



Dr. Ralph Guderlei, Ralf Zozmann
eXXcellent solutions gmbh

Xtext als Template-Sprache zur Dokumentenerzeugung aus Ecore-Modellen

Agenda

- Basics: Dokumentenerzeugung aus (Ecore)-Modellen
 - Motivation
 - Konzept
- Demo: Einsatz Xtext-basierter Templates
 - Allgemeiner Workflow
 - Customizing/Erstellen von Templates
- Erkenntnisse: Erfahrungen, Kritik, Fazit

Modelle sind die Grundlage von
allem!

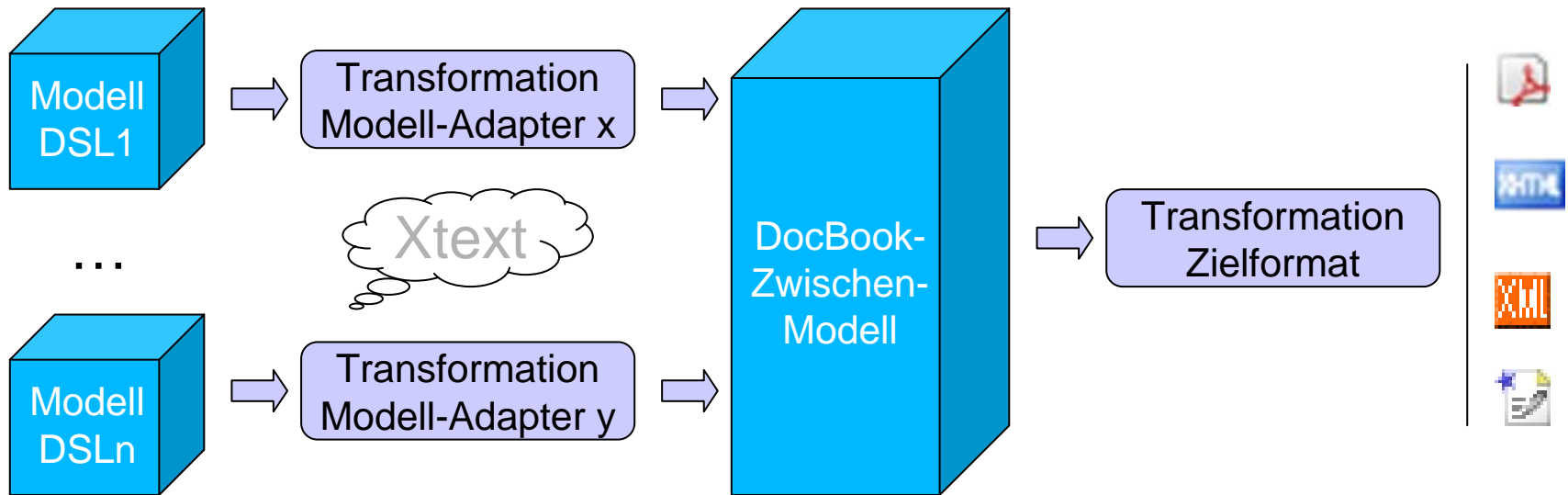
Dokumentation vs. Report

- Report
 - Relationaler Ansatz: Datensätze werden über eine Query aus einer Quelle entnommen (**Auswertung**), Fokus auf Listendarstellungen
 - Report muss vom Benutzer erstellt und mit wachsendem Modellinhalt nachgepflegt werden
- Dokumentation
 - **Darstellung** von Inhalten in repräsentativer Form, Modellstruktur ist Basis
 - Modellorientierter Ansatz: Abbildungsvorschrift für Modellelemente, unabhängig von Tiefe/Menge

Motivation

- Analyseartefakte müssen mit dem Kunden abgestimmt werden
 - Export/Aufbereitung notwendig ...als Dokument
- Systembeschreibungen (Ist-Stand) in weitgehend automatisch erstellen
- Einhaltung von Corporate Design
 - Trennung von Layout und Inhalt
- Unterschiedliche Dokumentenformate gleichzeitig bedienbar

Konzept



- Weitgehende Konfigurierbarkeit der Transformation
 - u.a. durch Zusatzangaben für jedes Modellelement direkt im zugehörigen Modelleditor
- Weitere Funktionalität zum Filtern und Sortieren

Funktionsweise

- Das Projekt als Ganzes mit allen Modellen ist relevant, nicht nur ein einzelnes Modell
- Traversieren des Modells (Rekursion) liegt in der Hoheit des Generators, nicht des Modell-Adapters (Template)
 - DocBook definiert einen 2-dimensionalen Baum (Parent-Child-Beziehung)
- Modell-Adapter dokumentiert exakt nur genau das übergebene Modellelement vollständig/atomar
 - Es ist spezifisch für eine Modellklasse

Warum Xtext?

