Fachgruppe 2.1.9: Objektorientierte Software-Entwicklung
Protokoll des 6. Fachgruppentreffens

München, 27.1.1999

Beginn: 10:00 *Ende:* 17:45

Protokoll: G. Engels, A. Zamperoni

Kurzbericht:

Auch das diesjährige Treffen der Fachgruppe 2.1.9 fand wieder im Rahmen der OOP in München statt. Es waren ca. 40 Personen anwesend, die wie auch schon in den letzten Jahren zur Hälfte aus der Industrie und zur Hälfte von der Hochschule kamen. Aufgrund des Einsatzes von Frau Paulisch (Mitglied der Fachgruppenleitung) und der freundlichen Unterstützung durch den Veranstalter SIGS Conferences stand auch in diesem Jahr wieder kostenlos ein Raum im neuen Konferenzzentrum ICM zur Verfügung.
Das Programm des Treffens bestand im wesentlichen aus der Präsentation aktueller Arbeiten von Mitgliedern der Fachgruppe und aus Berichten aus den einzelnen Arbeitskreisen. Insbesondere wurden die folgenden Vorträge präsentiert:

- Ansgar Schwegmann (Universität Münster): *Nutzen objektorientierter Konzepte bei der fachkonzeptuellen Modellierung betriebswirtschaftlicher Sachverhalte*
- Ulrich W. Eisenecker (FH Heidelberg), Krzysztof Czarnecki (DaimlerChrysler AG): *Objektsynthese*
- Josef Voss (Fernuniversität Hagen): *Framework-spezifische Modellierung im objektorientierten Entwurf*
- Frank Zeidler (agens Consulting GmbH): *Eine Komponentenarchitektur auf Grundlage eines Enterprise Application Frameworks*
- Mario Jeckle (DaimlerChrysler AG): *XML basierter Metadatenaustausch*
- D. Bienhaus (Universität-GH Kassel): *Entwurfsmuster-orientierter Ansatz zur einfacheren Realisierung Verteilter Systeme*

Die Zusammenfassungen der einzelnen Vorträge findet man am Ende dieses Protokolls.

Im Rahmen der Berichte aus den Arbeitskreisen haben die beiden Sprecher des Arbeitskreises Verteilte Objekte (AKVO), Christian Zeidler (ABB AG) und Fridtjof Toenniessen über ein zweijähriges Buchprojekt zum Thema *Erfahrungen mit Java* berichtet. Das Buch ist mittlerweile im dpunkt-Verlag erschienen. Nähere Angaben hierzu findet man unter http://www.dpunkt.de/produkte/java-erf.html.
In der Abschlußdiskussion wurde u.a. diskutiert, in welcher Form und wann die anstehenden Neuwahlen der Fachgruppenleitung durchzuführen sind. Nach kurzer Diskussion bestand Einigkeit darüber, daß keine Briefwahl durchgeführt werden sollte, sondern noch in diesem Jahr ein weiteres Fachgruppentreffen stattfinden sollte. Dieses Treffen wird im Rahmen der GI-Jahrestagung INFORMATIK'99 am 6. Oktober 1999 in Paderborn stattfinden. Als Vorbereitung für die Neuwahlen sind bereits jetzt alle Mitglieder der Fachgruppe aufgefordert, Kandidaten für die Fachgruppenleitung vorzuschlagen. Dies sollte schriftlich (am besten per email) an den Sprecher der Fachgruppenleitung Gregor Engels (engels@uni-paderborn.de) vor dem 30. April 1999 erfolgen. In der nächsten Ausgabe der Softwaretechnik-Trends wird dann eine Einladung zum nächsten Fachgruppentreffen erscheinen.

Zusammenfassungen der Vorträge:

Ansgar Schwegmann (Universität Münster): Nutzen objektorientierter Konzepte bei der fachkonzeptuellen Modellierung betriebswirtschaftlicher Sachverhalte. Für die Dokumentation von Informationssystemmodellen existieren zahlreiche Modellierungssprachen, die auf unterschiedlichen Paradigmen beruhen. Objektorientierten Modellen werden im Vergleich zu „klassischen" Datenmodellen, Prozeßmodellen, Datenflußdiagrammen etc. Eigenschaften wie eine hohe Wiederverwendbarkeit, Änderbarkeit und vor allem Verständlichkeit zugesprochen.

