objekte "verstehen" und welches Format die Nachrichten besitzen. Die Kommunikation zwischen Objekten eines in einer objektorientierten Programmiersprache entwickelten Programms folgt dem Client/Server-Prinzip recht unmittelbar: Ein Client-Objekt sendet eine Nachricht an ein Server-Objekt, welches die Nachricht erhält und nach Ausführung einer Methode ein Ergebnis an das Client-Objekt zurück sendet. Bei einer unmittelbaren Kommunikation zwischen Client und Server spricht man von einer 2-Tier-Architektur, falls das Server-Objekt zur Beantwortung der Anfrage seinerseits eine Anfrage an ein weiteres Objekt richtet, so handelt es sich um eine 3-Tier-Architektur [Lewa98].

Man spricht von lokalen Objekten, wenn sich alle miteinander kommunizierenden Objekte einer Anwendung in einem Programm befinden, welches auf einem Rechnersystem ausgeführt wird. Eine solche Struktur trifft man bei der objektorientierten Programmierung häufig an, etwa in Smalltalk- oder C++-Programmen. Man spricht demgegenüber von verteilten Objekten bzw. von verteilten Objektsystemen, wenn Objekte über Rechengrenzen hinweg miteinander kooperieren und gemeinsam eine Anwendung bilden. Bei verteilten Objekten erfolgt die Kommunikation von Client- und Server-Objekten über ein Netzwerk und die dazugehörige Software, heute meist das Protokoll TCP/IP, auf dem auch das Internet basiert. Dieses Protokoll stellt Dienste zur Verfügung, um etwa Verbindungen zwischen entfernten Rechnern aufzubauen oder Daten zwischen Rechnern zu übertragen. Bezüglich verteilter Objekte bedeutet dies, dass die Netzwerk-Software für die eigentliche Übermittlung von Daten zwischen Client und Server verantwortlich ist.

Da die Programmierung von Netzwerk-Software im allgemeinen recht aufwendig ist, wird eine Software eingesetzt, die die Kommunikation zwischen Clients und Servern auf einfache Weise ermöglicht, es wird also eine Schicht zwischen der Anwendungsebene (Client/Server) und der Netzwerksoftware (TCP/IP) verwendet. Diese Software wird

aufgrund ihrer Position im Englischen als *Middleware* bezeichnet. Die Middleware verwendet die durch die Netzwerksoftware bereitgestellten Dienste, um einfache Kommunikation zwischen Clients und Servern zu ermöglichen. Damit kann sich der Entwickler mit den Problemstellungen der Anwendungs- bzw. Objektebene konzentrieren – die Netzwerkkommunikation wird durch die Middleware realisiert.

3.2 CORBA

Mit der Object Management Group (OMG) hat sich ein internationales Konsortium von derzeit über 900 Firmen die Spezifikation eines Standards für verteilte, objektorientierte Anwendungen zum Ziel gesetzt. In diesem Abschnitt wird die CORBA-Architektur (Common Object Request Broker Architecture) vorgestellt, und es wird skizziert, welche Dienste und Komponenten spezifiziert sind und wie verteilte, objektorientierte Anwendungen auf der Basis dieser Architektur entwickelt werden können [OMG97a, Sieg96].

Der CORBA-Ansatz stellt mit der Object Management Architecture (OMA) und dem Corba Object Request Broker (ORB) eine Infrastruktur zur Verfügung, innerhalb derer verteilte Objekte über standardisierte Schnittstellen miteinander kooperieren können. Die Schnittstellen von Objekten werden in der Corba Interface Definition Language (Corba IDL) spezifiziert. Diese rein deklarative Sprache beschreibt die Struktur und das Verhalten von Objekten und ist – dem Objekt-Paradigma folgend – unabhängig von deren Implementierung. Die technischen Details der Spezifikation und Implementierung von Corba-Objekten sind in [Sieg96, OMG97a] umfassend dargestellt.

Corba-Objekte können unabhängig voneinander entwickelt, in unterschiedlichen Programmiersprachen implementiert und auf verschiedenen Hard- und Software-Plattformen ablauffähig sein. Auch in diesen Fällen können Objekte miteinander kooperieren und eine gemeinsame Anwendung bilden. Diese Eigenschaft macht Corba insbesondere im Kontext heterogener Plattformen, wie sie bei großen Unternehmungen oft anzutreffen sind, attraktiv; dies ist darüber hinaus eine wichtige Voraussetzung, um Komponenten – etwa Business-Objekte – unterschiedlicher Hersteller zu komplexen Anwendungssystemen zu koppeln (siehe Abschnitt 4).

