

Generative Programmierung

Ulrich W. Eisenecker
Mathias Henss, Markus Lang, Max Schlee
Fachhochschule Kaiserslautern

www.generative-programming.org
www.gp-web.org

Übersicht

Teil 1

- Beginn 15^{30}h

- Einführung
- Beispiel
- Ausblick & Quellen

- Pause 17^{00}h -17^{30}h

Teil 2

- Beginn 17^{30}h

- Merkmalmodellierung
- GUI-Generierung
- Fallstudie GP-WEB

- Ende 19^{00}h

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

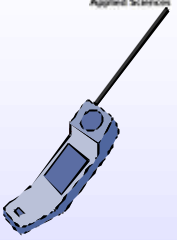


Übersicht

- ➢ Herausforderungen
- • Einzelsystementwicklung
- • Entwicklung von Systemfamilien
- • Technikprojektionen
- • Model Driven Architecture

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Komplexität



- Einsatz von Software in neuen Anwendungsgebieten
- Software unterstützt zunehmend schwierigere Aufgaben
- Anforderungen an die Software hinsichtlich Leistungsmerkmalen und Leistungsfähigkeit wachsen beständig
- Notwendigkeit der Integration mit bestehenden Anwendungen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

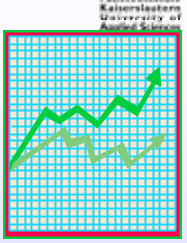
Qualität



- Einsatz von Software in lebenskritischen Bereichen
- Wirtschaftliche Verluste bei Ausfall oder Fehlfunktion von Software
- Benutzbarkeit als Akzeptanzkriterium
- Anwender erwarten eine bestimmte, aus verschiedenen Kriterien zusammengesetzte Softwarequalität

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

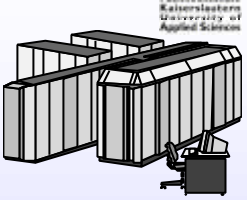
Produktivität



- Der Bedarf an neuen Systemen und Varianten steigt ständig
- Produktivitätsfortschritt und Personalzuwachs können die Nachfragelücke nicht schließen
- Standardsoftware nimmt an Bedeutung zu, kann aber nicht an alle Anforderungen angepasst werden

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Wartung & Altsysteme



- Oftmals lange Lebensdauer von Anwendungssoftware (20 Jahre und mehr)
- Softwaresysteme müssen beständig an neue Anforderungen angepasst werden
- Häufiger Wechsel der Systemumgebung
- Die Pflege der Altsysteme bindet in hohem Maße Entwicklungskapazität
- Softwaresysteme von heute sind die Altlasten von morgen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Objektorientierung



- Theorie
  - Wiederverwendung soll Produktivität und Qualität steigern
- Praxis
  - Klassen als Wiederverwendungseinheiten sind zu klein
  - Frameworks sind größer, ihre Integration ist jedoch oft schwierig bis unmöglich
  - Entwurfsmuster sind wiederverwendbares Wissen, liegen aber nicht als Code vor

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Softwarekomponenten



- Standards und Plattformen verbessern die Zusammenarbeit von Komponenten und ihre verteilte Ausführung
- Viele Komponenten haben eine spezielle Betriebsart für die Anpassung
- Geringer Nutzen kleiner Komponenten vs. aufwendige Anpassung großer Komponenten
- Komponentenbasierte Systeme weisen mitunter eine ungenügende Performanz auf

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Eine Ursache



Die vorherrschenden Verfahren und ihre Werkzeuge sind auf die Entwicklung einzelner Systeme ausgerichtet

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Herausforderungen
- ➢ Einzelsystementwicklung
- Entwicklung von Systemfamilien
- Technikprojektionen
- Model Driven Architecture

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

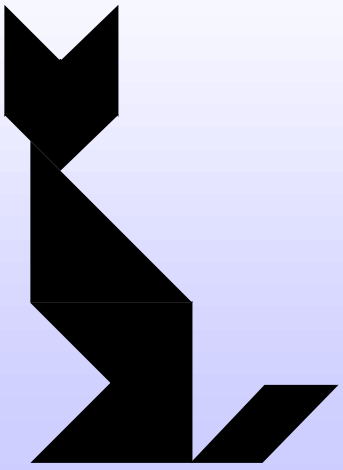
Einzelsystementwicklung

- Analyse, Entwurf und Implementierung eines einzelnen Systems
- Wichtiges Ziel: Aufteilung in Bausteine (Modularisierung)
  - Einzelne Bausteine sind leichter wartbar, austauschbar und wiederverwendbar
  - Doch nach welchen Kriterien erfolgt die Aufteilung?

