

# Entwicklung in der OMG Model Driven Architecture™

-

## ein Szenario für die Praxis

## Abstract



Der Beitrag setzt auf die im vorherigen Beitrag (D4) eingeführten Konzepte der Model Driven Architecture™ der OMG auf und zeigt ein praxiserprobtes Szenario für den Methoden- und Tool-Einsatz in großen Entwicklungsprojekten auf Basis objektorientierter Technologien. Bereits heute können durch den Einsatz von geeigneten UML-Modellierungstools und XMI-basierenden Code-Generatoren die Vorteile und Einsparungspotentiale der Trennung zwischen plattformunabhängigen und plattform-spezifischen Modellen genutzt werden. Gezeigt wird, wie mit Hilfe eines generativen Ansatzes, der auf einem technologieneutralen Anwendungsmodell aufsetzt und für die verschiedensten Plattformen einen Großteil des Codes generiert, Zeit und damit Kosten gespart und die Qualität gesteigert werden kann. Zusätzlich werden verschiedene technologische Aspekte der Model Driven Architecture™ beleuchtet, die zur Abrundung des Ansatzes und Steigerung des Nutzens beitragen können. Hier sind vor allem die Themen Metamodell-Modellierung mit den Meta Object Facility (MOF), Datenaustausch mit XML Metadata Interchange (XMI) und Modell-Verifikation mit der Object Constraint Language (OCL) zu nennen.

# Inhalt des Vortrags



- Kurzüberblick zur MDA™
- Anforderungen aus der MDA™-Spezifikation
- Umsetzung der Anforderungen
- Kriterien für die Werkzeugauswahl
- Einfluß auf das Projektmanagement
- Partner im Szenario