Ziel des Beitrags ist es zu untersuchen, inwieweit diese vermeintlichen Vorteile im Rahmen der fachkonzeptuellen Modellierung betriebswirtschaftlicher Sachverhalte zum Tragen kommen. Von besonderer Bedeutung für die fachkonzeptuelle objektorientierte Modellierung sind die Systemstrukturierung durch Klassen und das Konzept der Vererbung. Bei der Beschreibung eines Systems wird im allgemeinen zwischen der Struktur und dem Verhalten eines Systems unterschieden. Um den Nutzen einer objektorientierten Informationssystem- bzw. Anwendungssystemmodellierung explizieren zu können, wird eine Differenzierung der Begriffe „Struktur" und „Verhalten" in Mikroverhalten/-struktur und Makroverhalten/-struktur motiviert. Mikrostruktur bezeichnet die innere Struktur eines Systemelements bzw. im Kontext der objektorientierten Modellierung einer Klasse. Strukturelle Beziehungen der Systemelemente untereinander determinieren die Makrostruktur des Systems. Das Mikroverhalten definiert das Verhalten eines einzelnen Systemelements in Bezug auf Ereignisse von außen. Auch die Interaktionen weniger, in einer Abhängigkeitsbeziehung stehender Systemelemente werden dem Mikroverhalten zugerechnet. Das Makroverhalten umfaßt die Interaktion unabhängiger Systemelemente und beinhaltet die komplexen Abläufe eines Systems.

Auf Basis dieser Differenzierung wird gezeigt, daß Klassen lediglich geeignet sind, das Mikroverhalten bzw. die Mikrostruktur von Systemelementen zu kapseln. Diese Kapselung führt nur unter bestimmten Bedingungen zu semantisch reichhaltigeren Modellen, welche sich im Vergleich zu „klassischen" ER-Modellen durch eine höhere Verständlichkeit und Anpaßbarkeit auszeichnen. Darüber hinaus sind Klassen zu feingranular und nur begrenzt geeignet, um eine komplexitätsreduzierende Wirkung umfangreicher Informationssystemmodelle zu erzielen. Folglich ist auch der Nutzen des Konzepts der Vererbung begrenzt, da sich der Vererbungsmechanismus nur auf genau eine Klasse bezieht. Die Ausführungen werden am Beispiel eines Modells der Problemdomäne Lagerverwaltung illustriert.

Ulrich W. Eisenecker (FH Heidelberg), Krzysztof Czarnecki (DaimlerChrysler AG): Objektsynthese. Die gegenwärtig verfügbaren Techniken der objektorientierten Softwareentwicklung unterstützen die Wiederverwendung und die Konfigurierbarkeit in unzureichendem Maß. Die aus der Domänenanalyse stammende Modellierung von Merkmalen und generative Implementierungstechniken können in Ergänzung objektorientierter Analyse- und Designmethoden eingesetzt werden, um dieses Problem zu beheben. Die Modellierung von Merkmalen erlaubt es, die Vielfalt der Objektvarianten einer Domäne zu beschreiben. Konkrete Objekttypen können dann aus abstrakten Spezifikationen synthetisiert werden.

Anhand eines einfachen Beispiels einer konfigurierbaren Liste werden die Modellierung von Merkmalen und die Implementierung eines Generators für spezifische Konfigurationen dieser Merkmale beschrieben. Dieser Vorgehensweise liegt eine Trennung von Problemraum und Implementierungsraum zugrunde. Der Anwender beschreibt die Anforderungen in einer anwendungsnahen Sprache, während die Implementierungskomponenten in implementierungsspezifischen Sprachen beschrieben und miteinander verbunden werden. Im Rahmen der Objektsynthese bildet der Generator somit die Spezifikation in anwendungsnaher Sprache auf die Verbindung von Implementierungskomponenten ab. Es wird gezeigt, wie Generatoren und sogenannte Konfigurationsrepositories - sie enthalten die endgültigen Merkmale eines Objekttyps - mit objektorientierten, generischen und generativen Sprachmitteln implementiert werden können. Der Generator ist als C++-Template-Metaprogramm realisiert, welches zur Übersetzungszeit ausgeführt wird.