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Rückschau auf ein Experiment

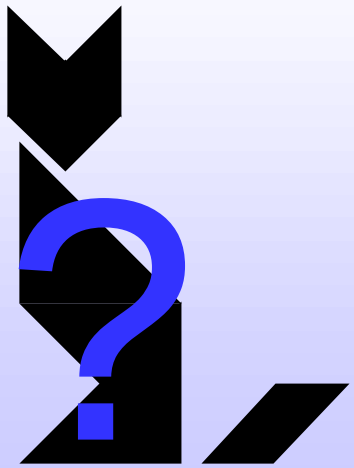
- Die rechts abgebildete Katze soll modularisiert, also in Bausteine aufgeteilt werden



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Rückschau auf ein Experiment

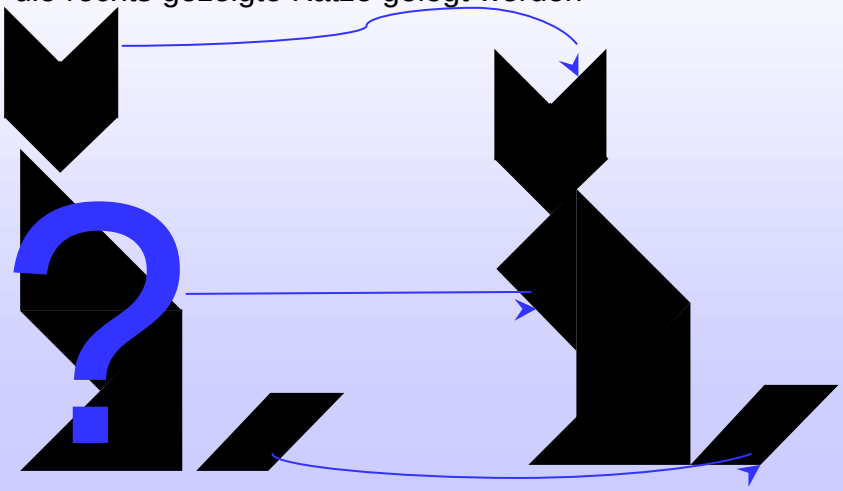
- Schwanz und Kopf werden meist sofort abgetrennt
- Die Aufteilung des Rumpfs fällt deutlich schwerer



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zweiter Teil des Experiments

Mit den gewonnenen Komponenten soll die rechts gezeigte Katze gelegt werden



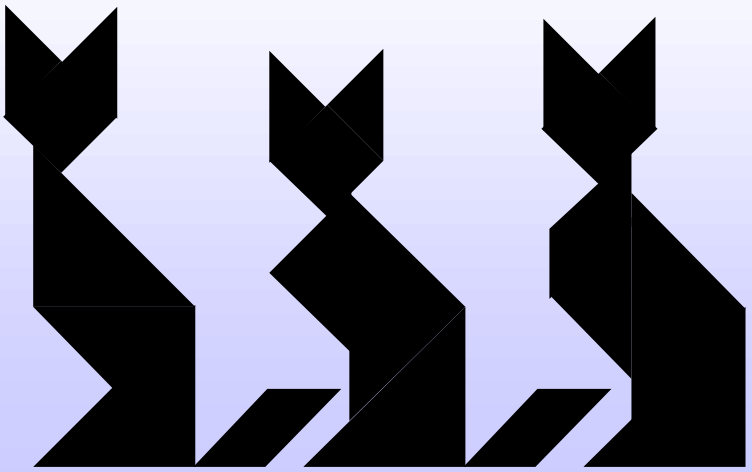
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht

- Herausforderungen
- Einzelsystementwicklung
- ➢ Entwicklung von Systemfamilien
- Technikprojektionen
- Model Driven Architecture

