cellent finance solutions AG

Erfahrungen mit modellgetriebener Komponentenentwicklung

Ralf Kretzschmar-Auer

Version 1.12 – 08.06.2008

cellent finance solutions AG

Agenda

- **Kurze Vorstellung zur Person und Firma**
- **Unser Komponentenmodell**
- **Design und Generierung**
- **Generierung von Benutzeroberflächen**
- **Maven – der Build-Experte**
- **Schluss**

cellent finance solutions AG

Cellent Finance Solutions AG

- **Spezialisiertes Consulting-Unternehmen und führender Anbieter von Software- und Produktlösungen für die Finanzwirtschaft**
  - 100%ige Tochter der LBBW
  - Sitz in Stuttgart, Niederlassung in München
  - Rund 200 Mitarbeiter
- **Produktfamilie SMARAGD: Bekämpfung von Finanzkriminalität**
  - Geldwäsche
  - Embargoüberwachung
  - Risikobewertung
  - Betrugsbekämpfung

cellent finance solutions AG

Zur Person

- **Ralf Kretzschmar-Auer**
- **44 Jahre alt, verheiratet, eine Tochter**
- **Runde 20 Jahre Erfahrung mit Objektorientierung**
  - C-Emulation (z.B: X-Windows)
  - C++
  - Java
- **Über 10 Jahre als Architekt unterwegs**
  - Netzmanagement-Systeme in C++ (Pionierära)
  - Java Systeme im Finanzbereich
- **Chefarchitekt für die Produktfamilie SMARAGD**

Architektur heißt in Fesseln tanzen – Walter Gropius

cellent finance solutions AG

Agenda

- Kurze Vorstellung zur Person und Firma
- Unser Komponentenmodell
- Design und Generierung
- Generierung von Benutzeroberflächen
- Maven – der Build-Experte
- Zum Schluss

5

cellent finance solutions AG

Ein wenig Historie

- Erste Ideen beim Aufkommen von Java
- 1999: Experimente mit Software Through Pictures (STP) und Corba
- 2001: Erstes Projekt mit EJBs
  - Handgestrickt
- 2003: Erste Version SMARAGD/Monitor
  - Generierung mit eigener Toolchain + Together
  - Vieles improvisiert
- 2007: Umstellung der Toolchain.
  - Modellierung mit MagicDraw, Generierung mit open Architectureware
  - Reuse in weiteren Projekten
  - Entscheidung für Produktfamilie auf Basis von Komponenten
- 2008: Herausziehen und Bereitstellen von Komponenten

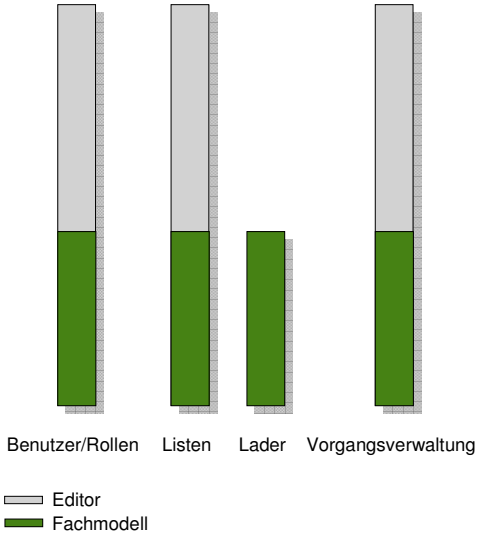
6

cellent finance solutions AG

Was ist eine Komponente ?