- Modellbasiert!
- **Endbenutzerfähig**
 - einfach, aber Sprache (Template-DSL) mit gleicher Mächtigkeit wie Java
- Textueller Editor, um alles im Überblick zu haben ohne ‚Rumgeklicke‘
- Ohne Deployment sofort im Projekt benutzbar
 - Ein Editor muss verfügbar sein: ContentAssist, Validierung, Highlighting...

DEMO

Funktionalität der Template-DSL

- Iterieren über Collections
- Bedingte Ausführung
- Benutzung/Import zusätzlicher Modelltypen und anderer Templatefiles
- Benutzerkonfigurbare (im UI) Optionen
- Wiederverwendbare ‚Unterfunktionen‘
- Aufruf von Java-Klassen/Methoden
 - Angabe zusätzlicher Classpath-Elemente
- Mehrsprachigkeit
- Auswertung von Ausdrücken

Struktur eines Templates

Modellklasse

Ein Kapitel mit Überschrift, wobei sich der Absatz aus dem Inhalt der ‚description‘-Property ergibt. Der Inhalt wird als strukturierter Text ala Mylyn/Wikitext interpretiert.

template for uml::Package

documentation:

section

title:

text expression 'element.name'

content:

markuptext expression 'element.description'

endsection

children:

expression 'element.ownedElement'

label:

expression "'Package \' + element.name + '\'"

endtemplate

Ein Name für das Modellelement, um z.B. bei Verarbeitungsfehlern schönere Texte zu bekommen

1. Inhalt

2. Kinder

3. Label

Welche referenzierten Modellelemente sollen als Unterkapitel erscheinen?

Realisierungsaspekte

- **Notwendig**
 - Definieren einer DSL als Xtext-Modell (+ Generieren)
 - Schreiben eines ScopeProviders
- **Sinnvoll**
 - Implementieren eines Validators
 - Implementieren eines LinkingService
- **Optional (Usability)**
 - Implementieren von ‚besserem‘ ProposalProvider (für ContentAssist), Highlighting und OutlineView

„Kritik“

- Verwendung von Dependency Injection (Google Guice) im API (inkl. Verhalten) eines Frameworks ☹
 - Beeinflusst (aus Benutzersicht) unkontrollierbar die Schnittstelle
 - Nicht transparente Abhängigkeiten
- Eigene Implementierung von EMF-ResourceSet (XtextResourceSet)
 - Verhindert generisches Laden von Xtext-Modellen

Erfahrungen & Eindrücke 1/2

- Es kann nur bedingt Kontrolle über die Ablage/Packagestruktur generierter Klassen ausgeübt werden
- Einige Klassen werden nur erstmalig erzeugt, danach aber nicht mehr aktualisiert/überschrieben
 - In diesen Klassen muss manueller Code eingefügt werden
 - Wie wird das Verhalten in neueren Versionen von Xtext sein?
 - Wie werden API-Änderungen vermittelt?

Erfahrungen & Eindrücke 2/2

- Vererbung statt Delegation
 - Funktionsnutzung nur mit Vererbung möglich, wenn redundanter Code vermieden werden soll
- Implementierung eines Code-Formatters praktisch nicht nutzbar/wartbar
 - Formatierung nur durch direkten Zugriff auf Eigenschaften der generierten Grammatik möglich (*....getEndChildrenKeyword_5_2()*)

Fazit

- Initialer Aufwand für den Einstieg ist moderat/klein
- Grösster Aufwand fliesst in die Anpassung des Editors
 - Xtext ist aber gerade wegen des ‚geschenkten‘ Editors interessant
- Gesamtaufwand in Relation zu einer Eigenentwicklung ist erfreulich gering

Xtext ist empfehlenswert!

Fragen



*Mehr Informationen am Stand von eXXcellent Solutions
oder unter <http://www.excellent.de/orchideo/>*