# Model Driven Architecture™



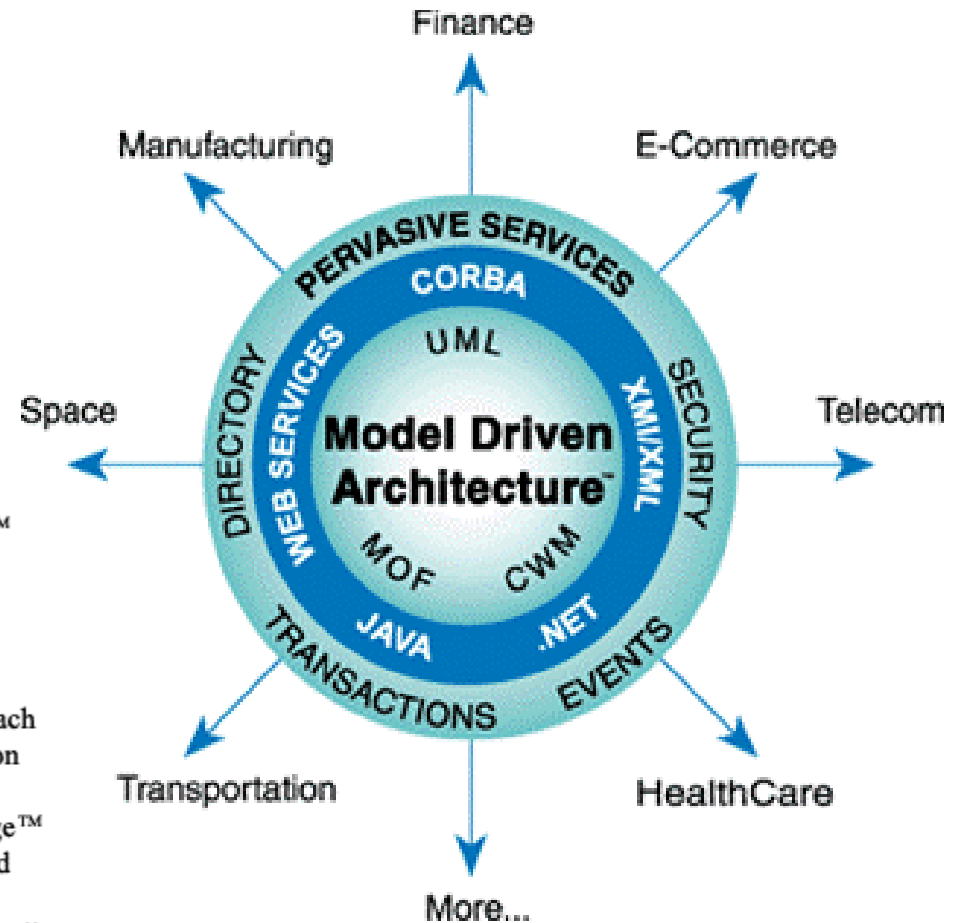
## Kurzüberblick



OMG Model Driven Architecture™

*How Systems Will Be Built*

MDA™ provides an open, vendor-neutral approach to the challenge of interoperability, building upon and leveraging the value of OMG's established modeling standards: Unified Modeling Language™ (UML™); Meta-Object Facility™ (MOF™); and Common Warehouse Meta-model™ (CWM™). Platform-independent Application descriptions built using these modeling standards can be realized using any major open or proprietary platform, including CORBA®, Java, .NET, XMI™/XML, and Web Base platforms.



JAVA FORUM  
 stuttgart 2002

# Anforderungen der MDA™



- UML™-Modellierung
  - Platform-Independent Model (PIM)
  - Platform-Specific Model (PSM)
- Datenaustausch über XMI™
  - XML Metadata Interchange (XMI™) als standardisiertes Schnittstellenformat.
- Code-Generierung auf Basis von Profilen
  - Abbildung der technologischen Aspekte in Form von Profilen als Basis der Code-Generierung.



Technologieneutrales Anwendungsmodell  
als Grundlage der Generierung

## Erweiterte Anforderungen



- Metamodell-Modellierung gemäß Meta-Object Facility (MOF™)
  - Anpassung und Ergänzung des UML-Metamodells.
- Einschränkungen per Object Constraint Language (OCL)
  - Erweiterung des Metamodells mit funktionalen Einschränkungen.
- Datenmodellierung mit CWM™-Unterstützung
  - Relationales Datenbankmodell als Ausschnitt des Common Warehouse Metamodel (CWM™).



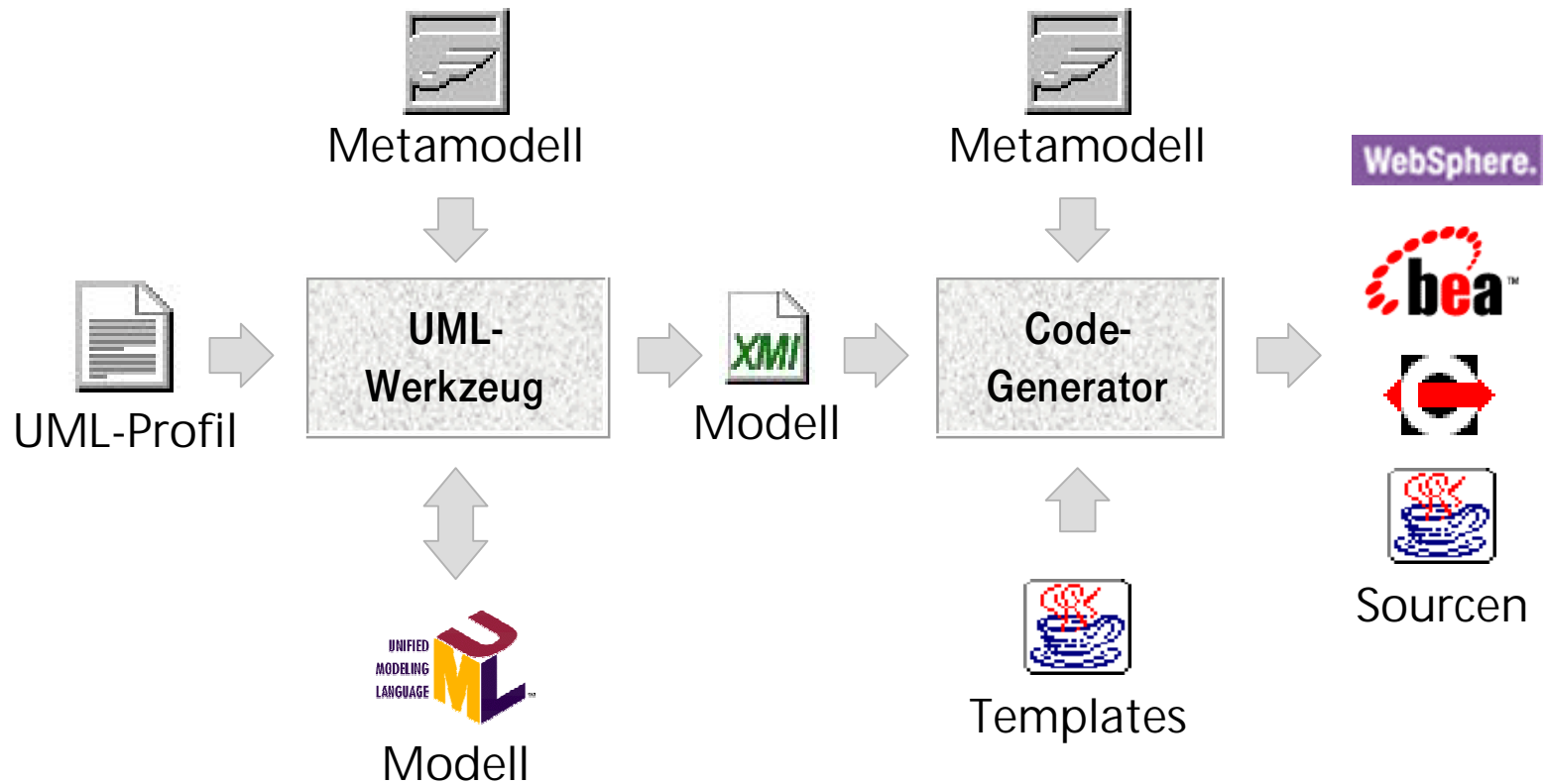
Vollständige Abdeckung der  
OMG-Spezifikation