cellent finance solutions AG

Architekturmuster: Das Säulenmodell

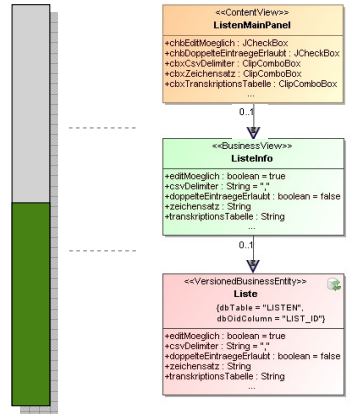


- **Fachliche Komponenten sind die Säulen des System**
  - fachliche Bausteine
  - lösen Fachprobleme, z.B.
    - Benutzerverwaltung
    - Textlisten
    - Laden von XML-Daten
  - können wiederverwendet werden
- **Hieraus *kann* der fachliche Kern einer Produktfamilie entstehen**

cellent finance solutions AG

Fachkomponente

- **Säulenarchitektur findet sich im Komponentenschnitt wieder**
  - 1 Komponente = 1 Modell
  - 1 Teilkomponente (z.B. GUI) = 1 Paket im Modell
  - Zusammen in 1 MagicDraw Design gepackt
- **Modell und Paket im Modell bekommen eigene Stereotypen**
  - Component
  - PartComponent

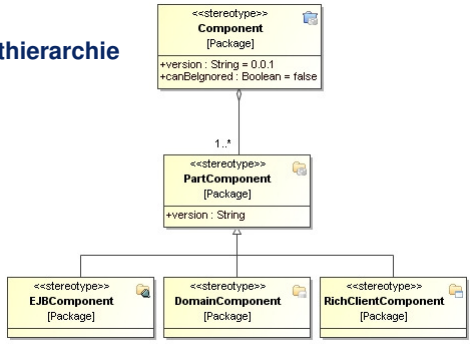


Listen

cellent finance solutions AG

Hierarchie der Komponenten

- **Zwei Ebenen**
  - Komponente
  - Teilkomponente
- **Darunter normale Pakethierarchie**

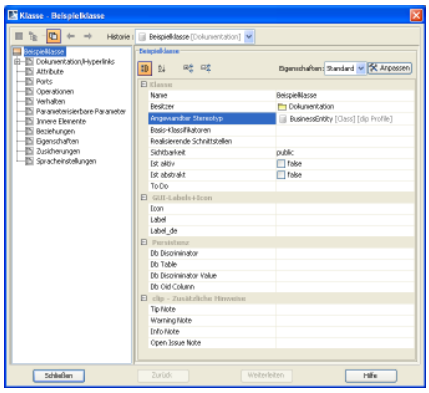


Agenda

- Kurze Vorstellung zur Person und Firma
- Unser Komponentenmodell
- Design und Generierung
- Generierung von Benutzeroberflächen
- Maven – der Build-Experte
- Zum Schluss

Domain-Specific Language

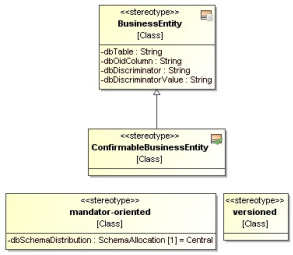
- **Gezielte Vorauswahl aus den Möglichkeiten der UML**
  - Strukturen passend zur späteren Architektur
  - vorbereitete Stereotypen
- **Anpassung von UML-Werkzeugen**
  - Profil: Vorbereitete Stereotypen nebst Attributen
  - Customization
    - Halbautomatische Vergabe von Stereotypen
    - Dialoge zur Eingabe von Eigenschaften
    - Konfiguration des Anzeige-verhaltens (Icons, Farben)



cellent finance solutions AG

Beispiel für einen Stereotyp: BusinessEntity

- **Persistente Objekte in der Datenbank**
- **Keine oder nur wenige Funktionen**
  - Implizite Funktionen zum Erzeugen, Finden und Löschen
- **Komplexes Implementierungsmodell**
  - Führt zu vielen Implementierungsdetails
  - Ergänzung von Fachfunktionen in der Impl-Klasse

