PLATTFORMÜBERGREIFENDE ENTWICKLUNG MITHILFE MODELLGETRIEBENER METHODEN UND TECHNOLOGIEN

Mathias Slawik, WI (M), 3. FS

Aktuelle Themen der Wirtschaftsinformatik, HTW Berlin, WS 10/11

Gliederung

- Methode & Technologie
  - Modellgetriebene Entwicklung
  - Eclipse Modeling Project
- Vorstellung Praxis-Projekt
- Live – Demo

Unterbrechen, falls Ausführung unklar!

Nachfragen, um etwas zu lernen!

Methode & Technologie

Modellgetriebene Entwicklung

Eclipse Modeling Project

Meta-Ebenen (1)

M3 = Meta-Metamodell

- Definiert formale Syntax und Semantik
- Beispiele: MOF, Ecore
- z.B. „Klasse“, „Beziehung“, „Attribut“

M2 = Metamodell

- Beispiele: UML, CWM
- (Domänenspezifisches) Modell in der M3-Sprache
- z.B. „Akteur“, „Anwendungsfall“, „Aktivität“

M1 = Modell

- Legt fest, in welcher Form das Metamodell in Programm umgewandelt werden soll
- Instanziiert M2 – Metamodell
- z.B. „Akteur: Kunde“, „Akteur: Bearbeiter“, „Aktivität: Angebot Schreiben“

M0 = Programm

- Laufzeit-Instanz des Modells
- z.B. „Java-Servlet“, „.NET-Applikation“, „PHP-Webanwendung“
- Manuell implementiert oder generiert

Modellgetriebene Entwicklung Elemente

- Rolle der Modelle = Rolle des Codes
  - Modell als Ausgang des Anwendungssystems
- Domänenspezifische Metamodelle und Sprachen
  - Möglichkeit, für jede (Teil-) Aufgabe ein passendes Metamodell zu verwenden
  - Transformationen zwischen unterschiedlichen Metamodellen
    - Möglich durch einheitliches Meta-Metamodell
- Generierung von Programmcode aus Modell
  - Voraussetzung: Flexible Code-Generatoren
  - Valides Modell = Valider Code (im Ideal)

Eclipse Modeling Project (1)

- EMF (Core)
  - Ecore = Meta – Metamodell
  - Java Code-Generator für Metamodell
  - Ergänzende Technologien = Workspace-Integration
    - Compare
    - Query
    - Transaction
    - …

Eclipse Modeling Project (2)

- Vormalige openArchitectureWare - Technologien
  - Einheitliches oAW-Typsystem
  - Xpand (Templatesprache)
  - Xtend (Funktionale Ergänzungen und M2M-Transformation)
  - Check (Validierung von Modellen)
  - Workflow Engine (jetzt MWE)
  - Xtext (Textuelles Modellierungsframework)
- GMP = Grafische Modellierungswerkzeuge
  - Tooling, Runtime, Notation, Graphiti

Eclipse Modeling Project (3)

- Viele weitere Projekte
  - Größtenteils Inkubatoren in Alpha und Beta-Qualität
  - Teilweise auch sehr ausgereift
    - EMF seit 2001 von IBM entwickelt
    - openArchitectureWare seit 2005 von itemis entwickelt
- Besonderheit des EMP
  - Komplettes Ökosystem von untereinander kompatiblen (Meta-)Modellierungstools
  - Freie Software (EPL), basierend auf Java
  - Offene Standards (OMG)

Modellgetriebene Entwicklung Chancen und Risiken

Chancen

- Skaleneffekte, v.A. bei Weiterentwicklung
- Gesamter Prozess nachvollziehbar
- Vereinfachte Validierung des Programmcodes
- Hohe „Griffigkeit“ der Modellierung

Risiken

- Generierung vs. Implementierung
- Hohe Fähigkeit zur Abstraktion nötig
- Validierung des Metamodells herausfordernd
- Erhöhte Komplexität durch Meta-Ebenen

Vorstellung Praxisprojekt

Praxisprojekt Developer Garden APIs

- Offene Web-Services (SOAP, REST) zur Nutzung in beliebiger Software
  - Voice Call (Telefongespräche aufbauen)
  - Conference Call (Outbound Telefonkonferenzen)
  - Send SMS
  - IP Location
- Nutzung ohne individuelle Verträge
- Kostenkontrolle durch Pre-Paid - Mechanismus
- Bereitstellung von SDKs in 3 Programmiersprachen
  - Java, PHP und .NET