Josef Voss (Fernuniversität Hagen): Framework-spezifische Modellierung im objektorientierten Entwurf. Frameworks sind aus der objektorientierten Softwareentwickung nicht mehr wegzudenken. Ausgereifte und wohlverstandene Frameworks tragen wesentlich zur hohen Produktivität objektorientierter Entwicklung bei. Als Hindernis für den weitverbreiteten Einsatz wird neben der grundsätzlichen Schwierigkeit, gute Frameworks zu entwickeln, der vergleichsweise hohe Einarbeitungsaufwand angesehen.

Auf dieses Problem zielen Ansätze für eine verbesserte Framework-Dokumentation auf der Basis von Entwurfsmustern [1] sowie Ansätze zu einer aktiven Unterstützung bei der Framework-Anwendung, sogenannte active Cookbooks [2]. Allerdings richten sich diese Maßnahmen in erster Linie an Entwickler, die ein Framework kennenlernen oder erstmals einsetzen wollen. Ein Entwickler, der das Framework, mit dem er eine Applikation entwickelt, gut kennt, stellt andere Anforderungen an eine geeignete Unterstützung. Insbesondere sollte eine Unterstützung schon im Software-Entwurf greifen und dort in geeigneter Form in Klassendiagramme, dem zentralen Hilfsmittel im objektorientierten Entwurf, integriert sein. In bezug auf Frameworks sind Klassendiagramme bislang hauptsächlich zur Beschreibung der Frameworks selbst eingesetzt worden, etwa zur Dokumentation der eingesetzten Entwurfsmuster. Zur Darstellung des objektorientierten Entwurfs eines Software-Systems stehen zunächst nur universelle Sprachmittel, insbesondere die von UML, zur Verfügung. Dabei ist aber gerade das wiederverwendbare Design einer der wesentlichen Werte eines Frameworks, und die aus der Verwendung des Frameworks resultierenden Komponenten und Strukturen sollten sich unmittelbar in Klassendiagrammen widerspiegeln, etwa in Form von Framework-spezifischen Konstrukten.

Wir stellen einen Ansatz vor, der die skizzierte Idee der Framework-spezifischen Klassendiagramme an hand eines konkreten Frameworks realisiert. Die dazu entwickelten Konstrukte sind zum Teil in UML schon vorgesehen, etwa Stereotypen, die hier in Form von Framework-spezifischen Stereotypen verwendet werden. Darüber hinaus werden aber auch Erweiterungen eingeführt, insbesondere komplexe Spezifikationen und symbolische Beziehungen. Zunächst wird kurz das als Fallbeispiel verwendete FLUID Extended GUI Framework beschrieben. Die Entwicklung dieses Frameworks ist wesentlich durch die Beobachtung motiviert, daß gängige GUI Frameworks keine ausreichende Unterstützung für große Applikationen mit vielen Fenstern und entsprechend umfangreicher Funktionalität bieten. Schließlich wird auf einige Aspekte der entwickelten Modellierungswerkzeuge eingegangen. Diese unterstützen neben der reinen Notation flexible Übergänge zwischen Analyse, Entwurf und Codierung (Roundtrip Engineering). Für die Framework-spezifischen Klassendiagramme wird die Anbindung an Analysemodelle auf der einen und an die Codierung auf der anderen Seite betrachtet, wobei insbesondere gezeigt wird, wie Detailkenntnisse über das zugrundeliegende Framework für eine effektive Entwicklungsunterstützung ausgenutzt werden.

Literatur

1. Johnson, R., *Documenting Frameworks Using Patterns*. Proceedings OOPSLA '92, ACM SIGPLAN Notices 27, 10 (Oktober 1992), 63-76.
2. Schappert, A., Sommerlad, P. und Pree, W., *Automated Support for Software Development with Frameworks*. Symposium on Software Reusability 1995, 123-127.

