

**UML-Erweiterung für
parallele/verteilte Applikationen
und Umsetzung mit dem
UML-Modellierungstool
TogetherControlCenter**

Manfred Harrer

Zusammenfassung

Mittels des *Extension Mechanism Package* von UML werden hier Erweiterungsvorschläge für die Modellierung von parallelen und verteilten Konstrukten wie Client/Server-Architekturen oder RMI vorgeschlagen. Diese Ansammlung von *Stereotypen*, *Tagged Values* und *Constraints* wird dann mit dem UML-Modellierungstool *Together ControlCenter* von TogetherSoft umgesetzt.

<i>INHALTSVERZEICHNIS</i>	<i>2</i>
---------------------------	----------

Inhaltsverzeichnis

1	Einleitung	3
2	UML	3
3	Extension mechanism package	3
4	Erweiterungsvorschläge	4
4.1	Client/Server-Architektur	4
4.2	Übersicht	8
5	Umsetzung mit dem Modellierungstool TCC	9
6	Zusammenfassung	9
	Literaturverzeichnis	10

1 Einleitung

In der heutigen Zeit, durch Fortschritt in Hardware- und Kommunikationstechnologien bedingt, gibt es immer mehr parallele und verteilte Softwareapplikationen.

Können diese Anwendungen aber auch ausreichend modelliert werden?

Für die objekt-orientierte Analyse und das Design von Softwaresystemen hat sich die UML als standardisierte Notation für objekt-orientierte Modelle herauskristallisiert. Folgender Artikel beschäftigt sich mit den Möglichkeiten und Grenzen des Einsatzes von UML in parallelen und verteilten Softwareprojekten. Es werden, wenn nötig, Erweiterungsvorschläge gemacht und diese werden im UML-Modellierungstool *TogetherControlCenter* (TCC) von TogetherSoft umgesetzt.

2 UML

Am 14. November 1997 wurde von der *Object Management Group* (OMG) das von den drei Amigos (James Rumbaugh, Grady Booch, Ivar Jacobsen) entwickelte UML (*Unified Modeling Language*) als Standard definiert. UML ist eine Sprache zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme (vgl. [GB99]). Hierbei handelt es sich um eine Modellierungssprache und nicht um eine Methode.

Momentane Version ist 1.4. Seit 1998/99 beschäftigt sich die *Revision Task Force* (RTF) unter Leitung von Chris Kobryn mit der Weiterentwicklung einer neuen Version, der Version 2.0.

Da es ein langwieriger Entwicklungsprozess ist und da der Einfluss von Privatpersonen oder von kleineren Firmen daran sehr gering ist, wird hier, anstatt an der Weiterentwicklung direkt teilzunehmen, mit dem Erweiterungsmechanismus (erstmalig vorgestellt in UMLv1.3) versucht, gefundene Ergänzungsvorschläge in bezug auf das Modellieren von parallelen und verteilten Softwareanwendungen umzusetzen.

3 Extension mechanism package

Grundsätzlich existieren in UML zwei Erweiterungsmechanismen. Zum einen kann das Metamodell der UML um neue Metamodellelemente erweitert werden, zum anderen kann für ein bestimmtes Anwendungsgebiet mittels des standardisierten Erweiterungsmechanismus ein UML Profile generiert werden.

Da UML eine universell einsetzbare Modellierungssprache bleiben soll, können nie alle Einsatzbereiche abgedeckt werden. Würde für jedes Anwendungsgebiet das Metamodell um neue Elemente erweitert werden, würde dies zu einer wahren Metamodell-Explosion führen.

Das *extension mechanism package* besteht aus folgenden drei Mechanismen:

Stereotypes, Tagged Values und Constraints (vgl. [JS00]).

- **Stereotype (Spracherweiterung):**
Stereotypen sind Mechanismen, die es erlauben, UML zu erweitern, ohne dabei das Metamodell der Sprache zu verändern. Stereotypen können als eigene Piktogramme (Icons) oder in eckigen Klammern dargestellt werden: `<<stereotype>>`
Sie klassifizieren oder markieren die möglichen Verwendungen eines Modellelements. Das UML-Metamodell wird dadurch spezialisiert, nicht erweitert.
- **Tagged values (Eigenschaftswerte):**
Eigenschaftswerte erweitern die Semantik von Modellelementen um spezifische charakteristische Eigenschaften.
- **Constraints (Zusicherungen):**
Einschränkungen, die als formloser Text oder in OCL (*Object Constraint Language*) formuliert werden können und die immer erfüllt sein müssen. Bei Klassen sind dies dann Invarianten, bei Operationen Pre- oder Postconditions.

Ziel ist es, in Form eines Profiles das UML-Modell additiv zu erweitern, so dass eine ausreichende Modellierung von parallelen und verteilten Softwaresystemen möglich ist.

