

Software-Ingenieurung und MDA

Vortrag im Rahmen der Vorlesung:
„Systemdenken und Gestaltungsmethodik“ (Prof.
Dr. Brunthaler)

Hendrik Jablonski, 21.11.2007

„Ingenieuring“ = Engineering ?

- **Software engineering (SE)** is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
- The discipline of software engineering encompasses knowledge, tools, and methods for defining software requirements, and performing software design, software construction, software testing, and software maintenance tasks

Quelle: www.wikipedia.org

Was passiert wenn wir
Software „kürzen“?



Ingenieurmäßiges Arbeiten:

„... ist durch **systematische Aneignung**, Beherrschung und **Anwendung** von **wissenschaftlich-theoretisch fundierten** und empirisch gesicherten **technischen Erkenntnissen** und **Methoden** gekennzeichnet“, VDI

Warum also dieser Vortrag...

- Nach den Anforderungen ist vor dem Modell
 - Auswahl, Hierarchien und Mathematik...
- Modellierung informationeller Systeme
 - Fundamental Modelling Concepts
 - Und was ist mit UML?
- Model Driven Architecture
 - OMG-Standard
 - Ziele
 - Tools
- Diskussion

Vor dem Modell

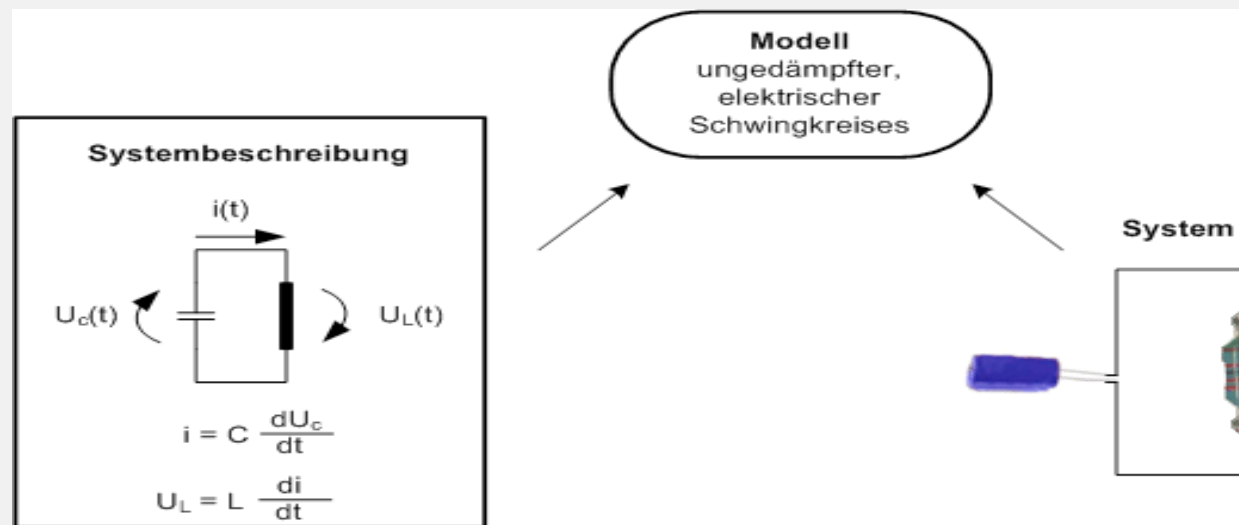
- **Materiell-energetische Systeme**
 - Umwandlung, Bearbeitung, Verarbeitung von Materie und/oder Energie
 - Sichtbarer/messbarer materiell/energetischer Nutzen
- **Informationelle Systeme**
 - Verarbeitung von Daten auf materiell/energetischer Ebene
 - Information entsteht durch Wahrnehmung, Interpretation und/oder Schlussfolgerung (durch den Menschen !!)

	<u>informationell</u>	<u>materiell/energetisch</u>
Verarbeitung	Textverarbeitung, Buchhaltung	Walze, Motor
Transport	Kommunikationssysteme	Rohrleitungssystem, Eisenbahn, Kabel
Speicherung	Buch, Festplatte	Tank, Kondensator

Vor dem Modell

Ein Modell ist eine Abstraktion zu einem System, welches nur eine Menge ausgewählter, interessierender Sachverhalte des betrachteten Systems aufweist.