Frank Zeidler (agens Consulting GmbH): Eine Komponentenarchitektur auf Grundlage eines Enterprise Application Frameworks. Durch diesen Beitrag soll ein Vorschlag aufgezeigt werden, wie verschiedene Modellierungsansätze im Business-Engineering sowie im Software-Engineering vereinheitlicht werden können. Betriebliche Organisationen und die verwendeten Anwendungssysteme werden durch ein zentrales und homogenes Modell spezifiziert. Es wird in der Praxis nur noch ein System modelliert und zur Ausführung gebracht, anstatt wie bisher zwei grundlegend verschiedene Systeme. Das zentrale Organisationsmodell, bestehend aus Class- und State-Charts, wird durch ein Organisations-Managementsystem (OrgMS), basierend auf einem „Enterprise application framework", modelliert und ausgeführt. Ein solcher Framework bildet die Infrastruktur zur Konstruktion, Ausführung und Wiederverwendung von Komponenten innerhalb des Organisationsmodells.

Es ergeben sich im wesentlichen zwei Vorteile: Erstens erhalten die Betriebsorganisatoren und Fachabteilungen eine formale und unmißverständliche Modellierungssprache, die auch von der DV-Abteilung verstanden wird. Die Fachabteilungen und die Betriebsorganisation müssen sich allerdings darauf einstellen, wesentlich detailliertere Organisationsmodelle zu entwerfen, als die bisherigen Organigramme und Flußdiagramme zur Ablaufdarstellung. Der zweite Vorteil ist, daß das Organisationsmodell, entworfen mit diesem Formalismus, ausgeführt werden kann. Dies geschieht durch ein OrgMS. Ein solches OrgMS ist im wesentlichen eine Vereinigung der Konzepte einer GUI-Entwicklungsumgebung und von Workflowmanagementsystemen. Wie bei den GUI-Entwicklungsumgebungen wird eine Klassenhierarchie mit Komponenten aufgebaut, allerdings auf einem höheren Abstraktionsgrad. Die Entwurfssprache ist auf dieser Abstraktionsebene nicht mehr eine Programmiersprache (Java oder Object-Pascal), sondern die Modellierungssprache bestehend aus Class- und State-Charts. Die DV-Abteilung wird zukünftig keine ganzen EDV-Systeme mehr programmieren, sondern zum einen in Einzelfällen die vorgegebene IT-Architektur des „Enterprise Application Frameworks" anpassen und zum anderen hauptsächlich konkrete Aktivitäten, gefordert von den Fachabteilungen, implementieren. Die DV-Abteilung wird somit besser in die Unternehmensorganisation integriert und auch wesentlich durchschaubarer. Software-Engineering und Business-Engineering laufen wesentlich verzahnter ab.

Mario Jeckle (DaimlerChrysler AG): XML basierter Metadatenaustausch. Durch die Entwicklung der „Object Analysis and Design" (OA&D) Modellierungssprache „Unified Modeling Language"

(UML) [1], welche die existierenden verschiedensten proprietären - aber zunehmend konfluenten - objektorientierten Entwurfsmethoden vereinigt, sollten die „Methodenkriege" der Vergangenheit angehören. Zwar wurde der ursprüngliche Anspruch der Vereinigung und Standardisierung von sowohl Notation als auch Entwufsmethodik (Prozeßmodell) bald auf die Zusammenführung der rein graphischen Modellierungssprachenelemente sowie der zugrundeliegenden Semantik beschränkt. Jedoch wurde mit der verabschiedeten Version 1.1 des UML-Metamodells und der Definition der graphischen Konstrukte der Sprache ein inzwischen verbreiteter Industriestandard geschaffen der - trotz einiger Schwächen - hinreichend expressive Konstrukte zur Definition verschiedenster objektorientierter Architekturen bietet. Die zeitlich kurz vor der UML verabschiedete „Meta Object Facility" (MOF) [2] bildet deren konzeptionelle Grundlage. Dies bedeutet das Meta-Modell der UML bildet als Instanz des in der MOF normierten Meta-Metamodells die dort definierten objektorientierten Konzepte ab.