# Umsetzung der Anforderungen



- UML™-Modellierungswerkzeug
- XMI™-Austauschformat
- Code-Generator
- Referenz-Implementierung
- Metamodell
- OCL-Einschränkungen
- MOF™-Repository
- CWM™-Modellierung

## Vom UML-Profil bis zur Anwendung



# UML-Modellierungswerkzeug



Klare Trennung zwischen PIM und PSM

## ■ Getrennte Profile

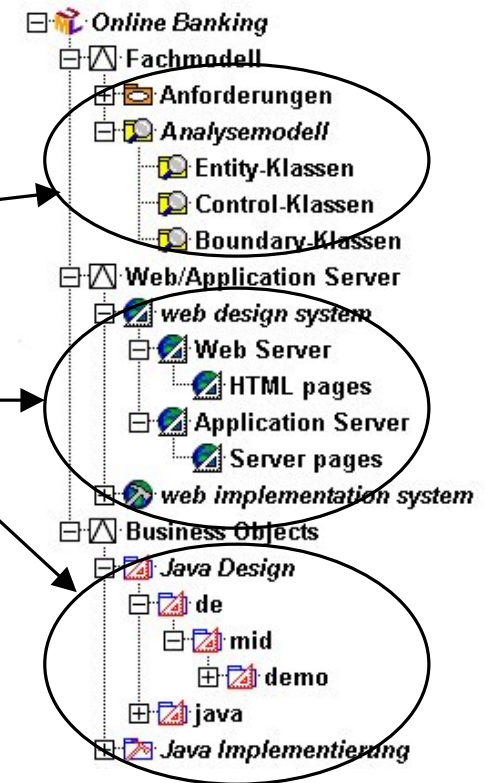
- PIM = Anforderungsmodell  
Analysemodell
- PSM = Java Design System,  
Web Design System,  
usw.

## ■ Syntax für Modellelemente

- UML-Syntax für PIM
- z.B. Java-Syntax für PSM

## ■ Mapping-Verfahren zwischen PIM und PSM

- Traceability muß gewährleistet sein



# UML-Modellierungswerkzeug



Flexibler und anpassbarer XMI-Export

- Modellinhalte
  - Komplettes statisches Modell
  - Auszug aus dem dynamischen Modell (meist nur Zustandsautomaten)
- Modellerweiterungen
  - TaggedValues
  - Stereotypen
- Full- vs. Teilexport
- „Versionsmix“
  - UML-Version: 1.1, 1.3 oder 1.4
  - XMI-Version: 1.0 oder 1.1

# XMI™-Austauschformat



XMI = **X**ML **M**etadata **I**nterchange

Das Hauptziel von XMI ist es,  
den einfachen Austausch von Metadaten  
zwischen  
Modellierungs-Werkzeugen (basierend auf OMG-UML)  
und  
Metadaten-Repositories (basierend auf OMG-MOF)  
in verteilten heterogenen Umgebungen  
zu ermöglichen.