- Modell vs. System vs. Systembeschreibung/dokumentation



- Häufige Modellausprägungen: Graph, Mengen, Relationen
--> diskrete Mathematik

Vor dem Modell

- Modellsystem ist das Modell des Systems aus verschiedenen Betrachtungswinkeln
- Aufbau --> Systemstrukturen/Komponenten während der Dauer der Existenz des Systems
 - Aktive und passive Komponenten
 - Operationen ausführend oder auf etwas Operationen ausgeführt
- Ablauf --> Vorgänge/Aktivitäten der Komponenten während eines Beobachtungsintervalls
 - Ereignisse, Operationen und kausale oder temporale Abhängigkeiten
- Wertebereiche/-strukturen
 - Informationelle Systeme verarbeiten u.U. komplexe Strukturen (z.B: Tabellen, Bäume, ...)
 - Entitäten/Typen, Attribute, Relationen

Modellierung informationeller Systeme

... genauer: Modellierung programmierter informationeller Systeme

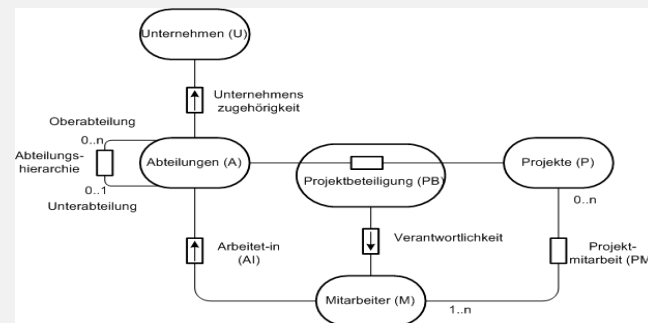
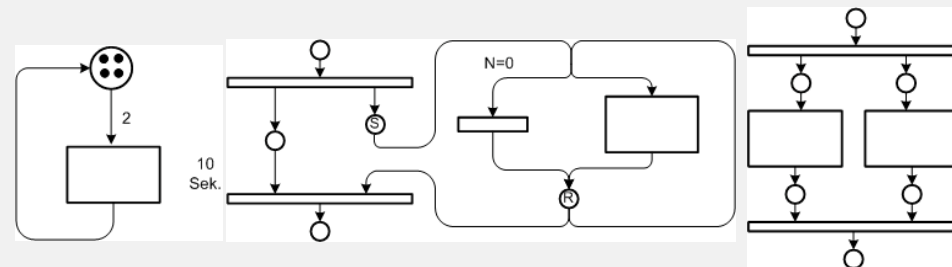
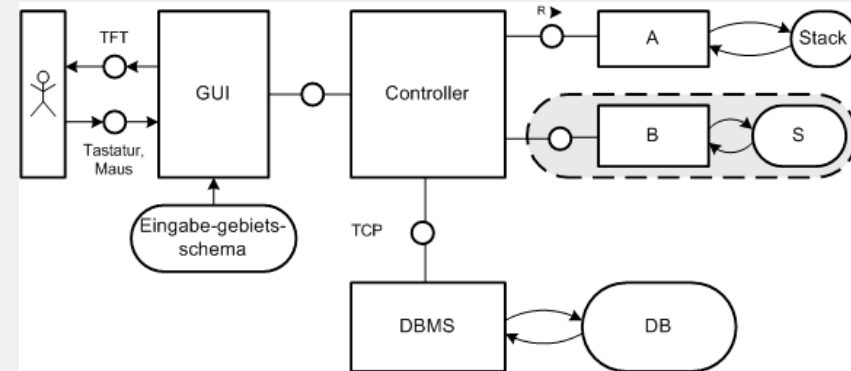
- System entsteht erst durch die Abwicklung der (hoffentlich korrekt modellierten und hinreichend dokumentierten) Systembeschreibung --> Programmcode
- Kunde weiß was er bekommt, Sourcecode ist wenig aussagekräftig
- Der Entwickler weiß was er umzusetzen hat (nicht wie)
- Fundamental Modelling Concepts (FMC)

Modellierung informationeller Systeme

FMC:

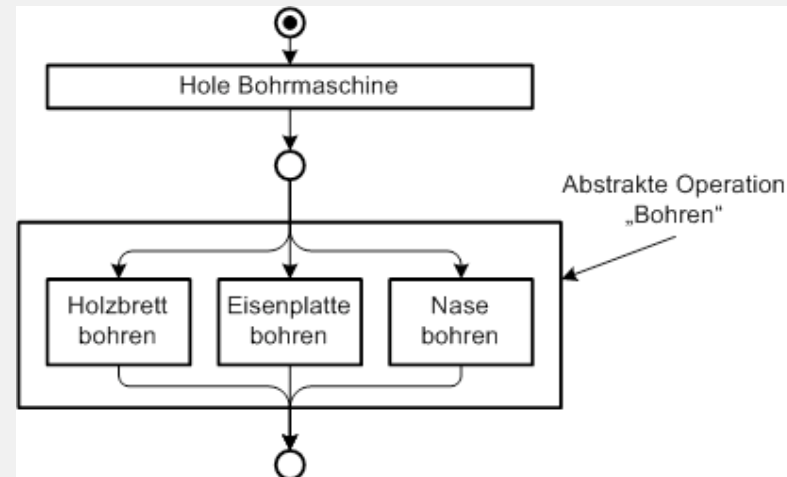
Bipartite Graphen

- Blockdiagramme
- Petrinetze mit Erweiterungen
- ER-Diagramme (Chen) mit neuer Notation und Algebra



Modellierung informationeller Systeme

- Layout und Semantik
 - Durch Layout wird ein Bild erst aussagekräftig
 - Semantik kann nicht formalisiert werden
 - Semantik ist nicht offensichtlich sondern entsteht durch Interpretation
 - Bsp.: Abb. Polymorphie in Petrinetzen
- Einsatzgebiet von FMC ist nicht auf programmierte Systeme beschränkt... siehe folgendes Beispiel (Folie 12/13)



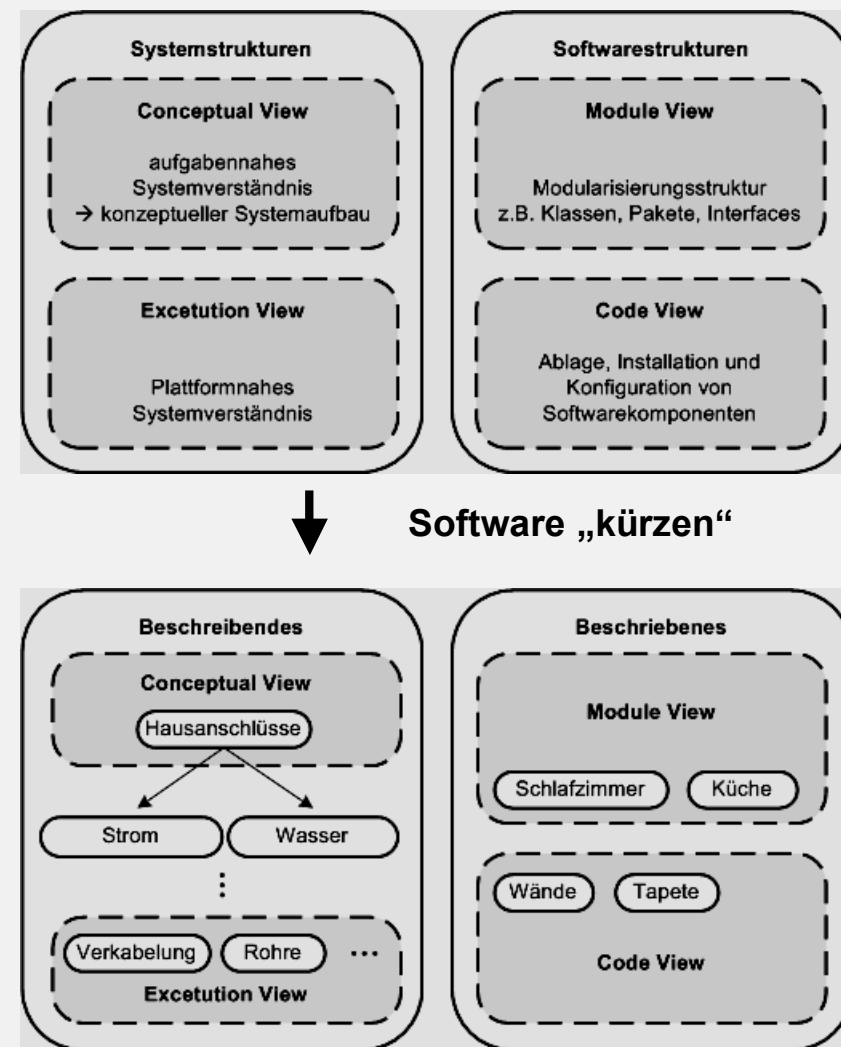
Modellierung informationeller Systeme

Was fehlt? Dokumentation programmierter informationeller Systeme

- Dokumentation der Systembeschreibung für Wartung, Erweiterung, Abstraktion von Syntax der Programmiersprache
- Der Entwickler weiß wie er umzusetzen hat
- Unified Modeling Language (UML), vorwiegend objektorientierte Modellierung