In der Corba-Architektur werden eine Vielzahl von Diensten spezifiziert, die von Corba-Anwendungen (Application Objects) wiederverwendet werden können. Um die Voraussetzungen für einen Markt für Dienste bzw. Komponenten zu schaffen, wird von der OMG ein offener Standard spezifiziert und publiziert. Die Standardisierungsaktivitäten der OMG begannen mit der Spezifikation einer Menge grundlegender Dienste, die als Corba Common Object Services (Corba COS) bezeichnet werden. Dabei werden die Schnittstellen der Dienste beschrieben, nicht aber deren Implementierung. Dies erlaubt es unterschiedlichen Herstellern, Implementierungen vorzuschlagen und sich mit ihren Produkten am Markt zu behaupten. Die Common Object Services stellen Dienste zur Verfügung, die für alle Corba-Anwendungen relevant sind und entsprechend genutzt werden können. Es gibt unter anderem Dienste, um Objekte persistent zu speichern (Persistence Service), um Beziehungen zwischen Objekten auf einem hohen Abstraktionsniveau verwalten zu können (Relationship Service), um relevante Ereignisse von Objekten verwalten zu können (Event Service) oder um Objekte anhand von Namen (Naming Service) oder bestimmter Eigenschaften (Trading Object Service) auffinden zu können. Die Entwicklung von Anwendungen auf der Basis von Corba ist effektiv, da grundlegende Funktionalität, die sonst für jede Anwendung neu programmiert wer-

den müsste, durch die Common Object Services zur Verfügung gestellt wird. Die Corba COS stellen eine wichtige Grundlage zur Entwicklung eigenständiger und miteinander kombinierbarer Software-Komponenten – insbesondere Business-Objekte – dar.

Nachdem die Spezifikation der Corba COS im wesentlichen als abgeschlossen bezeichnet werden kann, wendet sich die OMG der Spezifikation höherer Dienste zu, die auf der Basis der Common Object Services implementiert werden. Diese Dienste werden als Corba Facilities bezeichnet. Corba Facilities werden unterschieden in horizontale und vertikale Corba Facilities. Horizontale Corba Facilities stellen Dienste zur Verfügung, deren Nutzung nicht auf eine spezielle Anwendungsumgebung beschränkt ist. Ein Beispiel hierfür ist die OMG Workflow Management Facility [CoCr98]. Die von vertikalen Corba Facilities bereitgestellten Dienste sind jeweils einer speziellen Anwendungsdomäne zugeordnet. Hier sei CORBAMED genannt, eine Corba Facility, welche spezielle Dienste für Anwendungen aus dem Krankenhaus- und Gesundheitswesen zur Verfügung stellt. Corba-Anwendungen werden als Objekte modelliert und als Application Objects bezeichnet; sie verwenden im allgemeinen Corba Facilities und Corba Common Object Services.

4 Frameworks für Business-Objekte

Objekte können durch Schnittstellenbeschreibungen unabhängig voneinander entwickelt und anschließend zu komplexen Objektsystemen kombiniert werden. Ist dieses Vorgehen bei der Verwendung relativ einfach strukturierter Objekte adäquat, so können bei dieser Vorgehensweise bei der Entwicklung großer Anwendungssysteme Probleme auftauchen.

Werden komplexe Anwendungssysteme auf der Basis herkömmlicher Objekt-Technologie entwickelt, so hat man es nicht mehr mit schwer wartbarem “Spaghetti-Code” zu tun. Aufgrund von unabhängigen, aber einander auf komplexe Weise referenzierenden Objekten können sich Strukturen bilden, die ebenfalls komplexen Charakter besitzen und das Verständnis des Objektsystems und seine Wartung erschweren können. Diese Eigenschaft wird in [Suth97] plastisch dargestellt:

We don't have spaghetti code in object systems. We have little balls of rigatoni that stick together producing mush, the result of poor design of object components.

Daher erfordert die Kombination von komplexen Business-Objekten hoher Abstraktionsebenen zusätzliche Mechanismen. Diese beziehen sich auf anwendungsspezifische Eigenschaften der Objekte, die durch Regeln (business rules) und spezielle, durch die Anwendung definierte Bedingungen (conditions) spezifiziert werden können. Dies bedeutet, dass neben syntaktischen auch semantische Spezifikationen zur Wiederverwendbarkeit und Interoperabilität von Business-Objekten notwendig sind. Diese werden durch spezielle Rahmen zur Verfügung gestellt, die als *Frameworks* bezeichnet werden. Frameworks beschreiben, wie Business-Objekte strukturiert sein müssen, damit sie zu komplexen betrieblichen Anwendungssystemen zusammengefügt werden können. Sie definieren Muster, innerhalb derer Objekte kooperieren können (collaboration patterns) und erlauben somit die Entwicklung und Kombination von anwendungsnahen Komponenten relativ hoher Granularitätsstufen, d.h., von Business-Objekten.