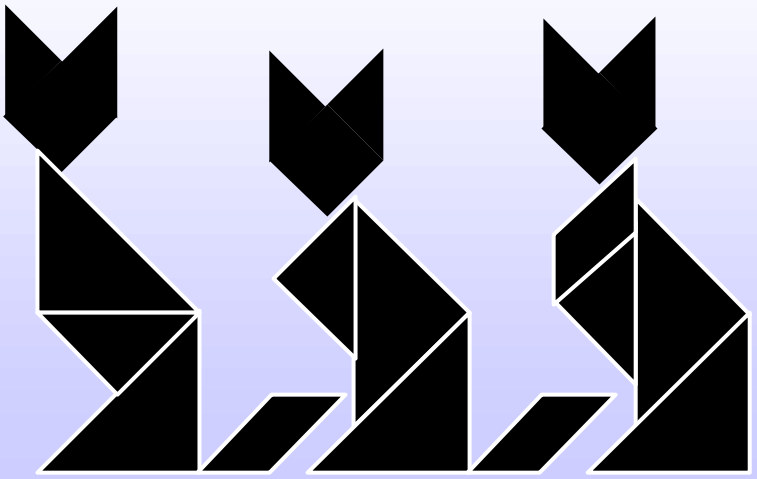
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Beispiele einer Systemfamilie



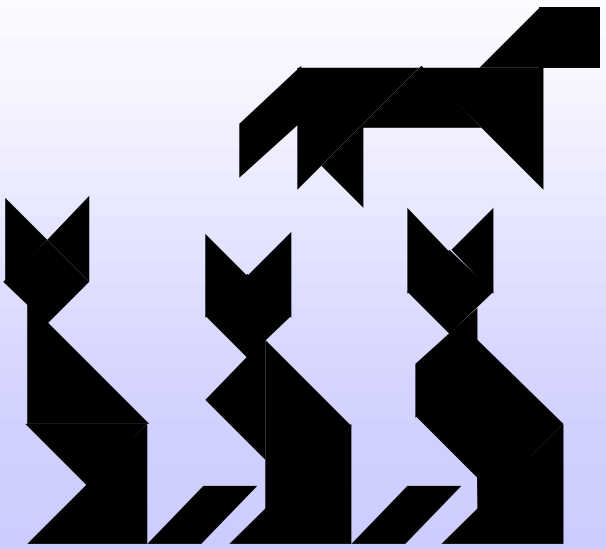
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Die "richtigen" Bausteine



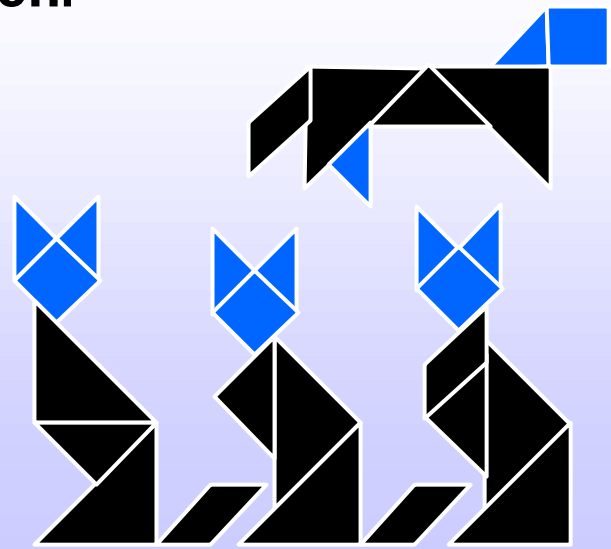
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Ein weiteres System ...



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

…kann zu Überraschungen führen!



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Wie werden Systeme bestellt?

- Angabe der Bausteine
  - Zwei gegeneinander stehende kleine Dreiecke oben mit eingepasstem kleinen Quadrat
  - Darunter ein liegendes großes Dreieck
  - …





- Beschreibung des Systems
  - Eine Katze
  - aufrecht sitzend
  - mit Schwanz



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Von der Bestellung zum Produkt



Bestellung aufgeben
- Fachsprache
- Formular, Katalog
- Beispiele benennen

Bestellung verarbeiten
- vervollständigen
- prüfen
- Bauanleitung
- optimieren



Produkt erzeugen
- wiederverwendbare, flexibel kombinierbare Bausteine montieren

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Generative Programmierung ...

... modelliert Softwaresystemfamilien so, dass ausgehend von einer Anforderungs-spezifikation mittels Konfigurationswissen aus elementaren, wiederverwendbaren Implementierungskomponenten ein hochangepasstes und optimiertes Zwischen- oder Endprodukt nach Bedarf automatisch erzeugt werden kann

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Industrielle Revolution



1826
austauschbare Teile
erfolgreich eingeführt von John Hall (nach 25 Jahren erfolgloser Versuche!)