SOAP-SDKs (bis Ende 2010)

- Externe Entwicklung
  - Relativ teuer
  - Aufwändige Abstimmungsprozesse
  - Termin- und Qualitätsprobleme des externen Lieferanten
  - Doku SDK != Doku Service
  - Keine einheitlichen Konventionen
  - Technisch veraltet (Axis1, WSE)

REST-SDKs (ab 2011)

- Ablösung der SOAP-SDKs
  - Eigenentwicklung (günstiger)
  - Kaum Abstimmungsprozesse
  - Qualitätssteigerung durch modellgetriebene Methoden
  - Doku SDK == Doku Service, da gleiches Verfahren
  - Einheitliche Konventionen durch Code-Generierung
  - Technische Verbesserungen
    - .NET – WCF
    - Java – keine externen Abhängigkeiten, Java 1.4 – kompatibel
    - PHP – cURL

REST-SDKs

Modellgetrieben, Plattformübergreifend (1)

- Ecore - Metamodelle
  - Service-Metamodell
    - Service, Methode, Parameter, Typen, …
  - Unit Test – Metamodell
    - Test, Test-Schritt, „Rufe Methode auf“ – Schritt, „Prüfe Rückgabewert“ – Schritt, …
  - Sample – Metamodell
    - „Definiere Variable“, „Rufe Methode auf“, …

REST-SDKs

Modellgetrieben, Plattformübergreifend (2)

- Developergarden – Modell
  - Instanz der Metamodelle mit Beschreibung der Schnittstelle, Samples, Unit-Tests, etc.
- Xpand Code – Templates
  - Umsetzung der Modelle in Code der Zielsprache
- Xtend Erweiterungen
  - Unterstützung der Code-Generierung
    - z.B. Kapselung häufig oder übergreifend verwendeter Ausdrücke
- Build – Workflow (Apache Ant)
- Dokumentations – Repository (Docbook)

REST-SDKs

Modellgetrieben, Plattformübergreifend (3)

Generierung (ca. 90%)

- Serviceklassen
- Datenklassen
- Daten-Factories
- Unit-Testklassen
- Sample-Klassen
- VS-Projektdateien
- Code-Doku (De & En)

Implementierung (ca. 10%)

- Elternklassen der Service- und Datenklassen
- Stylesheets
- Ant-Workflows
  - Maven-Build
  - MSBuild
  - PHPDoc
  - Test-Ausführung

Szenario 1 – Neuer Service

Alt

Serviceklassen Java - De
Serviceklassen Java - En
Serviceklassen PHP - De
Serviceklassen PHP - En
Serviceklassen .NET - De
Serviceklassen .NET - En
Datenklassen Java - De
Datenklassen Java - En
Datenklassen PHP - De
Datenklassen PHP - En
Datenklassen .NET - De
Datenklassen .NET - En
Daten-Factories Java - De
Daten-Factories Java - En
Daten-Factories PHP - De
Daten-Factories PHP - En
Daten-Factories .NET - De
Daten-Factories .NET - En

Unit - Tests Java - De
Unit - Tests Java - En
Unit - Tests PHP - De
Unit - Tests PHP - En
Unit - Tests .NET - De
Unit - Tests .NET - En
Samples Java - De
Samples Java - En
Samples PHP - De
Samples PHP - En
Samples .NET - De
Samples .NET - En
Doku Java - De
Doku Java - En
Doku PHP - De
Doku PHP - En
Doku .NET - De
Doku .NET - En

Neu

- Service-Modell
  - Methoden
  - Parameter
  - Typen
  - Samples
  - Tests
  - Beschreibung DE & EN
- Aufwand bzgl. Anzahl der Zielsprachen gleichbleibend

Szenario 2 – Neue Zielsprache

Alt

- Alle Klassen für alle Services in allen Zielsprachen
- Je mehr Services, umso größerer Aufwand pro neuer Zielsprache

Neu

- Vorlagen in Zielsprache
  - Wiederverwendung der Xpand - Templates
- Aufwand bzgl. Anzahl der Services gleichbleibend

Ausblick

- Release der SDKs Ende 2010 / Anfang 2011
- Aufarbeitung des Konzepts in Masterarbeit
- Falls möglich: Freigabe als REST SDK Generation Framework auf dem Developer Garden

Live - Demo