Nachfolgend werden die zentralen Eigenschaften von Frameworks anhand zweier aktueller Ansätze zur Spezifikation und Entwicklung von Business-Objekten weiter konkretisiert. Zunächst werden die Aktivitäten der OMG bei der Spezifikation einer Business Object Facility vorgestellt; anschließend wird das von der Firma IBM entwickelte San Francisco Framework beschrieben, das eine Menge in Java implementierter Business-Objekte und ein Programmiermodell zu deren Erweiterung und Kombination zur Verfügung stellt.

4.1 OMG Business Object Facility

Das Konzept der Business-Objekte wurde zunächst von der Business Object Management Special Interest Group (BOMSIG) im Rahmen der OMG-Aktivitäten zur Spezifikation einer entsprechenden Corba Business Object Facility eingeführt; diese Gruppe nannte sich später in Business Object Domain Task Force (BODTF) um. Das Ziel der BODTF besteht – wie bereits oben angesprochen – darin, den Spalt zwischen softwaretechnischen Modellen und betriebswirtschaftlichen Modellen zu überbrücken und auf der Basis von Corba ein Framework zu spezifizieren, innerhalb dessen komplexe Anwendungssysteme unter Verwendung von Business-Objekten entwickelt werden können.

Die OMG veröffentlichte im Jahre 1996 einen “Request For Proposals” (RFP) zur Einreichung von Vorschlägen für Common Business Objects sowie für eine Business Object Facility [OMG96], die dort wie folgt definiert werden:

[Common Business Objects are] objects representing those business semantics that can be shown to be common across most businesses.

[A Business Object Facility defines] the infrastructure required to support business objects operating as cooperative application components in a distributed object environment.

Es gab eine Reihe von Vorschlägen, die nachfolgend in den Gremien der OMG diskutiert wurden und in einen Vorschlag gemündet sind, der aus der Beschreibung der Business Object Component Architecture [DaAT98], der Common Business Objects [NIIP98] sowie aus einer Interoperability Specification [EDSD98] besteht. Dieser Vorschlag wird nun in seinen wesentlichen konzeptionellen Eigenschaften vorgestellt.

Abbildung 1 ordnet die Business Object Facility und Common Business Objects in den Corba-Kontext ein [DaAT98]. Auf der obersten Ebene befinden sich die Business-Objekte, die die Anforderungen spezieller Anwendungen erfüllen und daher unternehmensspezifisch sind (Enterprise Specific Business Objects). Diese können Objekte unterschiedlicher Abstraktionsebenen verwenden: Common Business Objects, Objekte der Business Object Facility sowie Objekte, die durch die Corba Facilities bzw. die Corba COS bereitgestellt werden. Common Business Objects sind Objekte, die in vielen Unternehmen verwendet werden können, etwa in den Bereichen Fertigung, Finanzen oder in der Personalverwaltung. Zur Implementierung dieser Business-Objekte können Objekte der Business Object Facility verwendet werden. Diese Ebene stellt auf der einen Seite Mechanismen zur Beschreibung von Business-Objekten auf einem hohen, anwendungsnahe Abstraktionsniveau zur Verfügung, auf der anderen Seite wird die technische Infrastruktur bereitgestellt, die zur verteilten Ausführung von Objektsystemen in Corba-Umgebungen notwendig ist; sie fungiert daher als Schicht zwischen Objekten der Anwendungsschicht und der technischen Corba-Infrastruktur.

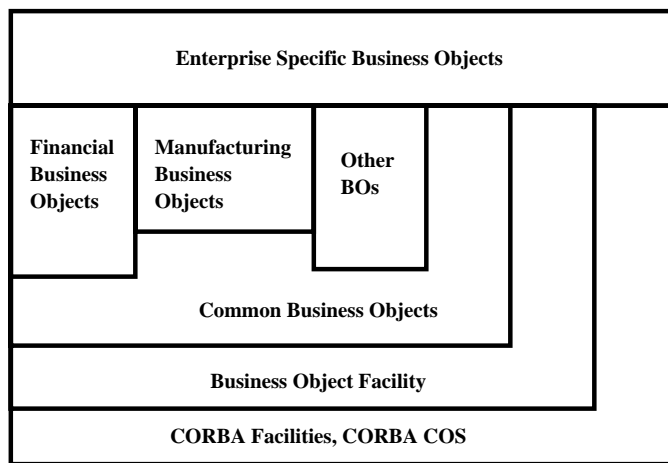


Abbildung 1: Einordnung der Business Object Facility und der Common Business Objects in den Corba-Kontext (in Anlehnung an [DaAT98]).