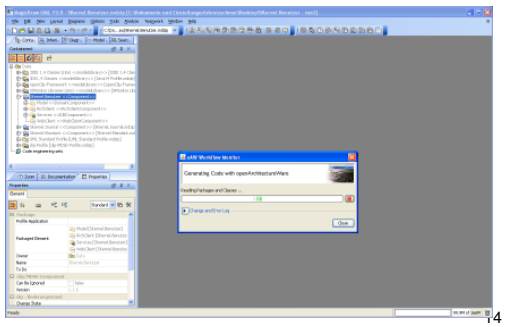


- **Varianten**
  - «ConfirmableBusinessEntity»: Unterstützt das 4-Augen-Prinzip
  - «mandator-oriented»: zusätzlich mandantenfähig (Flag-Stereotyp)
  - «versioned»: zusätzlich zeitliche Versionierung (Flag-Stereotyp)

cellent finance solutions AG

Tools: Modellierung mit MagicDraw

- **Günstiges Preis/Leistungsverhältnis**
- **Flexible Lizenzierung**
  - Installation auf 3 Rechnern je Lizenz
- **Gute Erweiterbarkeit in der (freien) Grundversion**
  - Eigenes Plugin für open Architectureware
- **Flexible Dokugenerierung**



cellent finance solutions AG

Tools: open Architectureware

- **Sehr flexibler Baukasten**
- **Mittlerweile ein Quasi-Standard (Eclipse-Bestandteil)**
- **Hat (fast) alles, was wir uns damals wünschten**
  - Einfache Sprache
  - Gute Fehlererkennung
  - Checks für die Prüfung
  - Exportierbare Modelle (bei Verwendung von ECore)
- **Aber auch Verbesserungsfähig (Version 4.2)**
  - Dank Opensource kein Problem. Wir haben ergänzt
    - Dateiorientierter Umgang mit geschützten Blöcken
    - Schutz vor Überschreiben von Dateien bei trivialen Änderungen
    - Einsammeln von Importen

cellent finance solutions AG

Modellgetriebene Entwicklung: Was ist essentiell?

- **Gut durchdachte Architektur**
  - Generator verlangt deutlich höhere Qualität als handcodierte Architektur
  - Entwicklung kann evolutionär erfolgen
    - Zuerst einfach, aber bereits mit Ansatz für später
    - Komplexere Dinge nachträglich ergänzen
    - Architekturumbau ist (fast) kein Problem!
- **Standards in den Generator gießen, Fachanteil von Hand ergänzen**
  - Führt oft zu Overlay-Klassen und höherer statischer Komplexität
- **Lesbaren Code erzeugen**
  - Dokumentation einfließen lassen
  - Auch generierten Code kommentieren (erleichtert Fehlersuche!)
- **Kein Roundtrip und kein Aufhacken möglich**
- **Unittests nur auf höherem Niveau notwendig**

cellent finance solutions AG

Agenda

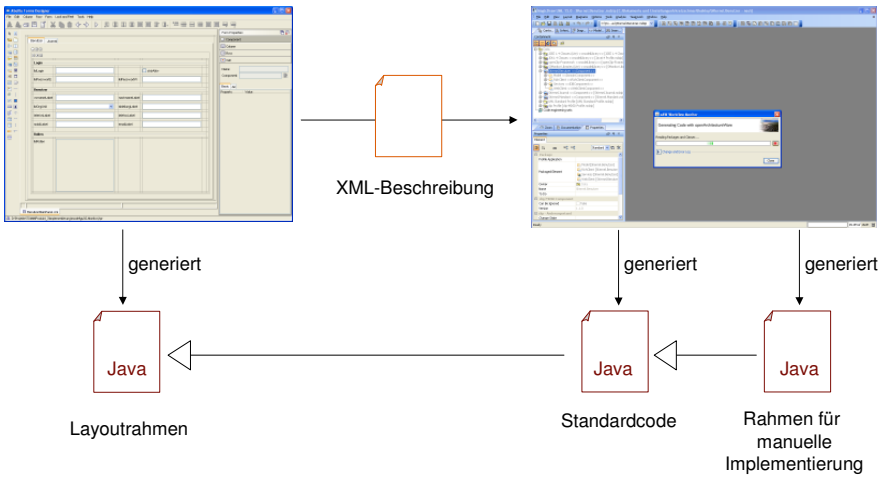
- Kurze Vorstellung zur Person und Firma
- Unser Komponentenmodell
- Design und Generierung
- Generierung von Benutzeroberflächen
- Maven – der Build-Experte
- Zum Schluss

