Benutzung und Implementierung von Middleware am Beispiel von CORBA im Rahmen von Informatik-Hochschulpraktika

Gerd Aschemann[1] Christian Becker[2] Kurt Geihs[2] Friedemann Mattern[1]

[1] TU Darmstadt, FB Informatik, FG Verteilte Systeme, Aschemann@Informatik.TU-Darmstadt.de

[2] Uni Frankfurt, FB Informatik, FG Verteilte Systeme und Betriebssysteme, Becker@Informatik.Uni-Frankfurt.de

28. Januar 1999

Sinn und Inhalte von Praktika im Rahmen des Informatik-Studiums sind regelmäßig Thema fachlicher und didaktischer Diskussionen. Eine fachliche Grundfrage ist dabei, ob eher die Benutzung eines entsprechenden Systems im Vordergrund stehen sollte oder die Interna, sprich die Implementierung. Eine wichtige didaktische Frage ist, ob man eher vom Abstrakten zum Konkreten hin vorgehen solle (top-down) oder von den Grundlagen zu den höheren Schichten (bottom-up).

Die Autoren verfügen über langjährige Erfahrungen mit verschiedenen Praktikums-Modellen im Bereich Verteilte Systeme. In der Regel wurden mit den verschiedenen Ansätzen gute Erfahrungen gemacht, es blieben aber auch immer wieder Kritikpunkte offen, die zu Änderungen in den Folgepraktika geführt hatten. Grundgedanke war immer, das Praktikum nicht einseitig nur auf Benutzung oder Implementierung verteilter Systeme zu beschränken. Es sollte den Teilnehmern durch eigene Erfahrung vermittelt werden, daß bestimmte Abstraktionen aus Sicht eines Benutzers (der jeweiligen Schnittstellen) sinnvoll sind. Gleichzeitig sollte klar werden, daß durch die Abstraktion auch Möglichkeiten der Einflußnahme auf das Systemverhalten verloren gehen, z.B. hinsichtlich des Kommunikationsmodells (synchron vs. asynchron, gepuffert vs. ungepuffert). Außer Frage stand immer, daß die verwendeten Systeme und Schnittstellen immer nur als Beispiel für gelungene (oder auch weniger gelungene) Umsetzungen allgemeiner Entwurfsprinzipien für verteilte Systeme dienen sollten und auch immer kritisch hinterfragt werden sollten.

Eine weiteres Problem bei der Konzeption von Praktika ist die Frage, ob man verschiedene Einsatzbereiche verteilter Systeme abdecken sollte, oder besser anhand eines "großen Beispiels" alle Möglichkeiten eines Systems ausloten sollte. An der Uni Frankfurt wurde dabei mehrfach versucht, das Distributed Computing Environment (DCE) in den Mittelpunkt zu stellen. Hier stellte sich als zusätzliche Schwierigkeit heraus, daß auch die Administration eines DCE-Systems während eines Praktikums enorme Anforderungen an den Veranstalter bedeutet. Die Praktika an der TU Darmstadt waren eher auf ein breiteres Spektrum hin angelegt [AMF97], bei dem auch verschiedene (zum jeweiligen Zeitpunkt) aktuelle Technologien (Web, Java/RMI, CORBA, PVM, ...) berücksichtigt werden sollten. Üblicherweise sind derartige Praktika auf die Dauer eines Semesters hin angelegt und die Teilnehmer sind in der Regel zeitlich gut ausgelastet und können nur wenige andere Veranstaltungen neben dem Praktikum wahrnehmen. Außerdem muß man bei der Konzeption berücksichtigen, daß zumindest einige Teilnehmer in einem derartigen Praktikum zum ersten Mal in einem größeren Projekt programmieren müssen und die entsprechende Einarbeitung auch einige Zeit kostet. Daher muß sorgfältig abgewägt werden, was man den Studierenden zumuten kann. Der Wechsel zwischen verschiedenen Paradigmen, Implementierungsebenen und gar Programmiersprachen führt zu einer hohen Belastung.

Mit dem Aufkommen und der Verbreitung der Common Request Broker Architecture (CORBA) bot es sich an, auch diese Technologie als Beispiel für Verteilte Systeme in die Praktika einzubeziehen. Da CORBA ein offener Standard (gesamte Spezifikation ist "frei" verfügbar, [OMG98a]) ist, kam in diesem Zusammenhang die Idee auf, ein CORBA-System für den Einsatz in der Lehre zu implementieren [Pud97], was schließlich in der MICO-Implementierung gipfelte [PR99]. Die Idee wurde schon frühzeitig zwischen Frankfurt und Darmstadt diskutiert und im Wintersemester 1997/98 wurde erstmals ein entsprechendes Praktikum in Frankfurt veranstaltet [Bec97, Gei98] mit der Option, bei nächster Gelegenheit in Darmstadt die Erfahrungen aufzugreifen und das Praktikum weiterzuentwickeln [Asc98].