Als Resultat der Focusierung auf die rein graphische Sprache wurden zusätzliche Aspekte wie Austausch der Modelldaten (Format- oder „streaming-"Problematik) sowie eine gleichmächtige textuelle Alternativdarstellung der entstehenden Modelle vollkommen außer Acht gelassen. Zwar wurde die UML mittlerweile durch eine einfache formale nichtgraphische Sprache zur Formulierung einschränkender Integritätsbedingungen, die „Object Constraint Language" (OCL), erweitert, welche jedoch keinerlei Ansdtze zur textuellen Darstellung der durch die formulierten Constraints präzisierten oder beschränkten Modelldaten bietet.

Der Problembereich eines generischen, herstellerneutralen Industriestandardformats zum Austausch beliebiger objektorientierter Modelle wird durch den Request for Proposal (RFP) für ein „Stream-based Model Interchange Format" (SMIF) der OMG adressiert [3]. Konkretes Ziel dieser Normierungsbemühungen ist die Schaffung einer auf dem MOF Meta-Metamodell basierenden textuellen Darstellung, die den Austausch einer Vielzahl heterogener Metamodellinstanzen, d.h. objektorientierte Modelle verschiedenster Notationen, ermöglicht.

Ähnlich wie der Aufruf zur Einreichung von Vorschädgen zur OA&D-Notation stieß der SMIF-RFP auf ein breites Interesse der Industrie und industrienahen Forschung. Neben vornehmlich im Toolgeschäft engagierten Firmen beteiligen sich auch breiter angelegte Unternehmen an den Normungsbemühungen. Die industrienahe Forschung wird unter anderem durch die Universitat Politecnica de Catalunya und das DaimlerChrysler Forschungszentrum (Ulm) vertreten.

Nach der initialen Einbringung dreier, inhaltlich verschiedene Akzente setzender, Vorschädge [4, 5, 6] einigten sich alle verbliebenen an der Normung Interessierten auf die Zusammenführung der Bemühungen und die Erarbeitung eines gemeinsamen Vorschlages auf der Grundlage der „Extensible Markup Language" (XML) - das „XML Metadata Interchange" Format (XMI) [7]. Technisch gesehen handelt es sich bei XMI um Bildungsvorschriften für die XML zugrundeliegenden Grammatiken („Document Type Definition" (DTD)) sowie zwei Beispiel-DTDs für MOF-basierte Meta-Metamodelle bzw. UML-basierte Modelle.

1. Object Management Group: *UML Specification V1.1*, Framingham (USA), 1997, document ad/97-08-02
2. Object Management Group: *MOF Specification V1.1*, Framingham (USA), 1997, document ad/97-08-14
3. Object Management Group: *Stream-based Model Interchange Format; Request For Proposal*, Framingham (USA), 1998, document ad/97-12-03
4. CDIF Partners: *Stream-based Model Interchange Format*, Framingham (USA), 1998, document ad/98-07-09
5. UOL Partners: *Universal Object Language (UOL)*, Framingham (USA), 1998, document ad/98-07-07
6. XMI Partners: *XML Metadata Interchange (XMI)*, Framingham (USA), 1998, document ad/98-07-01
7. World Wide Web Consortium: *Extensible Markup Language (XML) 1.0*, REC-xml-199808210

D. Bienhaus (Universität-GH Kassel): Entwurfsmuster-orientierter Ansatz zur einfacheren Realisierung Verteilter Systeme.

1. Überblick und Problemstellung

Herausforderungen bei der Entwicklung von verteilten oder zumindest entkoppelten Anwendungssystemen bestehen einerseits darin, neue Anwendungssysteme so zu entwerfen und zu implementieren, daß sie mit Hilfe von middleware als verteilte Applikationen ausgeführt werden können, und andererseits im Redesign bestehender, sog. „Alt"-Systeme (legacy systems).

Der Entwurf neuer und das Redesign bestehender Applikationen sind nur von erfahrenen Entwicklern und Software-Designern zu bewältigen. Gewisse Lösungswege und Vorgehensweisen treten beim Entwurf jedoch immer wieder in Erscheinung. Solche grundlegenden Lösungswege so zu beschreiben, daß

Entwurfsinformationen wiederverwendbar sind, ist die Zielsetzung von Software-Entwurfsmustern (software design pattern, design pattern).

Durch eine relativ formale Darstellung erlauben Entwurfsmuster die Darstellung von Design-Lösungen auf einem hohen Abstraktionsniveau.