Modellierung informationeller Systeme

- 4-Sichten-Modell
- Systemmodell <> Systemarchitektur
- UML für das „wie“ --> Dokumentationsmittel, Softwarestrukturen
- FMC für das „was“ --> Kommunikationsmittel, Systemstrukturen



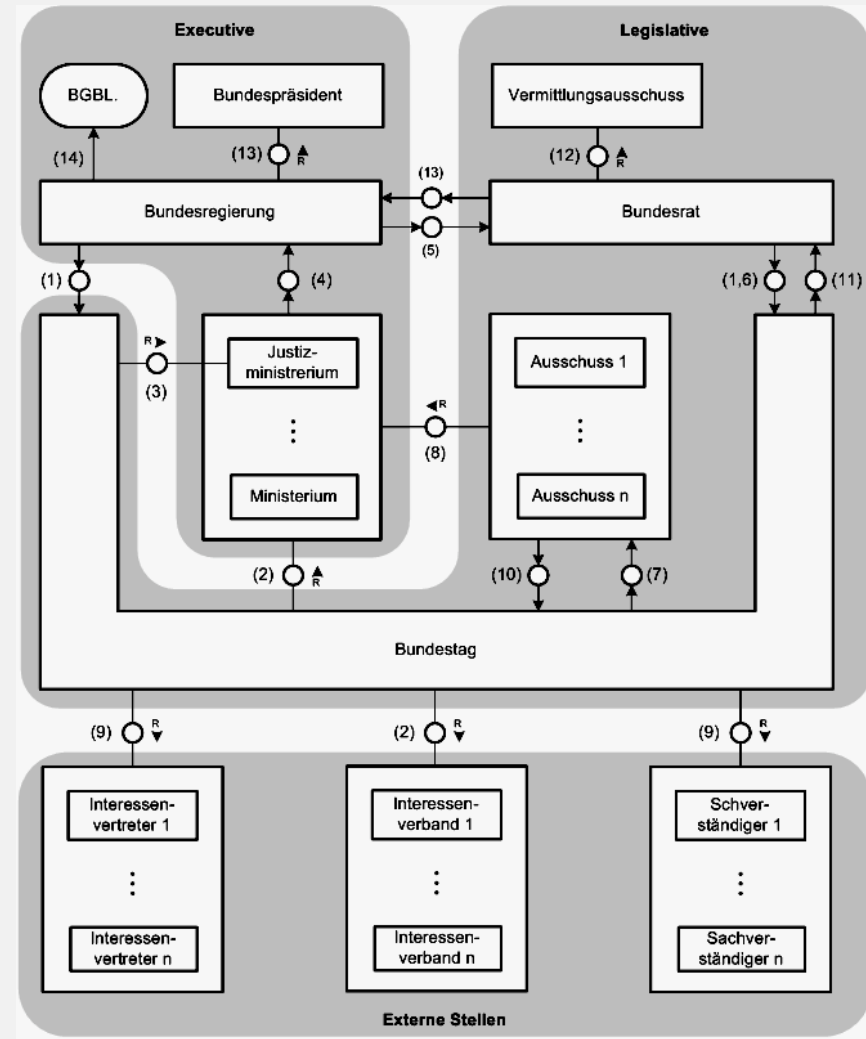
Aufbaumodell der deutschen Gesetzgebung

- Stark vereinfachte Version
 1. Bundesregierung, Bundesrat und MdB dürfen Gesetze (neue oder Gesetzes-Novelle) zur Beratung in Bundestag einbringen
 2. Fachlich beteiligte Ministerien und Interessensverbände werden gehört
 3. Nach der Abstimmung des Textes erfolgt Prüfung der Rechtsförmigkeit durch das Justizministerium
 4. Zuständiger Minister gibt Gesetzentwurf an Bundesregierung, Bundeskanzler/-in gibt diesen an den Bundesrat für Stellungnahme
 5. Stellungnahme + Gesetzentwurf zurück in Bundestag