Die Verwendung von CORBA in einem Praktikum bietet die Möglichkeit, verschiedene der oben skizzierten Pro-

bleme zu lösen:

- Man kann sowohl den Einsatz, als auch die (partielle) Implementierung eines CORBA-Systems zum Gegenstand des Praktikums machen.
- Man kann sowohl top-down, als auch bottom-up vorgehen.
- Man kann viele wichtige Prinzipien beim Entwurf und der Implementierung verteilter Systeme im Rahmen einer einzigen Architektur durchgehen und anhand eigener Erfahrungen vermitteln.
- CORBA liegt als offener Standard vor und ist sehr einfach zugreifbar. Der Standard ist zwar umfangreich, aber bei entsprechender Anleitung durchaus auch für Studierende erfaßbar und verstehbar. Während der Praktika hat es sich ergeben, daß man die wichtigen Teile des Standards nicht auf einmal erfassen und verstehen muß, sondern sie sich nach und nach erarbeiten kann.
- Durch die mittlerweile zahlreich verfügbare Sekundärliteratur, freie Implementierungen und Diskussionen in Mailinglisten und News-Groups, an der sich auch Entwickler und Verfasser von Standards beteiligen, sind Designentscheidungen dokumentiert und nachvollziehbar oder zumindest "nachfragbar". Auch das bei vielen Lehrveranstaltungen häufige Problem adäquater Einführungstexte und Glossare stellt sich mittlerweile nicht mehr.

Sowohl in Frankfurt, wie auch in Darmstadt, wurde das Praktikum in vier Aufgaben unterteilt, die jeweils in drei bis vier Wochen gelöst werden sollten. In Frankfurt gab es dabei eine klare Trennung: Die ersten beiden Aufgaben sollten in die Benutzung von CORBA einführen, die nachfolgenden beiden Aufgaben die Implementierung eines (einfachen) CORBA-Systems zum Gegenstand haben (MiniCORBA). Dabei konnte eine verteilte Implementierung nicht realisiert werden. Daher wurde im zweiten Anlauf – in Darmstadt – der erste Teil auf eine einzige Aufgabe reduziert und der zweite Teil auf drei Aufgaben ausgedehnt, mit dem Ziel, am Schluß ein verteiltes MiniCORBA zu erstellen. Gleichzeitig wurden in Darmstadt in geringem Umfang andere Konzepte der Benutzung von CORBA verstärkt in das Praktikum aufgenommen (Factory-Pattern, Callbacks/geschachtelte Aufrufe) und andere (eigene Programmierung von DII/DSI) gekürzt.

Im einzelnen hatten die Aufgaben folgende Inhalte:

- Zunächst sollte an einem einfachen Beispiel die Benutzung von CORBA kennengelernt werden: Definition einer Schnittstelle in IDL, Umgang mit IDL-Compiler und anderen Tools, Implementierung der Applikationsfunktionalität ("Server") unter Berücksichtigung von CORBA-Eigenheiten (insbesondere Speicherverwaltung im Zusammenhang mit C++), Implementierung eines "Clients".
- In Frankfurt gab es eine zweite Aufgabe zur Erkundung der verschiedenen Aufruf-Modi des – inzwischen aus dem Standard entfernten, aber immer noch weit verbreiteten – Basic Object Adapters (BOA, [OMG97]).
- Der (verteilte) Aufrufpfad durch die verschiedenen Komponenten (selbstimplementierte Aufrufe und Methoden, generierte Stubs und Skeletons, ORB-Implementierung) der Applikation aus der ersten Aufgabe sollte untersucht und dokumentiert werden. In Frankfurt sollten dabei zusätzlich eigene dynamische Aufrufe (DII) bzw. deren Auflösung (DSI) implementiert werden. In Darmstadt (zweite Aufgabe) reichte das schrittweise Verfolgen mittels Debugger.
- Die vierte (in Darmstadt dritte) Aufgabe gab die Schnittstellen für einen simplen ORB vor, bei dem die von der OMG vorgesehenen Schnittstellen und Konzepte entsprechend eingeschränkt wurden und eine einfache Applikation vorgegeben war, die mit dem ORB ausgeführt werden sollte: nur wenige, skalare Datentypen, keine Aktivierungsmodi. Nichtsdestotrotz mußten bei der Implementierung des Mini-ORBs alle wesentlichen Konzepte einer vollständigen (lokalen) Implementierung für die Vermittlung von Aufrufen innerhalb eines Adreßraums umgesetzt werden.
- In Darmstadt gab es noch eine weitere Aufgabe, die darin bestand, den in der vorherigen Aufgabe implementierten lokalen ORB zu einer verteilten Implementierung, über Prozeß- bzw. Rechnergrenzen hinweg, auszubauen. Die Problematik potentiell verschiedener Datendarstellungen auf verschiedenen Rechnerarchitekturen und deren Vereinheitlichung durfte dabei ausgelassen werden.