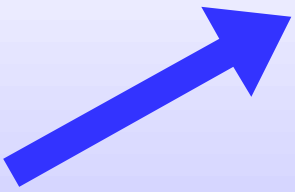
1901
Fließband
eingeführt von Ransom Olds; verbreitet und verbessert durch Henry Ford (1913)

1980
automatisierte Fließbänder
1961 erster Industrieroboter bei General Motors; Fortschritte durch Weiterentwicklung des Mikrochips (1970)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Entwicklung mit Komponenten

bisher



geeignete Komponente suchen, anpassen und von Hand montieren

benötigte Komponente spezifizieren und automatisch montieren

generative Programmierung

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Der Anwender sagt ...



... *was* er will

AUTO_GENERATOR<*EcoLine,Anhaengerkupplung*>::**RET**
LIST_GENERATOR<*Base,cp,poly,with_counter*>::**RET**
KONTO_GENERATOR<*Festgeld*<>,*EUR*>::**RET**
MATRIX_GENERATOR<*matrix*<*double,structure*
<*rect*<> > > >::**RET**

und erhält die **gewünschte Komponente**!

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

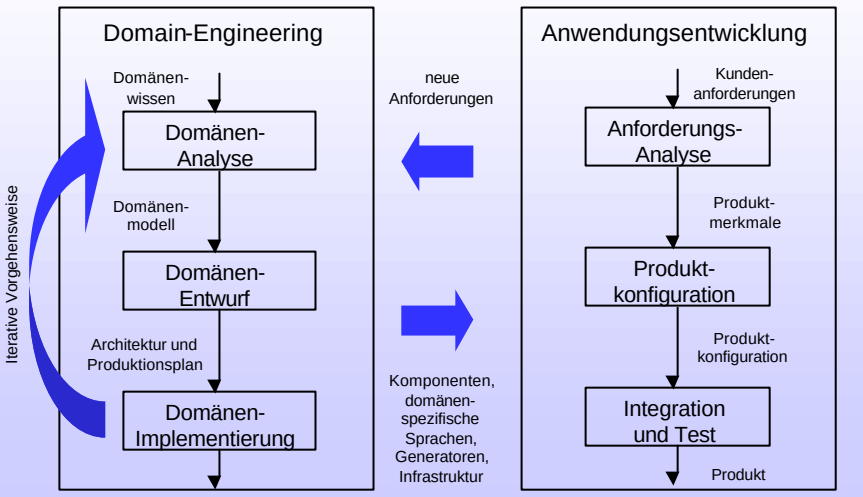
Systemfamilienentwicklung



- Domain Engineering
  - Analyse: Interessensbeteiligte, Fokussierung, variable und gemeinsame Merkmale, Abhängigkeiten
  - Entwurf: gemeinsame Systemarchitektur
  - Implementierung: wiederverwendbare Komponenten, domänenspezifische Sprachen, Konfigurationsgeneratoren
- Application Engineering
  - unter Verwendung der vorgenannten Ergebnisse werden konkrete Systeme und Komponenten anforderungsspezifisch produziert

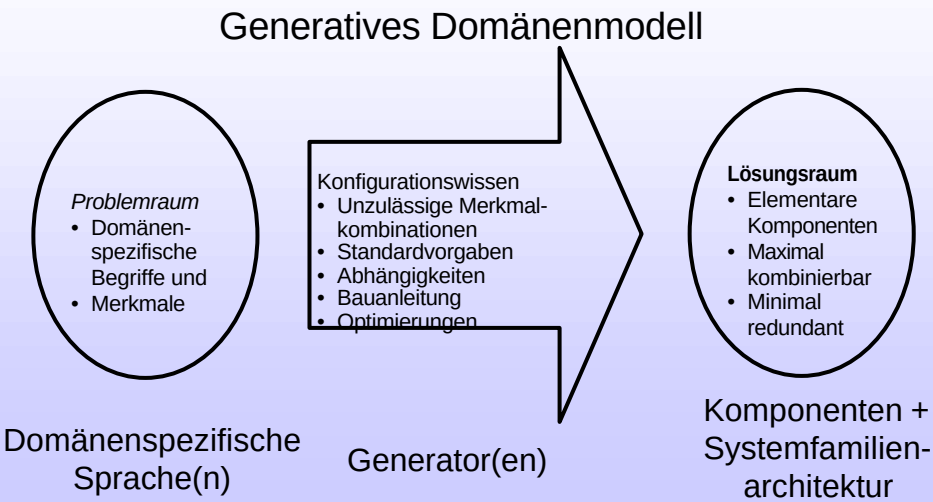
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zwei Prozesse