 1. Lesung --> Überweisung an Ausschüsse, Meinung von Sachverständigen und Interessensvertretern einholen
 2. Lesung --> Beratung der vorgenommenen Änderungen
 3. Lesung --> endgültige Beschlussfassung, Abstimmung
 6. (falls Erforderlich) Zustimmung des Bundesrats einholen
 7. Ggf. Vermittlungsausschuss bilden --> Kompromisse finden
 8. Unterschrift Bundesregierung: zuständige Minister, Bundeskanzler/-in
 9. Prüfung, Unterschrift Bundespräsident/-in
 10. Veröffentlichung in Bundesgesetzblatt

Aufbaumodell der deutschen Gesetzgebung

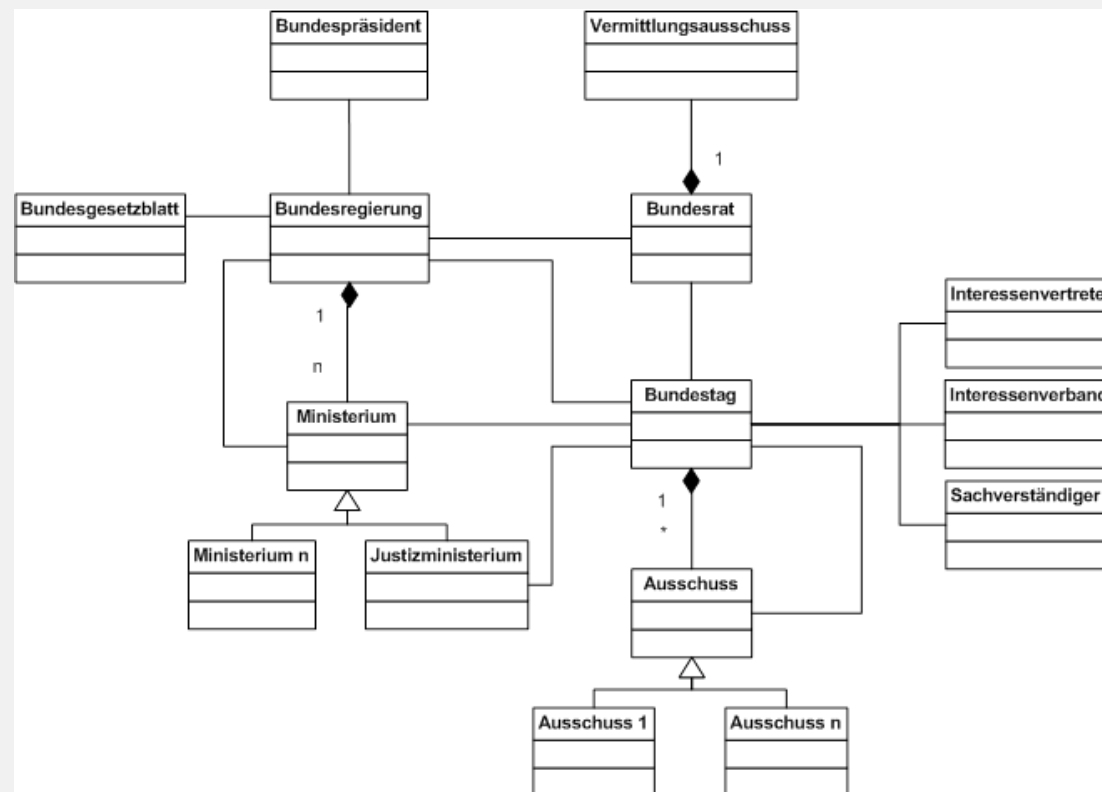
- Das Modell der **Gesetzgebung** (Aufbau)
- Beteiligte Instanzen aus Legislative und Executive
- Datenflüsse und Richtung an Kanälen
- Semantik in Anordnung (z.B: Hierarchie)

Und jetzt bitte in UML...