Die Struktur von Business-Objekten wird in der Business Object Architecture (BOCA) spezifiziert [DaAT98]; sie gibt den allgemeinen Rahmen vor, innerhalb dessen Business-Objekte entwickelt werden. Dabei spielt ein BOCA-Meta-Modell die zentrale Rolle; es beschreibt die Struktur von Anwendungssystemen, die aus Business-Objekten aufgebaut sind. Dabei können Business-Objekte unter Verwendung spezieller anwendungsnahe Konstrukte erstellt werden, die als BOCA-Komponenten (BOCA Components) bezeichnet werden und selbst wiederum Business-Objekte, aber auch einfacher Konstrukte sein können. An dieser Stelle wollen wir das BOCA-Meta-Modell nicht vollständig vorstellen, sondern lediglich die zentralen Komponenten und die wichtigsten Erweiterungen (BOCA Features) des Corba-Ansatzes besprechen. Business-Objekte sind Corba-Objekte und besitzen daher alle Eigenschaften dieser Objekte: Sie sind persistent, d.h., der Zustand von Business-Objekten wird auf einem persistenten Medium gespeichert, so dass auch nach Beendigung einer Anwendung der Zustand eines Business-Objekts erhalten bleibt und bei Bedarf unter Verwendung des Corba Persistency Service geladen werden kann. Business-Objekte besitzen einen eindeutigen Objektidentifikator, der sich während der Lebenszeit des Objekts nicht verändert; sie können unter Vermittlung eines Object Request Brokers Nachrichten an andere Objekte senden bzw. von anderen Objekten Nachrichten erhalten. Sie reagieren auf empfangene Nachrichten in Abhängigkeit ihres Zustands. Business-Objekte können sich auf unterschiedlichen Rechnern befinden, die durch ein Netzwerk miteinander verbunden sind; die Kommunikation erfolgt durch die Corba-Middleware.

Üblicherweise werden die Schnittstellen von Corba-Objekten – wie oben beschrieben – durch die Corba Interface Definition Language spezifiziert. Dabei wird auf syntaktischer Ebene beschrieben, wie Objekte aufgebaut sind und wie Methoden dieser Objekte aufgerufen werden können. Diese syntaktische Ebene ist zur Spezifikation von Business-Objekten nicht mächtig genug und somit zu deren Spezifikation ungeeignet, da bei Business-Objekten zusätzlich deren anwendungsspezifische Bedeutung, d.h. ihre Semantik im Kontext betrieblicher Anwendungen zu spezifizieren ist, die in IDL nicht ausgedrückt werden kann. Um die semantischen Eigenschaften von Business-Objekten

spezifizieren zu können, wurde die *Corba Component Definition Language (CDL)* entwickelt; sie erlaubt es, die Eigenschaften von Business-Objekten, die auf dem BOCA-Meta-Modell basieren, in textueller Form zu spezifizieren. Die Interoperabilitätsaspekte eines Business-Objekts werden durch die Übersetzung seiner CDL-Spezifikation nach IDL vorgenommen; das Augenmerk bei der Spezifikation von Business-Objekten kann daher auf die CDL-Ebene und damit auf die semantischen, anwendungsnahen Eigenschaften gerichtet sein; die Überführung auf die Ebene der Schnittstellenbeschreibung erfolgt durch eine Abbildung von CDL nach IDL unter Verwendung des BOCA-Meta-Modells.

Für Operationen von Business-Objekten können in CDL spezielle Eigenschaften definiert werden, die semantische Informationen von Business-Objekten charakterisieren, unter anderem Vor- und Nachbedingungen. Dies sind spezielle Bedingungen, die vor dem Start bzw. nach der Termination einer Operation für dieses Business-Objekt gelten müssen. Ist die Vorbedingung einer Operation nicht erfüllt, so wird eine Ausnahme generiert, die der Anwendung mitteilt, daß eine Operation aufgrund der Verletzung einer (als Vorbedingung definierten) Business-Regel nicht ausgeführt werden konnte. Damit können anwendungsrelevante Eigenschaften spezifiziert werden und müssen nicht in den Anwendungen programmiert werden. Diese Regeln spielen bei der Zusammenstellung von Business-Objekten zu komplexen Anwendungen eine wichtige Rolle, da semantische Eigenschaften der verwendeten Business-Objekte bei deren Kombination berücksichtigt werden können.

CDL erlaubt es mächtige, bei der objektorientierten Analyse definierte Eigenschaften bei der Beschreibung von Business-Objekten unmittelbar zu verwenden. Beziehungen zwischen Business-Objekten können in CDL mit ihren Kardinalitäten explizit spezifiziert werden. So kann durch eine 1..1-Beziehung zwischen Objekten der Klassen A und B spezifiziert werden, dass zu jedem Zeitpunkt jedes Objekt der Klasse A mit genau einem Objekt der Klasse B in Beziehung steht; 0..* beschreibt, dass jedes Objekt von A mit beliebig vielen Objekten von B in Beziehung stehen kann. Unter Verwendung des Rollenkonzepts können unterschiedliche Rollen für Objekte spezifiziert werden. So kann beispielsweise eine Firma in einer Anwendung sowohl als Produzent als auch als Lieferant und sogar als Käufer in Erscheinung treten. Diese verschiedenen Eigenschaften des Business-Objekts "Firma" können dann durch unterschiedliche Rollen-Objekte adäquat modelliert und implementiert werden.