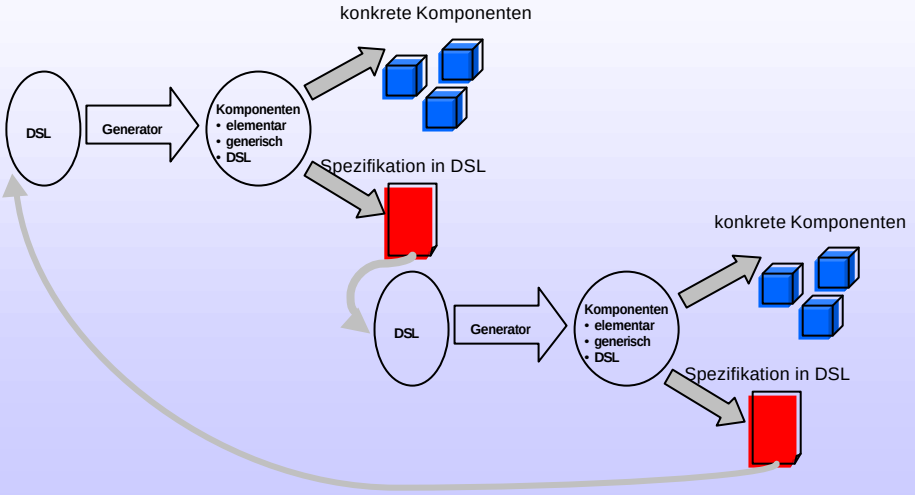
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Generatives Domänenmodell



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Kooperierende Generatoren



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Begriffe

Hier

- Domain Analysis: Bestandteil von Domain Engineering
- Produktfamilie: auf gemeinsamer Grundlage erzeugte Systeme
- Produktlinie: kundengruppen-spezifische Zusammen-stellung von Produkten

Abweichend

- Domain Analysis geht dem Domain Engineering voraus
- "Produktfamilie" und "Produktlinie" werden gleichbedeutend gebraucht

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Herausforderungen
- Einzelsystementwicklung
- Entwicklung von Systemfamilien
- ➢ Technikprojektionen
- Model Driven Architecture

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Eine Technikprojektion ist ...



eine Abbildung des generativen Domänenmodells auf

- andere Softwareentwicklungsmodelle,
- eine Programmiersprache oder auf
- mehrere Entwicklungswerkzeuge, die in einer Umgebung oder auf einer Plattform zusammengefügt sind

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Existierende Technikprojektionen



C++
- Template-Metaprogrammierung
- Statische Konfiguration
- Nur spracheigene Mittel

Java und JavaBeans
- Interaktive Parametrisierung von JavaBeans
- Dynamische Konfiguration
- Nur spracheigene Mittel

Java und AspectJ
- Einsatz eines AspectWeavers
- Statische Konfiguration
- Gesonderte Vorverarbeitungsstufe

ANGIE
- Einsatz eines Frame-Prozessors
- Statische Konfiguration
- Gesonderte Vorverarbeitungsstufe

Small Components
- Einsatz einer XML Konfigurationsdatei
- Statische und dynamische Konfiguration
- Gesonderte Vorverarbeitungsstufe

XML, TL und Ada 83
- grafisch-interaktive Konfiguration
- statische Konfiguration
- Gesonderte Vorverarbeitungsstufe

Nicht gegenseitig substituierbar!