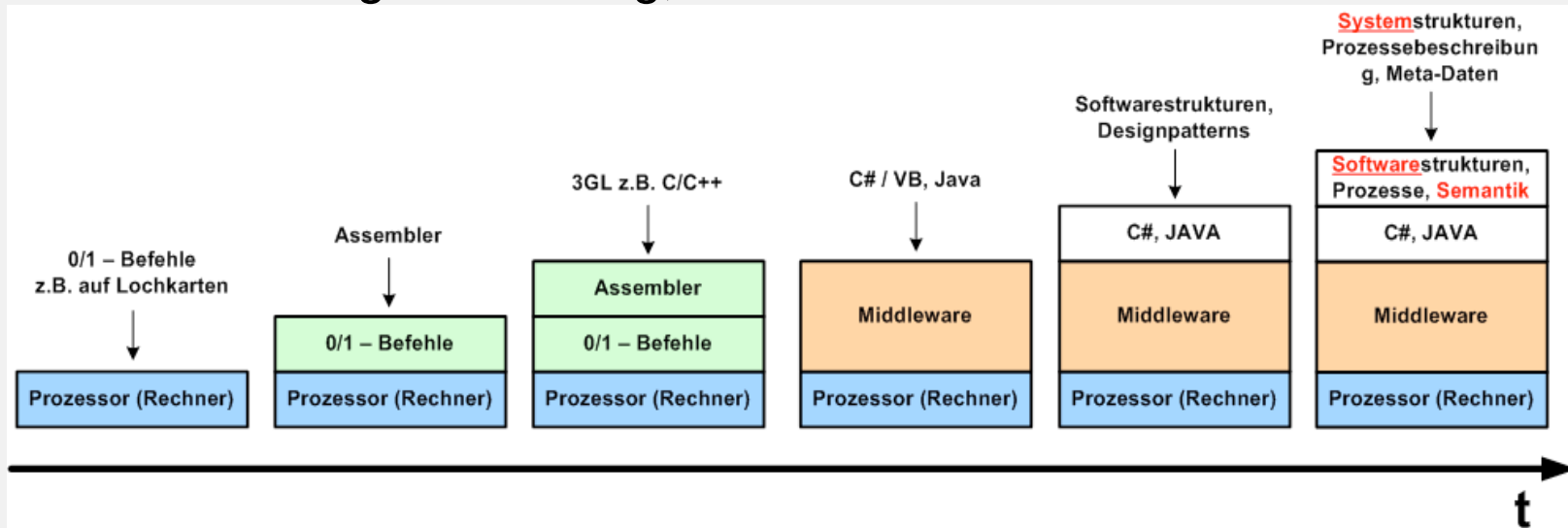
Aufbaumodell der deutschen Gesetzgebung

- UML Klassendiagramm
- Zur Kommunikation über das „Was“ wenig geeignet



Model Driven Architecture

- Schnellere Entwicklungszyklen, teure Programmierer, Bananensoftware...
- Suche nach Formalisierung von Programmierung, ausgehend von Ziel(=System)-Beschreibung
- automatisiert => idealerweise wiederholbar, schnell, billig, verifizierbar, fehlerfrei
- Visuelle Programmierung, MDA



Model Driven Architecture

- Rollentrennung Programmierer - Modellierer
 - Model = design artefact
 - 3GL-Code = development artefact
 - Führt zu informellen-“wenig nützlichen“ Modellen für den Programmierer
- Aufbrechen der Rollen in MDA
 - Formalisierung
 - Maschinenverarbeitung der Modelle
 - Modelle als Produktionsmittel
 - Das Konzept kennen wir --> „Blaupause“ und Fräsroboter bei Concept-Cars

„The modeling activity is a programming activity...“

Model Driven Architecture

- Neue Anwendung -> neue Randbedingungen
 - Erweiterung von UML über UML-Profile
 - Neue definieren über MOF
 - --> Modelle nach Plattform, Sprache, Domäne
- „.NET, JAVA oder CORBA“ kommt am Ende
 - PIM - Platform-independent Model generiert
 - PSM - Platform-specific Model
- Generieren der aktuell gewünschten Funktionalität abhängig von Parametern
- Zugrundeliegende Technologie (Ziel) muss standardisiert sein z.B. .NET, JAVA...

Model Driven Architecture

- Verschiedene Profile und Sprachen
 - CWM für Datenbankmodelle
 - UML für EJB, CORBA, UML OCL
 - UN/CEFACT Modeling Methodology für B2B-Prozesse, Services, ...