Die Business Object Component Architecture stellt einen wichtigen Ansatz zur Entwicklung komponentenbasierter Anwendungssysteme dar. Mit ihr können Entwicklungszeiten von Anwendungssystemen reduziert und bestehende Systeme sehr viel schneller an neue Anforderungen angepasst werden. Um die Vorteile dieser neuen Technologie realisieren zu können, ist es allerdings notwendig, dass betriebliche Aufgaben unter Verwendung von objektorientierten Design- und Analysemethoden modelliert werden. Darüber hinaus sind die technischen Voraussetzungen zu schaffen, etwa die Integration von Alt-Anwendungen und von Datenbanksystemen durch Objektschnittstellen; damit erscheinen diese Systeme ebenfalls als Business-Objekte, deren Funktionalität dann von anderen Business-Objekten genutzt werden kann.

4.2 San Francisco Framework

Das San Francisco Framework wurde aus der Motivation heraus entwickelt, dass viele Firmen ihre Informationssysteme aufgrund von neuen Herausforderungen des Marktes anpassen und ihre Funktionalität entsprechend erweitern müssen [Bohr98a, Bohr98b, Bohr97]. Es ist allgemein anerkannt, dass solche Veränderungen in objektorientierten Anwendungssystemen leichter realisiert werden können als in traditionellen, monolithischen Systemen.

Der Einführung objektorientierter Techniken im betrieblichen Umfeld stehen allerdings eine Reihe praktischer Hindernisse gegenüber, etwa Kosten für die Aus- oder Weiterbildung des IT-Personals sowie die Risiken einer neuen Technologie. Das vorgeschlagene Framework soll den Einstieg in die objektorientierte, komponentenbasierte Anwendungsentwicklung erleichtern. Dies wird zum einen dadurch geleistet, dass eine objektorientierte Infrastruktur bereitgestellt wird, in der Business-Objekte bereits vorhanden und dokumentiert sind, die dann im Rahmen der Anwendungsentwicklung erweitert werden können, um die konkreten betrieblichen Anforderungen zu erfüllen.

Der Einstieg in objektorientierte Techniken wird auch durch die Wahl der Programmiersprache Java erleichtert. Entwickler mit Erfahrungen in der Programmiersprache C fällt der Umstieg auf Java erfahrungsgemäß recht leicht. Java stellt überdies ein hohes Maß an Plattformunabhängigkeit zur Verfügung, da Java-Programme in einen speziellen Byte-Code übersetzt werden, der dann auf unterschiedlichen Rechnerplattformen ohne Veränderung ausgeführt werden kann. Java hat mit dem Internet besondere Verbreitung gefunden, da kleine Java-Programme, sogenannte Applets, unter Verwendung eines Web-Browsers herunter geladen und lokal ausgeführt werden können.

Im San Francisco Framework werden keine fertigen Komponenten bereitgestellt, die zu Anwendungen verknüpft werden können. Stattdessen werden in Java implementierte Objekte und sogenannte *Design Patterns* zur Verfügung gestellt, d.h. wiederverwendbare und oft anzutreffende Muster, die bereits die allgemeinen Strukturen und das allgemeine Verhalten von Anwendungsobjekten besitzen, die aber erweitert werden müssen, um den spezifischen Anforderungen der jeweiligen Anwendung zu genügen. Dabei ist beabsichtigt, dass ein signifikanter Teil des Codes bereits durch das Framework geliefert werden, der übrige Code ist durch den Entwickler hinzuzufügen.

Abbildung 2 zeigt die Architektur des San Francisco Frameworks [Bohr98a]. Basierend auf der *Java-Virtual-Machine* wird eine Menge von Objekten spezifiziert, die sogenannte *Foundation*. Diese Ebene stellt Mechanismen zur Verfügung, mit denen Objekte höherer Abstraktionsebenen, d.h. Common Business Objects und Core Business Processes entwickelt werden können. Auf der Foundation-Ebene werden unterschiedliche Dienste zur Verfügung gestellt, die sich an den Corba COS orientieren und in Java implementiert sind. Es werden Dienste unter anderem in den Bereichen Persistenz (Persistence), Verwaltung von Beziehungen zwischen Objekten (Relationship) und eine Transaktionsverwaltung (Transactions) zur Verfügung gestellt.