4 Erweiterungsvorschläge

Mittels dieses Erweiterungsmechanismus wurde UML so erweitert, dass eine Modellierung von parallelen und verteilten Konstrukten besser möglich ist. Ein ausgesuchtes Beispiel (Client/Server-Architektur) verdeutlicht die Vorgehensweise.

4.1 Client/Server-Architektur

Das Client/Server-Modell ist ein Software-Architektur-Modell mit den zwei Komponententypen *Client* und *Server*. Zwischen einer Client-Klasse und einer Server-Klasse existiert eine unidirektionale Relation. Die Initiative zu einer Interaktion geht stets vom *Client* aus. Der *Client* formuliert Aufträge und schickt diese zur Ausführung an einen *Server*. Server bieten ihre Dienste (*Services*) öffentlich an, bearbeiten Requests, können aber keine fremden, nicht eigene Services in Anspruch nehmen.

Mit UMLv1.3 kann dieses Szenario zwar modelliert werden (siehe Abb. 1), aber die spezielle Funktionalität der beiden Klassen (Client- und Server-Eigenschaften) bzw. der Assoziation (Client/Server-Assoziation) wird erst dann ersichtlich, wenn man sich mit der Implementierung auseinandergesetzt hat.

Ziel ist es, diese besondere Beziehungen zwischen zwei Komponententypen so



Abbildung 1: Client/Server-Beziehung mit UMLv1.3

zu modellieren, dass dieses Szenario visuell verständlich gemacht wird (siehe Abb. 2).

Wenn der Client den Server immer kennt, wird die Assoziation mit dem Attribut *permanent* versehen. Andernfalls mit *transient*. Ausserdem werden die beiden Klassen mit einem erklärenden Stereotypen versehen, um so deren Funktionalität besser zu verdeutlichen (vgl. [LM99]).

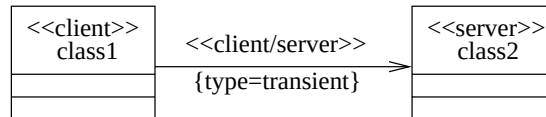


Abbildung 2: Modellierungsvorschlag mit den neuen Stereotypen

Tabellarische Notation der neuen Stereotypen und Eigenschaftswerte:

Stereotype	Base Class	Parent	Tags	Constr.	Description
client/server	Association	N/A	type:String	keine	Verdeutlichung einer Beziehung zwischen einer Client-Klasse und einer Server-Klasse.
server/client	Association	N/A	type:String	keine	s.o.

Tag	Stereotype	Type	Multiplicity	Description
type	client/server	String	1	Gibt Art des Beziehung (permanent oder transient) an.
type	server/client	String	1	Gibt Art des Beziehung (permanent oder transient) an.

Stereotype	Base Class	Parent	Tags	Constr.	Description
client	Class	N/A	servertype:String	keine	Kennzeichnet eine Client-Klasse.
server	Class	N/A	clienttype:String	keine	Visualisierung einer Klasse, die Server-Funktionalität implementiert.

Tag	Stereotype	Type	Multiplicity	Description
servertype	server	String	1	Gibt Art des Servers an (z.B. Ftp- o. Mailserver).
clienttype	client	Integer	1	Gibt Art des Clients an (z.B. Web-Browser).

Graphische Notation der neuen Stereotypen:

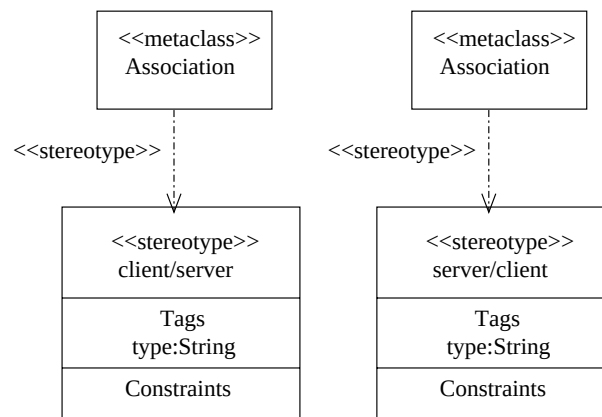


Abbildung 3: Stereotypen „client/server“ und „server/client“: Graph. Notation

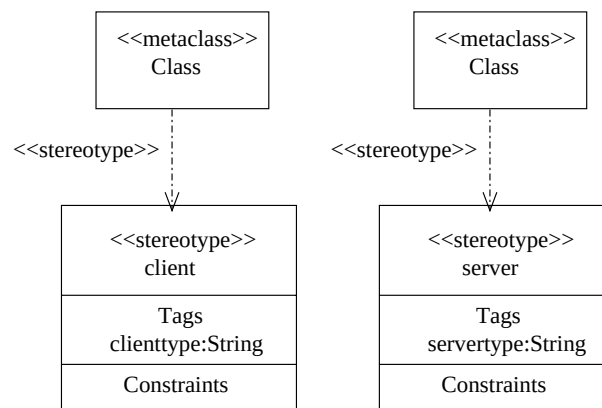


Abbildung 4: Stereotypen „client“ und „server“: Graph. Notation

4.2 Übersicht

Die folgende Tabelle gibt einen Überblick über alle neuen Stereotypen und dessen Eigenschaftswerten:

Stereotypname	Einsatz	Meta-Element	Tags	Constr.
socket	Kommunikation über Sockets	Attribute	protocol IP-address port	Ja
virtual	virtuelle Knoten	Node	Nein	
network	Netzwerk	Node	type bandwidth	Ja
local area network	Netzwerk	Node	Nein	
wide area network	Netzwerk	Node	Nein	
pick up	RPC/RMI	Message	Nein	
put	RPC/RMI	Message	Nein	
remote	RPC/RMI	Message	Nein	
remote	RPC/RMI	Object	Nein	
remote	RPC/RMI	Operation	Nein	
remote	RPC/RMI	Class	Nein	
test	RPC/RMI	Message	Nein	
client/server	C/S-Architektur	Association	type	
server/client	C/S-Architektur	Association	type	
client	C/S-Architektur	Class	clienttype	
server	C/S-Architektur	Class	servertype	
client	C/S-Architektur	Object	clienttype	
server	C/S-Architektur	Object	servertype	
service	Jini/JavaSpaces	Object	type	
service	Jini/JavaSpaces	Class	type	
entry	Jini/JavaSpaces	Object	Nein	
entry	Jini/JavaSpaces	Class	Nein	
read	Jini/JavaSpaces	Message	entry	
write	Jini/JavaSpaces	Message	entry	
take	Jini/JavaSpaces	Message	entry	
multiple instances	Vielfache Instanzen	Object	number	Ja
single instance	Vielfache Instanzen	Object	Nein	
shared variable	Kommunikation über gemeinsame Variable	Attribute	Nein	
monitor	Synchronisierte Klasse	Class	Nein	
monitor	Synchronisierte Klasse	Object	Nein	
buffer	Kommunikation über Nachrichten	Object	Nein	
send	Kommunikation über Nachrichten	Message	buf, dest tag	
receive	Kommunikation über Nachrichten	Message	source tag	
aSend	Kommunikation über Nachrichten	Message	buf, dest tag	
aReceive	Kommunikation über Nachrichten	Message	source tag	
broadcast	Kommunikation über Nachrichten	Message	buf root	
barrier	Kommunikation über Nachrichten	Message	Nein	

5 Umsetzung mit dem Modellierungstool TCC

Das TCC von TogetherSoft ist ein UML-Modellierungstool, das neben zahlreichen Features wie Versionskontrolle, Reverse Engineering oder anpassbare Metrik auch die Möglichkeit anbietet, dass zugrundeliegende UML-Metamodell mit neuen Stereotypen und Constraints zu versehen.

So kann mittels einer in einer Skriptsprache geschriebenen Konfigurationsdatei¹ den Diagrammelementen neue Eigenschaften hinzugefügt werden und dessen Erscheinungsbild verändert werden (siehe Abb. 5).



Abbildung 5: neue Stereotypen im TCC

6 Zusammenfassung

Mit dem Erweiterungsmechanismus von UML wurde hier ein UML Profile, eine Ansammlung von Stereotypen, Tagged Values und Constraints, für parallele und verteilte Softwaresysteme vorgestellt.

Konstrukte aus Applikationen, bei denen nebenläufige oder auch verteilte Algorithmen eine Rolle spielen, können mittels dieses Profiles deutlicher als nur mit dem standard UML modelliert werden.

Mit dem Together ControlCenter wurde diese Profile umgesetzt, so dass Modellelemente mit den neuen Stereotypen und Eigenschaftswerten versehen werden können und sich dann deren graphisches Erscheinungsbild verändert.

¹erhältlich über <http://www.jambit.com> oder mail: manfred.harrer@jambit.com

Literatur

- [GB99] Ivar Jacobsen Grady Booch, Jim Runbaugh. *Das UML-Benutzerhandbuch*. Addison Wesley Verlag, München, zweite edition, 1999.
- [JS00] Jürgen Wolff von Gudenberg Jochen Seeman. *Softwareentwurf mit UML*. Springer Verlag, Berlin, erste edition, 2000.
- [LM99] Christoph Maier Luis Mandel, Nora Koch. *Extending UML to Model Hypermedia and Distributed Systems*, 1999.