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

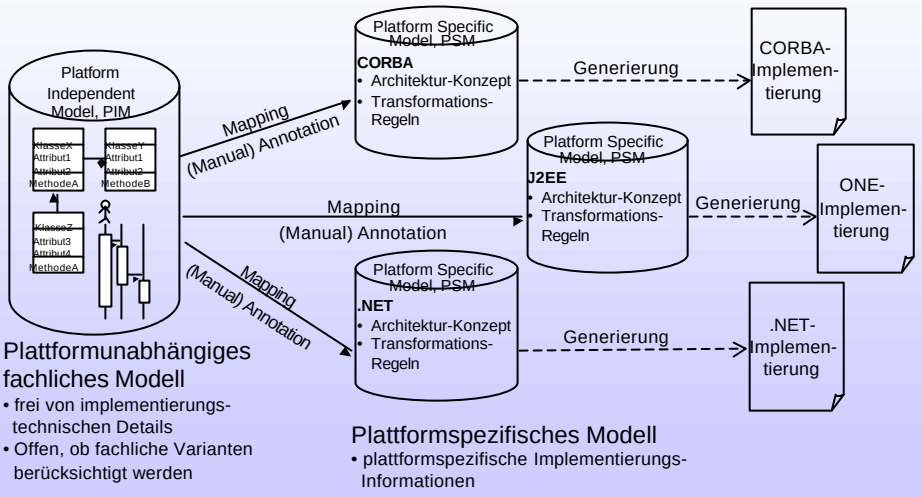
Übersicht

- Herausforderungen
- Einzelsystementwicklung
- Entwicklung von Systemfamilien
- Technikprojektionen

➢ Model Driven Architecture

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Model Driven Architecture



Plattformunabhängiges fachliches Modell
- frei von implementierungstechnischen Details
- Offen, ob fachliche Varianten berücksichtigt werden

Plattformspezifisches Modell
- plattformspezifische Implementierungs-Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Model Driven Architecture

- Fachliche Anpassungen
  - müssen auf fachlogischer Ebene des PIM erfolgen (Forward-Engineering)
  - dürfen nicht im PSM oder
  - in der Implementierung vorgenommen werden (kein Round-Trip-Engineering)
- Plattformspezifische Implementierungen eines Systems bilden eine Systemfamilie

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch



Übersicht

- ➢ Merkmalmodellierung
- • Implementierungskomponenten
- • Systemarchitektur
- • DSL-Entwurf
- • Generator

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Domain Engineering: Analyse

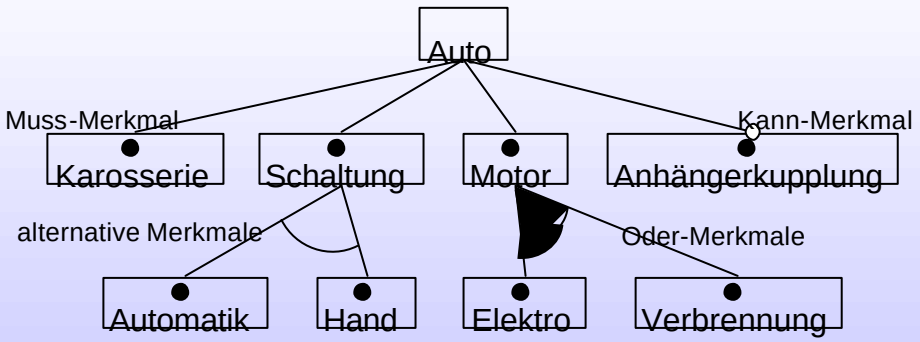


- Analyse der Interessensbeteiligten
- Fokussierung der Domäne
- Analyse von variablen und gemeinsamen Merkmalen und Abhängigkeiten
- Dokumentation mit Merkmalmodellen (FODA)
  - Merkmaldiagramme zur implementierungsneutralen und übersichtlichen Darstellung gemeinsamer und variabler Merkmale
  - Tabellen und freier Text für Abhängigkeiten, Wechselwirkungen, Kurzbeschreibung, Herkunft und Verwendung von Merkmalen, Standardvorgaben, Bindungszeit usw.

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Merkmaldiagramm

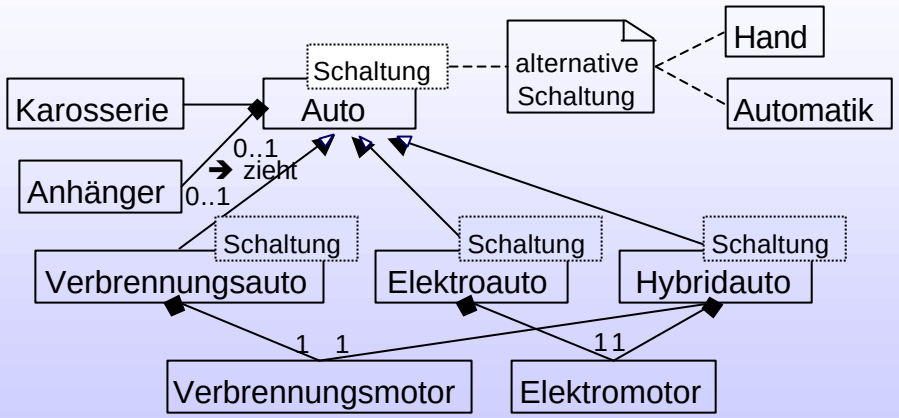




Keine Entscheidung darüber, wie das Merkmal und dessen Variabilität implementiert werden soll.