cellent finance solutions AG

Unsere Ansätze

- **2004: Vollgenerierung aus der UML**
  - Spezifikation als Klasse
  - GUI-Eigenschaften als Properties
  - Entwicklungs- und Wartungskatastrophe
    - Layout nur schwer zu warten
    - Zu viele Callbacks für Detaillösungen (kontextsensitives Verhalten)
- **2005: Hybride Architektur für Swing**
  - Wir nutzen aus der Welt das, was uns am meisten hilft
  - Grafisches Design der Oberfläche mit einem GUI-Tool (Abeille)
    - Layout
    - Kontextsensitives Verhalten
  - Technisches Design der Oberfläche mit der UML
    - Anbindung an das Fachmodell
    - Automatischer Abgleich
    - Automatische Standardvalidierung (Feldlängen, leere Angaben etc)

Verbindung der Werkzeuge



Warum eine hybride Lösung?

- **Reine UML Modellierung kann nur Standardsysteme erzeugen**
  - Wir wollen einen benutzerzentrierten Ansatz fahren
  - Bei datenzentrierten Ansätzen kann sie funktionieren
- **Layout komplexer Dialoge nicht in der UML zu spezifizieren**
  - Was uns das Leben schwer machte waren die Ausnahmen
- **Swing-Entwickler kommen mit kompletter Generierung nicht klar …**
  - … und normale Entwickler auch nicht

cellent finance solutions AG

Agenda

- Kurze Vorstellung zur Person und Firma
- Unser Komponentenmodell
- Design und Generierung
- Generierung von Benutzeroberflächen
- **Maven – der Build-Experte**
- Zum Schluss

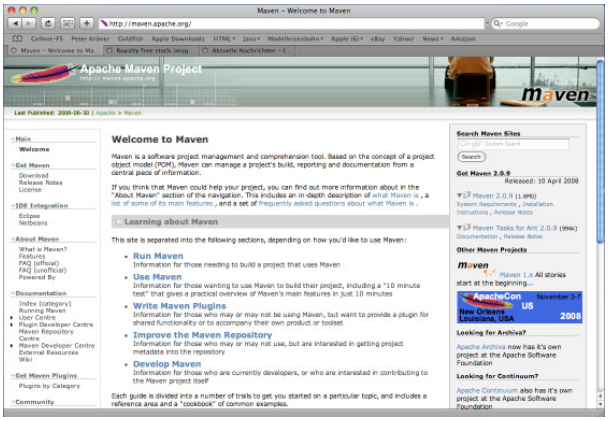
cellent finance solutions AG

Unser Build-Historie

- **1999: Ant-Skripte**
  - Bis heute bei den „alten“ Projekten im Einsatz
  - wunderbar flexibel: Man kann alles damit tun
  - Wunderbar Zeitaufwändig – eben weil man alles damit tun kann
  - einer Modularisierung im großen Stil nicht gewachsen
- **2003: Cruise Control**
  - Permanente Builds – permanente Integration
  - Wert liegt in der gesicherten Produktion
- **2008: Maven**
  - Viele Dinge über Standardlayouts geregelt
  - In vielen Fällen einfach die Zielstruktur benutzen
  - Hohe Flexibilität, aber Umdenken erforderlich

Was ist Maven?

- **Buildsystem einer neuen Generation**
  - Hochgradig komponentenorientiert
  - Sehr flexibel
  - Viele Dinge auf Standards basiert
  - Schnell
  - Aber leider auch mit Tücken versehen

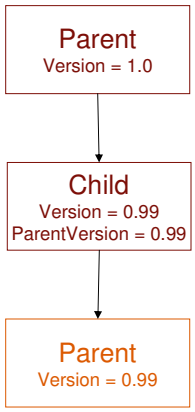


Erfahrungen mit Maven

- **Grundlegende Dinge einfach**
  - „ein jar in 5 Minuten“ – mit allem Komfort
  - Sehr stabiles System
- **Verstecktes KnowHow**
  - Wenig brauchbare Literatur verfügbar
  - Unsere 3 Top-Probleme
    - Wie geht man richtig mit Versionsnummern um?
    - Wie baut man wiederverwendbare WAR-Komponenten?
    - Das Maven Hierarche-Paradxon (→ nächste Folie)
- **Versteckte Bugs**
  - Warum baut unser System unser System unter Windows so und unter Linux anders?
  - Festhalten von Versionsnummern bei Plugins oftmals zwingend notwendig