Quelle: OMG XML Metadata Interchange (XMI) Specification; Version 1.1; Nov. 2000;  
<http://www.omg.org>



## XMI *ist nicht gleich* XMI

- Versionsstände
  - XMI- und UML-Version müssen abgestimmt sein.
- xmi.id und xmi.uuid
  - Identifizierung von Modellelementen in XMI-Dateien.
    - xmi.id            innerhalb der Datei gültig
    - xmi.uuid        global gültig
- xmi.idref vs. href
  - Referenzierung von Modellelementen in XMI-Dateien.
    - xmi.idref        innerhalb einer Datei
    - href             dateiübergreifende Referenzierung

*Nicht gleich alles auf einmal!*

- Bei großen Modellen ist der Export des gesamten Modells in eine XMI-Datei nicht praktikabel.
  - Zu lange Export- und Importzeiten
  - Unhandliche Dateigrößen
  - Inkrementelle Vorgehensweise nicht möglich

Strukturierter XMI-Export in kleinen Dateien

- fachliche oder organisatorische Gruppierung der Modellelemente in XMI-Komponenten.
- Dateiübergreifende Referenzierung durch href-Verweise.
- Globale Identifizierung der Modellelemente durch uuid-Kennung.

## Flexibel und performant zugleich!

### ■ XMI-Import

Versionsstände und Modellinhalte werden durch den Generator bestimmt und müssen mit dem UML-Modellierungswerkzeug abgestimmt sein.

### ■ Metamodell

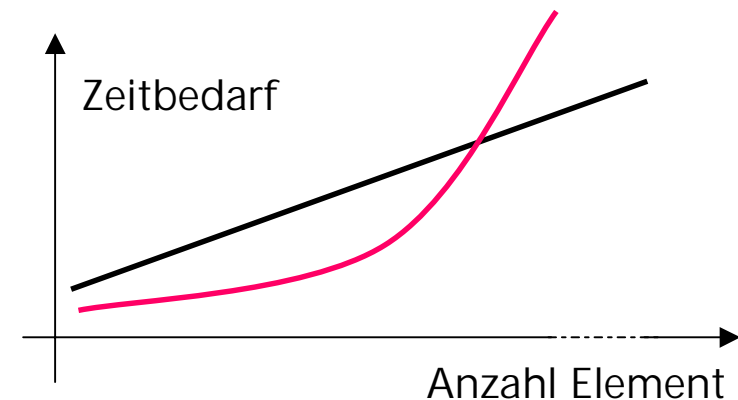
- UML™-konform und MOF™-compliant
- erweiterbar

### ■ Inkrementell mit Protected-Regions

Geschäftslogik muß in Protected-Regions plziert eine inkrementelle Generierung ermöglichen.

## Flexibel und performant zugleich!

- Templates bzw. Code-Schablonen
  - Kernstück und entscheidender Grundbaustein
    - nicht fest codiert
    - Schablonen, die zum Generierungszeitpunkt ausgewertet werden
    - möglichst in bekannter Notation (z.B. JSP)
    - Protected-Regions definierbar
- Performance
  - möglichst linear
  - Generierung auch teilweise möglich



Entwicklung in der OMG MDA™ - ein Szenario für die Praxis  
Andreas Ditze, MID GmbH, © 2002, Seite 15

# Referenz-Implementierung



*„Auferstanden aus Ruinen“*

- „von Hand“ erstellt
  - Alt- bzw. Legacy-System
  - Pilot-Anwendung
  - Inkrement 0 (Durchstich)
- Grundlage für die Code-Templates

Trennung zwischen generierbare Code-Teilen und Business-Logik.

  - Generierbarer Code                      Code-Template
  - Business-Logik                              Protected-Regions
- Abbildung und Verifikation des Architektur-Konzepts

# Metamodell



Alle Modellierungsregeln und -konventionen sollten als Metamodell dokumentiert werden.