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Im Vergleich ein UML-Diagramm



Variabilität ist konkret implementiert, z.B. Vererbung, Aggregation und Assoziation!

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Wechselwirkungen

- Das Merkmaldiagramm zeigt
  2 (Schaltung)
  x 3 (Motor)
  x 2 (Anhängerkupplung)
  = 12 Kombinationen
- Einschränkung (separat dokumentiert): Elektromotor erfordert Automatik
- Somit ergeben sich 8 zulässige Kombinationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Merkmalmodellierung
- ➢ Implementierungskomponenten
- Systemarchitektur
- DSL-Entwurf
- Generator

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Domain Engineering: Entwurf



- Implementierungskomponenten identifizieren
- Äquivalenzkategorien bilden
- Abhängigkeitsanalyse
- gemeinsame Systemarchitektur
  - GenVoca-Schichtenarchitektur
  - GenVoca-Grammatik

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Äquivalenzkategorien



- Der nächste Schritt besteht in der Bildung von Äquivalenzkategorien
- Eine Äquivalenzkategorie umfasst Merkmale, die auf gegeneinander austauschbare Implementierungs-komponenten abgebildet werden
- Die Abhängigkeitsanalyse erfolgt dann auf der Ebene der Äquivalenzkategorien; dies ist eine Vereinfachung gegenüber der Betrachtung jeder einzelnen Implementierungskomponente

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

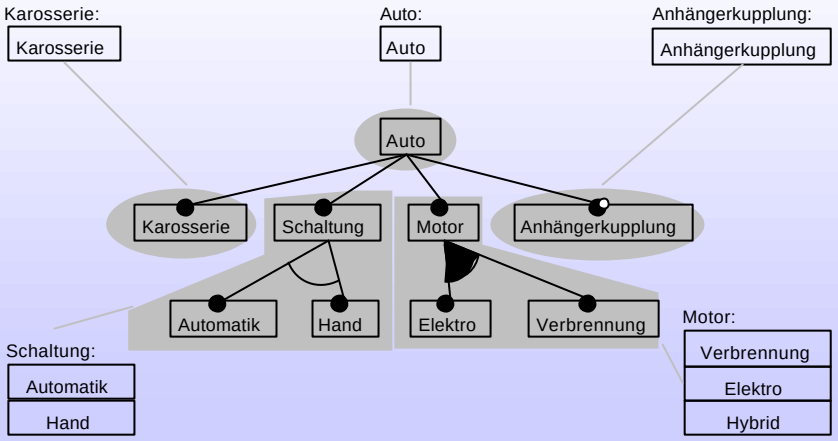
Äquivalenzkategorien



- In der Regel greift folgende Vorgehensweise
- Ein alleinstehendes konkretes Merkmal oder Untermerkmal bildet eine Äquivalenzkategorie mit nur einer Implementierungskomponente
- konkrete alternative Merkmale oder Untermerkmale bilden eine Äquivalenzkategorie
- konkrete Oder-Merkmale oder Untermerkmale bilden eine Äquivalenzkategorie