Model Driven Architecture

- Meta-Modelle für korrekte Generierung
 - Modelle müssen formal definiert sein
 - Funktioniert mit formal definiertem Metamodell
 - Wird definiert über MOF (Ebene 3 der MDA Meta-Level)

M3	Meta Object Facility („Factory“)	MOF Klasse, MOF Attribut, MOF Assoziation
M2	Wie definiere ich einen Typ/ eine Klasse	Klassennamendefinition, Attributdefinition,... für UML in UML
M1	Typdefinition, Klassifizierung,...	Klasse Student, Klasse Vorlesung,... in UML
M0	Konkteres Objekt	Objekt, „Hendrik Jablonski“

Model Driven Architecture

- Meta-Daten...
 - Unternehmens-spezifische Daten
 - Datenmodelle
 - Transformationsregeln
 - Beschreibung bestehender Dienste+Schnittstellen
 - Tuningparameter für MDA-Generatoren
- Metadatenaustausch mit XMI (XML Metadata Interchange)
- MOF nach JAVA mit JMI (JAVA Metadata Interface)



Model Driven Architecture

- JMI
 - Metadaten als JAVA-Objekte
 - Metamodell wird als JAVA-API abgebildet
--> Klasse „Objekt“, Klasse „Attribut“, ...
 - (MOF = Spezifikation dieser API...)
 - konkrete Modelle = Objekte/Objektkompositionen
 - Strukturmodell --> Objekt „Student“, Objekt „Dozent“,
Assoziation „betreut Masterarbeit“, ...
 - Ergebnis: eine eigene (generierte) API für meine
Anwendung, bei entsprechenden Ablaufplänen sogar mit
fertigen Methoden

Model Driven Architecture

- MDA live, eine neue Rollenverteilung
 - Business Analyst -> Geschäftsmodell, Produktvision, Zielvorstellung, Anforderungen
 - Requirements Analyst -> ViewPort-Definition für MDA, Anforderungen mathematisch festschreiben
 - Application Engineer -> Produktkomponenten und Programme, PIM --> PSM, händische Erweiterung
 - Middleware Engineer -> PSM-Betreuung, Plattform Verwaltung
 - Quality Assurance Mgr -> testet PSM + Plattform gegen das PIM
 - Deployment Engineer -> analysiert Reports, erstellt Runtime-Profile PIM --> Deployment Model
 - Network Admin -> Performance, Umsetzung Deployment
 - Architect --> Erstellt MDA, Schnittstellen zu Frameworks
 - Infrastructure Engineer --> betreut InHouse-Services, 3rd-Party Middleware

Model Driven Architecture

- Ziele Von MDA
 - Wortwörtlich: „Modell-getriebene Architektur“ --> guter/richtiger Ansatz
 - Codegenerierung --> Wegen NP-Vollständigkeit gibt es immer Lösung, wenn auch nicht optimal
 - Schnelleres Prototyping
 - Externe Programming --> Extreme Modelling

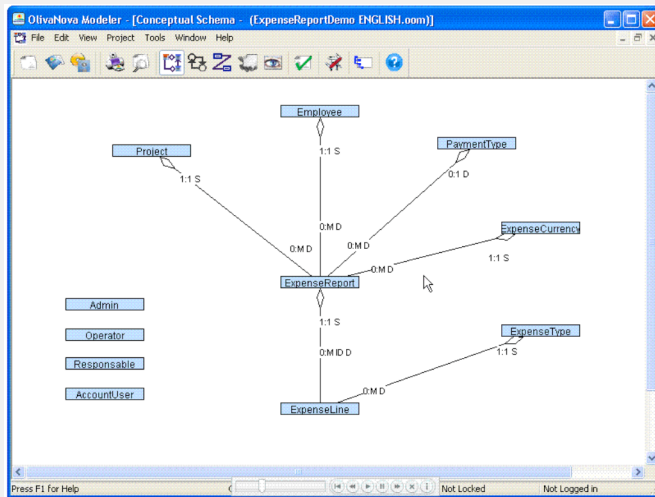
Model Driven Architecture - Tools

- ICONNECT, www.micro-epsilon.de + Uni Passau
 - Applikationsgenerator für Messen, Steuern, Regeln, Automatisieren von Prozessen
 - Grafische Entwicklung/Programmierung auf Basis Signalgraphen und Datenflussdiagramme
 - Kein echtes MDA
 - API für Nutzung in anderen Umgebungen
 - Signalerfassung, Signalverarbeitung, Visualisierung