- UML™-Metamodell als Grundlage
  - liegt als UML-Klassenmodell vor
  - DTD vorhanden
- Erweiterung im Rahmen der MOF™-Spezifikation
  - TaggedValues und Stereotypen
  - Spezialisierungen und Assoziationen
  - OCL-Einschränkungen
- Abbildung im UML™-Modellierungswerkzeug
  - UML™-Profile bilden die Ausgangsbasis
  - Erweiterungen können nur bedingt in den Modellierungswerkzeugen abgebildet werden.

# OCL-Einschränkungen



Alle funktionale Einschränkungen sollten als Ausdrücke in Object Constraint Language (OCL) Notation definiert werden.

## Vorteile

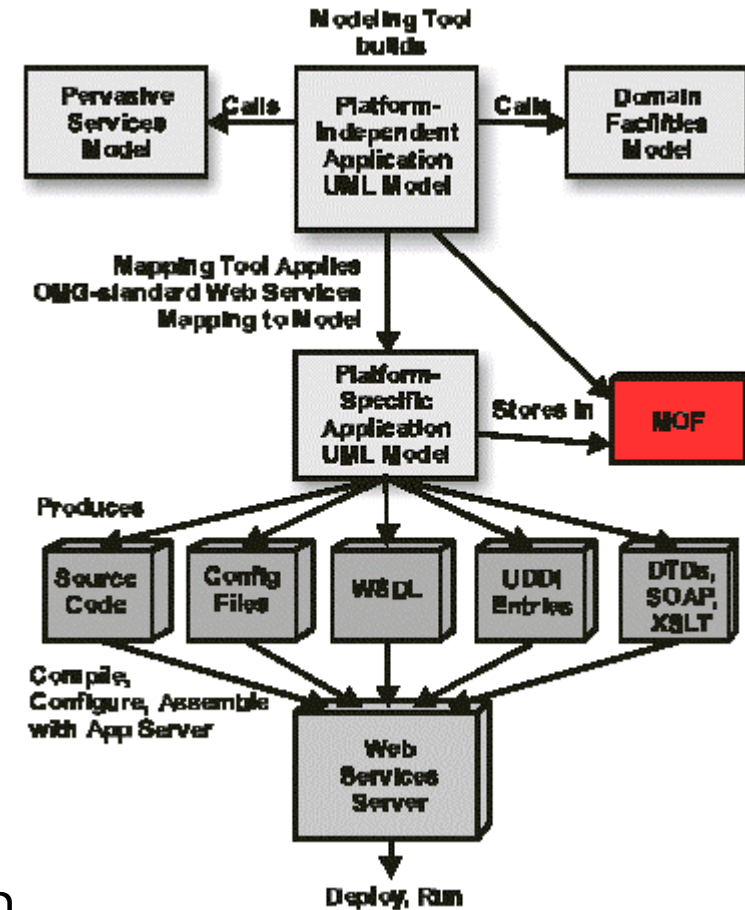
- UML™ selbst per OCL spezifiziert
- Ausdrücke können validiert werden
- vollständig objektorientiert
- Code-Generatoren können OCL auswerten

## Nachteile

- kaum Werkzeugunterstützung
- OCL-Parser sehr aufwendig
- weitere DTD-Erweiterungen für XMI notwendig