Auf der nächsten Ebene befinden sich die Common Business Objects (CBO). Diese sind aufgeteilt in Business-Objekte, die üblicherweise in Anwendungen unterschiedlicher Domänen verwendet werden, z.B. Objekte für Adressen, Kunden und Währungen. Eine weitere Art von CBOs stellt Schnittstellen zu Objekten der oberen Ebene, d.h. zu Core Business Processes zur Verfügung. Damit kann Interoperabilität zwischen Business-Objekten unterschiedlicher Ebenen auf einfache Weise realisiert werden. Insbesondere

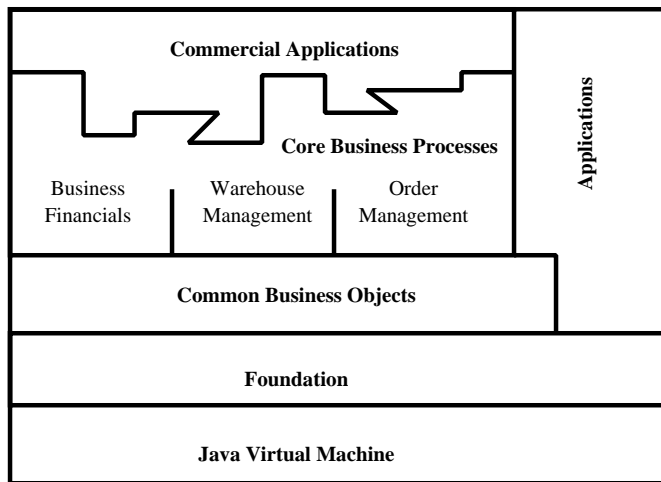


Abbildung 2: Architektur des San Framework Framework (in Anlehnung an [Bohr98a]).

erlauben diese Objekte (i) voneinander unabhängig entwickelten San Francisco Anwendungen miteinander zu kooperieren und (ii) die Interoperabilität von San Francisco Anwendungen mit Alt-Anwendungen. Letzteres wird durch entsprechende Schnittstellen erreicht, die Struktur und Verhalten der Objekte spezifizieren. Eine Alt-Anwendung kann beispielsweise durch Definition einer Schnittstelle eingebunden werden, welche die Struktur und das Verhalten eines Objekts beschreibt, das die Eintragung eines Wertes in ein bereits bestehendes Abrechnungssystem vornimmt. Ist diese Funktionalität implementiert, so kann sie unter Verwendung des Schnittstellen-Objekts dann von allen Business-Objekten der San Francisco-Anwendung verwendet werden.

Die Core Business Processes stellen Objekte zur Verfügung, die den Ausgangspunkt für die eigenen Entwicklungen bilden; jeder Core Business Process gruppiert eine Menge von fachlich miteinander in Beziehung stehenden Business-Objekten. Derzeit stehen Core Business Processes für die Domänen Finanzen (Business Financials), Auftragsverwaltung (Order Management) sowie Lagerhaltung (Warehouse Management) zur Verfügung. Es wird erwartet, dass etwa 40% der Funktionalität bereits durch das San Francisco Framework implementiert werden. Einen Großteil der darüber hinaus notwendigen Entwicklungsarbeit ist im Bereich der Benutzerschnittstelle zu erwarten. Dort kann allerdings ebenfalls auf Funktionen zurückgegriffen werden, die durch Java oder die Foundation des San Francisco Frameworks zur Verfügung gestellt werden.

Das San Francisco Framework stellt ein offenes System dar, und derzeit arbeiten eine Reihe namhafter Softwarefirmen an der Entwicklung von Programmierwerkzeugen für San Francisco, unter anderem IBM, Borland International, Rational Software und Symantec. Es stehen bereits Tools für die objektorientierte Analyse und das objektorientierte Design von Anwendungssystemen zur Verfügung, die in das Framework eingebettet sind und mit denen sich San Francisco Objekte erweitern lassen. Eine Anwendung des San Francisco Frameworks im Bereich Buchhaltung wird in [Inma98] ausführlich dargestellt; weitere Informationen stehen unter www.ibm.com/java/sanfrancisco zur Verfügung.

5 Diskussion und weitere Entwicklungen

Das San Francisco Framework stellt einen interessanten Ansatz zur Entwicklung von Business-Objekten auf der Basis von Java dar. Die vorgeschlagene Architektur weist eine starke Ähnlichkeit mit der hier vorgestellten Architektur der OMG auf. Durch die San Francisco Foundation wird überdies eine ähnliche Basisfunktionalität zur Verfügung gestellt wie durch Corba COS, um die Entwicklung von Business-Objekten effizient zu gestalten und ein hohes Maß an Wiederverwendbarkeit zu gewährleisten. Durch Design-Patterns wird die Struktur von Business-Objekten spezifiziert und deren Kombination zu komplexen Anwendungssystemen ermöglicht. Die Unterstützung des Frameworks durch namhafte Software-Firmen, und insbesondere die Entwicklung von Tools, die die Spezifikation anwendungsspezifischer Erweiterungen auf einem hohen Abstraktionsniveau erlauben, lassen für die Zukunft interessante Entwicklungen in diesem Bereich erwarten. Der Erfolg dieses Ansatzes wird von den zur Verfügung gestellten Objekten und deren Eignung in großen Produktivanwendungen abhängen.