Beide Praktika beruhen auf der MICO-Implementierung (C++) bzw. den in MICO vorgesehenen Konzepten (Micro-ORB mit verallgemeinerter Schnittstelle zu Objekt-Adaptern [Röm98]). In Darmstadt wurden die dritte und vierte Aufgabe in Java implementiert[1]. Dabei konnte noch speziell auf die besonderen Anforderungen aus der Abbildung von CORBA-IDL auf Java (language mapping, [OMG98b]) eingegangen werden: Um die zur Implementierungszeit generierten Stubs und Skeletons zur Laufzeit mit einem beliebigen (CORBA-konformen) Java-ORB verbinden zu können, müssen einige zusätzliche Schnittstellen (Java portable interfaces) implementiert werden.

Zusammenfassend läßt sich sagen, daß

- die gewählte Konzeption – Abdeckung verschiedener Bereiche innerhalb eines gemeinsamen, überschaubaren Rahmens – eine höhere Motivation für Veranstalter und Teilnehmer gebracht hat,
- die fachlichen Anforderungen im wesentlichen erfüllt wurden,
- die didaktische Vorgehensweise (top-down) sich bewährt hat. Ob die umgekehrte Vorgehensweise auch sinnvoll ist, müßte diskutiert oder sogar ausprobiert werden.

Die trotz größerer Gemeinsamkeiten unterschiedlichen Praktika in Frankfurt und Darmstadt haben gezeigt, daß ausgehend von demselben Basis-Design Schwerpunkte verschoben werden können. Es wäre wünschenswert, auf Basis dieser Erfahrungen zu einer Ergänzung von Aufgaben zu kommen. Analog zu dem OSP-Projekt [Kif91] kann so eine Reihe von Aufgaben entstehen, die zu einem auf die jeweiligen Anforderungen maßgeschneiderten Praktikum führen.

Mögliche Schwerpunkte als Ergänzung des bisherigen Konzepts können in der ersten Phase (Benutzen von CORBA als Verteilungsplattform) liegen oder aber in der zweiten Phase klassische Probleme der Betriebssysteme, wie sie in Middleware-Plattformen vorkommen, vertiefen (Scheduling etc.). Neben dem Spaß, den uns die Zusammenarbeit gemacht hat, verschwinden einige Unsicherheitsfaktoren, wie sie durch ein neukonzipiertes Praktikum nicht zu vermeiden sind.

Literatur

[AMF97] Gerd Aschemann, Gerd Meister, and Stefan Fünfrocken. Praktikum Verteilte Systeme SS 97. `http://www.informatik.tu-darmstadt.de/VS/Lehre/SS97/VertSysPrak/`, 1997.

[Asc98] Gerd Aschemann. Praktikum Verteilte Systeme WS '98/99. `http://www.informatik.tu-darmstadt.de/VS/Lehre/WS98-99/VertSysPrak/`, October 1998.

[Bec97] Christian Becker. Praktikum Verteilte Systeme. `http://www.vsb.informatik.uni-frankfurt.de/lehre/praktikum/WS_97-98/`, 1997.

[Gei98] Kurt Geihs. MICO in Education. In Arno Puder and Kay Römer, editors, *Proceedings of the "First MICO Workshop"*, November 1998.

[Kif91] Michael Kifer. *Osp: An Environment for Operating System Projects*. Addison-Wesley, 1991.

[OMG97] Object Management Group. The Basic Object Adapter. `ftp://www.omg.org/pub/docs/formal/97-10-13.pdf`, 1997.

[OMG98a] Object Management Group. CORBA. `http://www.omg.org/library/c2indx.html`, 1998.

[OMG98b] Object Management Group. Mapping OMG IDL to Java. `ftp://www.omg.org/pub/docs/formal/98-02-29.pdf`, 1998.

[OOC98] Object Oriented Concepts. *ORBacus, For C++ and Java*, 1998.

[PR99] Arno Puder and Kay Römer. *MICO Is CORBA*. dpunkt-verlag, 1999.

[Pud97] Arno Puder. Personal Communications, 1997.

[Röm98] Kay Römer. MICO — MICO is CORBA. Master's thesis, Uni Frankfurt, FB Informatik, February 1998.

[1] Hierzu wurde der IDL-Compiler von ORBacus [OOC98] benutzt.