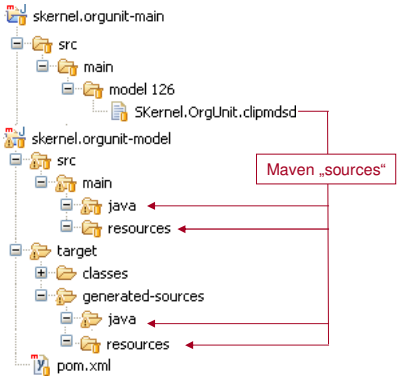
Das Maven-Paradoxon

- **Module**
  - Bau auf Basis von Verzeichnisverweisen
  - Es wird verwendet, was auf der Platte liegt
  - Bau bricht ab, wenn Module fehlen
- **Kind-Eltern**
  - Besondere, versionierte Dependency
  - Bau auf Basis von Verzeichnisverweisen und Versionsnummer
  - Enthält das Elternverzeichnis nicht die geforderte Version wird auf das Repository zugegriffen
- **Fehlerquelle mit hohem Suchfaktor**
  - Besonders Ärgerlich: Versionsänderung im Parent



Maven und Codegenerierung

- **Umsortierung der Paketstruktur**
  - Super-Projekt für CruiseControl
  - Main-Projekt für Modelle
  - Weitere Projekte für Teilkomponenten
- **Nach Fertigstellung einer Komponente Export des UML-Modells aus MagicDraw**
- **Generierung erfolgt aus dem Main-Projekt oder aus MagicDraw heraus**
- **Unterdrückung der Generierung über Steuerdateien**



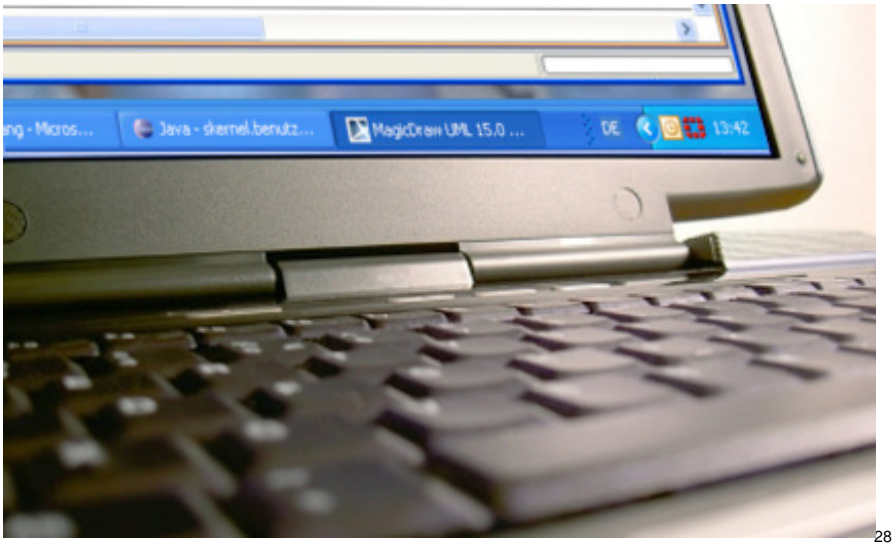
cellent finance solutions AG

Agenda

- Kurze Vorstellung zur Person und Firma
- Unser Komponentenmodell
- Design und Generierung
- Generierung von Benutzeroberflächen
- Maven – der Build-Experte
- Zum Schluss

cellent finance solutions AG

Zum Schluss



Cellent Finance Solutions AG

Calwer Strasse 33
70173 Stuttgart
Telefon 0711/222 992 812
Telefax 0711/222 992 999

Ralf Kretzschmar-Auer
Software Architekt
Mobil 0170/323 58 38
Ralf Kretzschmar-Auer@cellent-fs.de

www.cellent-fs.de