Entwurfsmuster werden nicht in einer Programmiersprache repräsentiert. Entwickler müssen selber die abstrakte Darstellung der Entwurfsmuster für ihr konkretes Problem anpassen und in Code umsetzen. Einerseits ist dies für den erfahrenen Entwickler eine eher monotone Arbeit. Andererseits kann dieses Umsetzen für den im Umgang mit Entwurfsmustern Unerfahrenen eine abschreckende Hürde darstellen. Diese Hürde wird fast unüberwindbar, wenn es sich um das Redesign einer (meist prozedural entworfenen) Alt-Applikation handelt und das objektorientierte Paradigma für die Beteiligten noch neu ist.

2. Übersicht des Lösungsansatzes

Der Lösungsansatz (vorgestellt auf der CUC '98. Handouts sind verfügbar unter:
cuc.ifa.ch/handouts/BienhausCUC98.zip und
www.teleport.com/Lambdagoldman/jell/
handouts98.html) vereinfacht die Realisierung Verteilter Systeme in der Design- (bzw. Redesign-)Phase und bei der Implementierung. Eckpunkte sind:

Einführung von Kopplungsobjekten
Kopplungsobjekte kapseln verteilungsspezifische Implementierungsdetails, die bei verteilten Systemen gegenüber monolithischen Applikationen aufwendiger sind. Kopplungsobjekte sind Objekte, die in einem verteilten Objektsystem aufbauend auf eine objektorientierte Integrationsschicht die Interaktion zwischen den Teilsystemen bzw. -applikationen übernehmen.

Zusammen mit einer (objektorientierten) Integrationsschicht bilden Kopplungsobjekte ein Framework. In diesem nehmen sie die Rolle von aktiven (grobgranularen) Schnittstellen ein; sie sind aktive ,,Andockpunkte".

Ausgangspunkt
Ausgangspunkt für die Generierung von Kopplungsobjekten ist die Spezifikation von Datentypen und strukturen für die Informationen, die im betrachteten Anwendungsbereich über die Grenzen der Teilsysteme ausgetauscht werden (,,Austauschattribute"). Zusätzlich erfolgt eine Festlegung anwendungsspezifischer Ausnahmetypen und deren Zuordnung zu den Austauschattributen. Aus dem (UML-)Modell der Austauschattribute erfolgt die Ableitung der Kopplungsobjekte.

Repräsentation
Die einzelnen Schritte von der Spezifikation der Anwendungsdaten und Ausnahmetypen bis zur Erzeugung von Bibliotheken (bzw. Frameworks) sind als Entwurfsmuster strukturiert. Dadurch sind die Einzelschritte leichter nachzuvollziehen und ihre Umsetzung kann prinzipiell automatisiert erfolgen.

Anwendung
Der Lösungsansatz unterstützt die Realiserung Verteilter Systeme zusätzlich, indem eine graphische Notation (Kollaborationsdiagramme, erweitert um die Darstellung von Prozeßgrenzen und Datenflußrichtungen) die Umsetzung von Verteilungs-Entwurfsmustern mittels Kopplungsobjekten skizziert.

Mustersprache
Die Repräsentation des gesamten Lösungsansatzes erfolgt in Form von eng zusammenhängenden Entwurfsmustern; sie bilden zusammen eine Mustersprache.

3. Erfahrungen

Dieser Lösungsansatz wurde bei der Realisierung einer 3D-Simulationsumgebung als Verteiltes System eingesetzt. Hardware-Komponenten und 3D-Visualisierungsprogramme kommunizieren über Kopplungsobjekte. Durch den Einsatz von Kopplungsobjekten konnten Studierende leichter ,,logische" Komponenten implementieren, ohne sich in die CORBA-Programmierung einarbeiten zu müssen. Weiterhin konnten ,,Altapplikationen", wie z.B. bestehende, prozedurale VR-Libraries und Demo-Programme von Positionserfassungssystemen, leicht in die heterogene Rechnerumgebung integriert werden. Die Architektur der 3D-Simulationsumgebung basiert auf einem Entwurfsmuster-orientierten Design. Die Dokumentation des Systems und Einarbeitung neuer Entwickler wurde dadurch vereinfacht.

gez. G. Engels, A. Zamperoni