Die OMG Business Object Facility stellt einen ambitionierten Ansatz zur Definition eines standardisierten Frameworks zur komponentenbasierten Entwicklung komplexer betrieblicher Anwendungssysteme dar. Dabei stehen alle in der verwendeten Corba-Umgebung vorhandenen Dienste zur Implementierung von Business-Objekten zur Verfügung. Dieser offene Standard bildet eine Voraussetzung dafür, dass unabhängig voneinander entwickelte Business-Objekte gemeinsam zur Lösung betrieblicher Problemstellungen eingesetzt werden können. Damit sind die Voraussetzungen zur Bildung eines Marktes für Business-Objekte geschaffen. Das große Interesse an der OMG Business Object Facility und die breite Unterstützung durch seine Mitglieder berechtigt zu einer optimistischen Einschätzung dieser neuen Technologie. Der Erfolg der OMG Business Object Facility wird allerdings davon abhängen, ob sich die Integration der Business-Objekte in Produktivanwendungen als flexibel, performant und ausfallsicher herausstellt.

Auch Anbieter betrieblicher Standardsoftware sehen in Business-Objekten ein großes Potential zur Entwicklung von Anwendungssystemen. SAP definiert beispielsweise ein Business Framework, worunter eine offene, komponentenbasierte Architektur verstanden wird, die die Interoperabilität von Software-Komponenten unterschiedlicher Hersteller erlauben soll [SAP97]. Zentrale Komponenten dieser Architektur bilden SAP Business-Objekte, die die Funktionalität zentraler SAP R/3-Komponenten über definierte Schnittstellen – sogenannte Business Application Programming Interfaces (BAPI) – zur Verfügung stellen. Die Implementierung von SAP Business-Objekten erfolgt somit nicht unter Verwendung von CORBA Common Object Services bzw. CORBA Common Business Objects, da bei SAP Business-Objekten lediglich existierende und in Form komplexer Softwaresysteme implementierte Funktionalität über entsprechende BAPI-Schnittstellen als Business-Objekte angesprochen werden kann.

In diesem Zusammenhang ist eine weitere Standardisierungsaktivität der OMG von Interesse: die der OMG Workflow Management Facility. Seit einigen Jahren wird Workflow-Management als eine wichtige Technologie zur Modellierung und kontrollierten Ausführung von Geschäftsprozessen angesehen [GeHS95, LeAl94]. Dabei werden die automatisierbaren Anteile von Geschäftsprozessen in einem Workflow-Schema modelliert, und ihre Ausführung als Workflow-Instanzen durch ein Workflow-Management-System kontrolliert. Da unterschiedliche Anwendungen von Workflow-Funktionalität profitieren

können, hat die OMG entschieden, einen Request for Proposals zur Definition einer Workflow Management Facility zu veröffentlichen [OMG97b]; es gab eine Reihe von Einreichungen, die in einen Vorschlag [CoCr98] gemündet sind. Dieser Vorschlag basiert auf dem Ansatz, Workflows als Corba-Objekte zu modellieren und damit Workflow-Funktionalität für Corba-Objekte, insbesondere für Business-Objekte zur Verfügung zu stellen [WHKS98]. Durch die gemeinsamen Schnittstellen von Business- und Workflow-Objekten (in Corba IDL beschrieben) wird die Integration von Business-Objekten in Workflow-Anwendungen unterstützt. Die Aktivitäten der OMG hinsichtlich der Definition einer Business Object Facility und einer Workflow Management Facility stellen wichtige, einander ergänzende Beiträge zur Förderung eines offenen Marktes für objektorientierte Business- und Workflow-Objekte und damit zur Entwicklung plattformunabhängiger, auf Objekten basierender Anwendungssysteme dar.