Model Driven Architecture - Tools

- CARE Technologies OLINOVA www.programmiermaschine.de
 - Echtes MDA Tool (ähnlich UML-Style)
 1. Statisches Modell (Klassen, Attribute, Dienste, Rollen, Assoziationen)
 2. Funktionales Modell (Wer kann was, Pre-/Post-Conditions, etc)
 3. Dynamisches Modell (Anwendungslogik mit Zustandsdiagrammen)
 4. GUI festlegen (Forms, Sprache, Views)
 - Programmerstellung erfolgt online bei care-t.com
 - Abrechnung per zertifizierter (Gartner Group) Funktions-Points-Berechnung
 - Ab 24 € je FP... 400 FP = 10.000 €

Model Driven Architecture - Tools



The dialog shows a list of available services categorized into Business Logic, User Interface, and Other services. A callout box explains the purpose of the profile selection.

Available services:

- Business Logic
 - Business Logic for Transactional Architecture Windows platform
 - Business Logic for Scalable Transactional Architecture Windows platform for CDM+
 - Business Logic for Highly Scalable CORBA-EJB Architecture. JAVA platform
 - CH Business Logic Scalable Transactional Architecture Windows platform for CDM+
- User Interface
 - User Interface Thick Forms Architecture Windows platform
 - User Interface Thin Web Architecture JSP platform
 - User Interface CH.NET Forms Architecture for Windows platform
 - User Interface Thin Web Architecture ASP.Net platform
- Other services
 - Function Point counter (IFPUG standard, validated by Gartner)

As you manoeuvre through STAR you find that you are able to define the profile of your application - selecting what kind of server-side logic you want and what kind of user interface you want the transformation engine to generate for you.

The login screen includes a checkbox for 'Apply manual changes' and a section for 'Olivanova Advanced Code Manager login'. It contains fields for Profile (User), Identifier (Chris), and Password. A callout box provides instructions on the manual changes option.

Apply manual changes

Olivanova Advanced Code Manager login:

Profile: User

Identifier: Chris

Password: *****

Save Login

This step allows you to run **OLIVANOVA Advanced Code Manager**. This application helps you to apply the manual changes in the source code automatically after the code is obtained.

The application displays a table of expense reports and a detailed view of expense lines.

Cause	Advances	Balance	Payment	Comments	Payment Date	Status	Total Expenses	Auth
Meeting in London	0	180	[null]		[null]	Closed	180	[null]

Nr Line	Date	LnDescription	Price	Quantity	Total Line
1	07/07/2006	Flight to London	100	1.0	100
2	07/07/2006	Two nights in Hotel	40	2.0	80

Model Driven Architecture - Tools

- objectiF, www.microtool.de
 - Tool für PIM -> PSM -> Code
 - Sehr ausgereift
 - Focus auf SOA und Webanwendungen
 - „... und es entsteht - natürlich - jede Menge Code“
 - Modell (Hülle) in UML, Transformation definieren, Ausgabe als Eclipse-, VS.Net-Projekt
 - Erweitern um „Kernfunktionalität“ weiter notwendig
 - „Verhältnis Logik : useless Code --> 5 : 95“
 - Ca 2000 € (oder 990€ für feste Plattform)

Model Driven Architecture - Tools

- EnterpriseArchitect, www.sparxsystems.com
 - Modellierung in aktueller Version Uml 2.1
 - UML Code-Generator und live Editing, kein echtes MDA
 - Vollversion ab 200 US-\$
 - Academic License ab 65 US-\$!!

Model Driven Architecture - Tools

- IBM Rational Rose www-306.ibm.com
 - Unterstützt nur Uml 1.x
 - „Architektur“ mit UML-Diagrammen erstellen
 - Product Lifecycle Management
 - CodeGenerator für Hülle
 - Lizenz ca 2200 €

Model Driven Architecture - Tools

- AndroMDA galaxy.andromda.org
 - OpenSource MDA Framework
 - Hauptsächlich J2EE Unterstützung
 - Weiterhin viel manuelles Programmieren notwendig

Was ist hängengeblieben?

- Software kürzen !!
- Modelle sind ein beschreibendes Werkzeug
- Modell und Entwurf haben zwei verschiedene Ziele
- FMC als Modellierungssprache
- „M“DA als systembeschreibendes Systemumsetzungsverfahren

Programmierung ist Handwerk