Metadaten-Repository zur

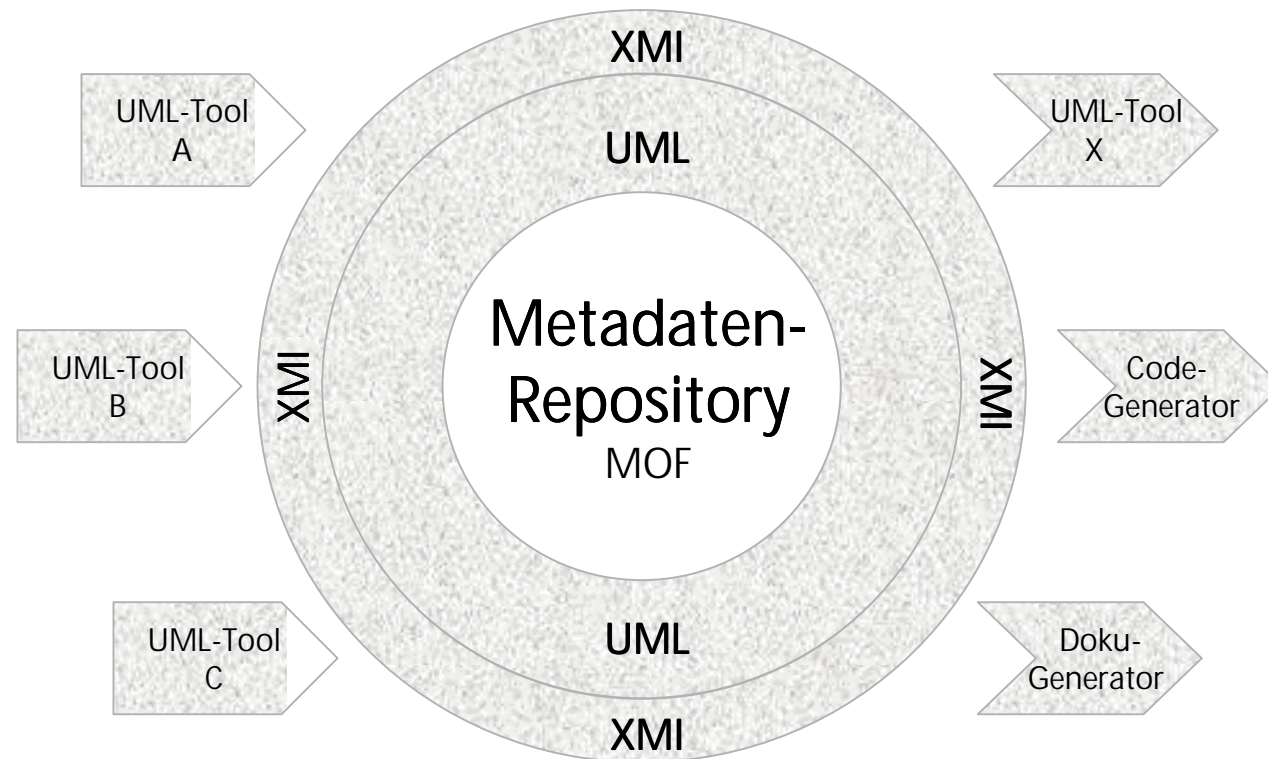
- Modell-Integration
- Modell-Validierung
- Modell-Versionierung
- Modell-Recherche
- Modell-Transformation



Quelle: Developing in OMG's Model-Driven Architecture by Jon Siegel, November 2001

Entwicklung in der OMG MDA™- ein Szenario für die Praxis  
 Andreas Ditze, MID GmbH, © 2002, Seite 19

## Metadaten-Repository als zentrale Integrationsplattform



# CWM™-Modellierung



## Common Warehouse Metamodel

Die OMG schlägt zur Beschreibung der Persistenzschicht das CWM vor.

Vorteile:

- Vollständige konzeptionelle und physische Datenmodellierung möglich
- Ausgereifte OMG-Spezifikation (obwohl V1.0)

Nachteile:

- Abbildungsregeln offen (UML vs. CWM)
- Noch nicht alle RDBMS können CWM
- Persistenz-Frameworks häufig praktikabler

# Projektmanagement



Die Einführung der Model Driven Architecture (MDA™) hat auch Einfluß auf das Projektmanagement.

- Neue Rollen
  - Code-Schablonen-Verantwortlicher
  - Metamodell-Verantwortlicher
- Neue Aufgaben
  - Code generieren
    - XMI aus UML-Modell exportieren
    - XMI-Datei verifizieren
    - Code aus XMI-Datei generieren
  - Code-Schablonen erstellen und verifizieren
  - Metamodell erstellen und verteilen
  - usw.

## Partner im Szenario



- UML-Werkzeug
  - INNOVATOR Object / Meta
  - MID GmbH, Nürnberg
  - [www.mid.de](http://www.mid.de)
  - [info@mid.de](mailto:info@mid.de)
  
- Code-Generator
  - iQgen
  - innoQ GmbH, Ratingen
  - [www.innoq.com](http://www.innoq.com)
  - [info@innoq.com](mailto:info@innoq.com)

**innovator**<sup>®</sup>  
strategic software development tool

Enterprise  
Software Solutions 



**innoQ**

Infos über Referenzen gibt es auf Anfrage.

Vielen Dank



JAVA FORUM  
stuttgart 2002

... jetzt sind **Sie** an der Reihe