Literatur

- [Banc88] Bancilhon, F.: *Object-Oriented Database Systems*. In Proceedings of ACM SIGACT/SIGMOD Symposium on Principles of Database Systems, 152–162. ACM, New York 1988.
- [Bohr97] Bohrer, K.: *Middleware Isolates Business Logic*. Object Magazine 7(9) 1997.
- [Bohr98a] Bohrer, K., Johnson, V., Nilsson, A., Rubin, B.: *Business Process Components for Distributed Object Applications*. Communications of the ACM, 43–48, Vol 41 No 6, 1998.
- [Bohr98b] Bohrer, K.: *Architecture of the San Francisco Framework*. IBM Systems Journal Vol 37, No 2, 1998.
- [Casa95] Casanave, C.: *Business-Object Architectures and Standards*. Proceedings OOPSLA '95, Workshop on Business Objects Design and Implementation, 7–28. Springer, London 1995.
- [CoCr98] CoCreate Software, Concentus, CSE Systems, Data Access Technologies, Digital Equipment Corp., DSTC, EDS, FileNet Corp., Fujitsu Ltd., Hitachi Ltd., Genesis Development Corp., IBM Corp., ICL Enterprises, NIIP Consortium, Oracle Corp., Plexus - Division of BankTec, Siemens Nixdorf Informationssysteme, SSA, Xerox: *BODTF-RFP 2 Submission Workflow Management Facility (jFlow)*. OMG Document bom/98-06-07, 1998.
- [DaAT98] Data Access Technologies, Electronic Data Systems, NIIP, Sematech, Genesis Development Corp., Prism Technologies, Iona Technologies.: *OMG Business Object Domain Task Force BODTF-RFP 1 Submission: Combined Business Object Facility – Business Object Component Architecture* OMG Document bom/98-01-07, 1998.
- [EDSD98] Electronic Data Systems, Data Access Technologies, Genesis Development Corp., NIIP, System Software Associates, Inc.: *OMG Business Object Domain Task Force BODTF-RFP 1 Submission: Combined Business Object Facility – Interoperability Specification*. OMG Document bom/98-05-03, 1998.
- [FeSi90] Ferstl, O.K., Sinz, E.J.: *Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)*. In WIRTSCHAFTSINFORMATIK 32 (6), Vieweg, Braunschweig 1990.

- [FeSi91] Ferstl, O.K., Sinz, E.J.: *Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)*. In WIRTSCHAFTSINFORMATIK 33 (6), Vieweg, Braunschweig 1991.
- [FeSi97] Ferstl, O.K., Sinz, E.J., Hammel, C., Schlitt, M., Wolf, S.: *Application Objects: fachliche Bausteine für die Entwicklung komponentenbasierter Anwendungssysteme*. HMD – Theorie und Praxis der Wirtschaftsinformatik, Schwerpunktheft Componentware, 1997.
- [GeHS95] Georgakopoulos, D., Hornick, M., Sheth, A.: *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995.
- [Heue97] Heuer, A.: *Objektorientierte Datenbanken: Konzepte, Modelle, Systeme*. 2. Auflage. Addison Wesley, Bonn 1997.
- [HuSR98] Hung, K., Simons, T., Rose, T.: *“The Truth Is Out There?”: A Survey of Business Objects*. Accepted at Object-Oriented Information Systems (OOIS’98) La Sorbonne, Paris, September 1998.
- [Inma98] Inman, E.E.: *Enterprise Modeling Advantages of San Francisco for General Ledger Systems*. 170–180, IBM Systems Journal 37(2) 1998.
- [Lewa98] Lewandowski, S.M.: *Frameworks for Component-Based Client/Server Computing*. ACM Computing Surveys Vol 30 No 1, 3–27, 1998.
- [LeAl94] Leymann, F., Altenhuber, W.: *Managing Business Processes as an Information Resource*. IBM Systems Journal 33, 1994, 326–347.
- [NIIP98] NIIP Consortium: *Combined Business Objects – Revised Final Submission*. OMG Document bom/98-06-01, 1998.
- [OMG96] OMG. *Common Facilities RFP-4: Common Business Objects and Business Object Facility*. OMG Document CF/96-01-04, 1996.
- [OMG97a] OMG. *CorbaServices: Common Object Services Specification*. Juli 1997.
- [OMG97b] OMG: *Workflow Management Facility: Request for Proposals*. OMG Document cf/97-05-06, 1997.
- [SAP97] SAP AG: *BAPI Introduction and Overview*. Version R/3 Release 4.0. Dezember 1997. www.sap.com/bfw/interf/bapis/edu/docu/caalbe/caalbe.htm, Abruf am 1998-10-26.
- [Sieg96] Siegel, J.: *Corba – Fundamentals and Programming*. John Wiley, New York 1996.
- [Suth97] Sutherland, J.: *The Object Technology Architecture: Business Objects for Corporate Information Systems*. In Proceedings of the OOPSLA’95 Workshop on Business Object Design and Implementation. Springer, Berlin 1997.
- [WHKS98] Weske, M., Hündling, J., Kuropka, D., Schuschel, H.: *Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems*. Wird erscheinen in: Informatik Forschung und Entwicklung. Springer, Berlin 1998.