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Äquivalenzkategorien



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Implementierungskomponenten

Auto:
Auto

Karosserie:
Karosserie

Schaltung:
Automatik
Hand

Motor:
Verbrennung
Elektro
Hybrid

Anhängerkupplung:
Anhängerkupplung

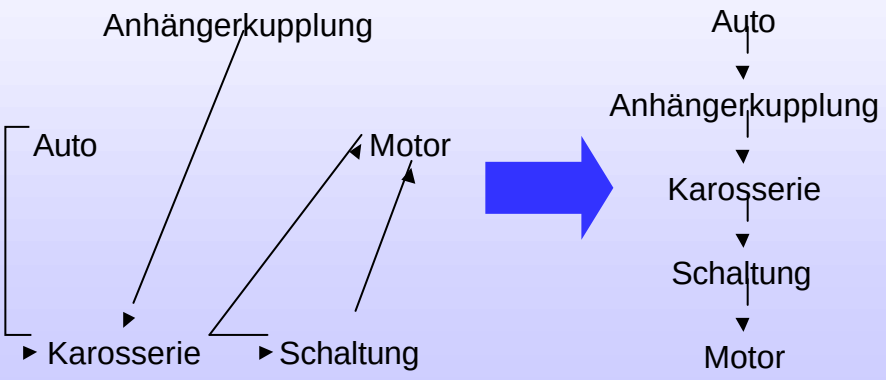
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zweck der Abhängigkeitsanalyse

- Die Abhängigkeitsanalyse identifiziert die Abhängigkeiten zwischen Implementierungskomponenten
- Sie ist die Grundlage zur Festlegung einer gemeinsamen Architektur für alle Systeme einer Familie
- Voraussetzung für die Abhängigkeitsanalyse sind die in Äquivalenzkategorien organisierten Implementierungs-komponenten
- Die Implementierungskomponenten müssen hierfür nicht konkret vorliegen!

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Abhängigkeitsanalyse



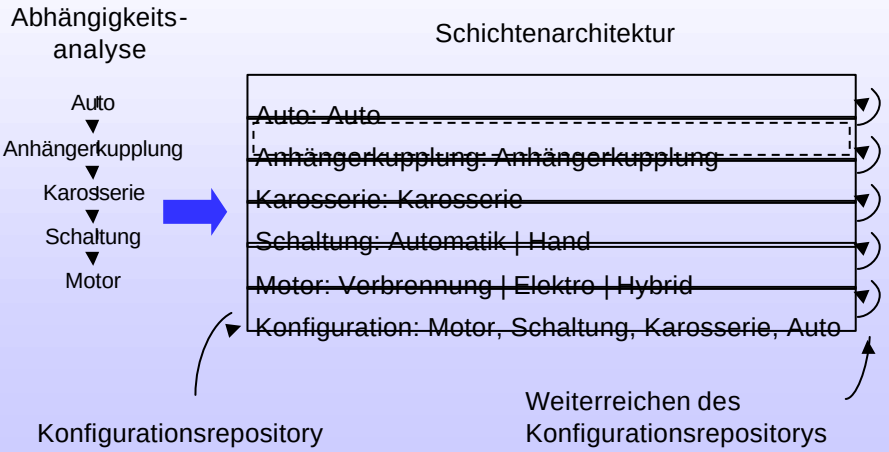
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht

- Merkmalmodellierung
- Implementierungskomponenten
- ➢ Systemarchitektur
- DSL-Entwurf
- Generator

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

GenVoca-Schichtenarchitektur



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

GenVoca-Grammatik



Auto:	Auto[Karosserie_mit_opt_AK]
Karosserie_mit_opt_AK:	Karosserie_mit_AK[Fertige_Karosserie] | Karosserie[Schaltung_mit_Motor]
Fertige_Karosserie:	Karosserie[Schaltung_mit_Motor]
Schaltung_mit_Motor:	Handschaltung[Motor] | Automatikschaltung[Motor]
Motor:	Verbrennungsmotor[Konfig] | Elektromotor[Konfig] | Hybridmotor[Konfig]

Konfig

Motor:	Verbrennungsmotor[Konfig] | Elektromotor[Konfig] | Hybridmotor[Konfig]
Schaltung:	Handschaltung[Motor] | Automatikschaltung[Motor]
Karosserie:	Karosserie[Schaltung_mit_Motor]
...	

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Fahrzeugkomponenten



```
template <class Konfig_> struct Verbrennungsmotor
{     typedef Konfig_ Konfig;
      Verbrennungsmotor() { cout << "Verbrennungsmotor "; }
};
template <class Motor> struct Automatikschaltung
{     typedef typename Motor::Konfig Konfig;
      enum { gaenge = Konfig::gaenge };
      Motor m;
      Automatikschaltung()
      { cout << gaenge << "-Gang Automatikschaltung ";}
};
template <class Schaltung> struct Karosserie
{     typedef typename Schaltung::Konfig Konfig;
      Schaltung s;
      Karosserie() { cout << "Karosserie ";}
}
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Konfiguration



```
template <class Karosserie> struct Auto
{     typedef typename Karosserie::Konfig Konfig;
      Karosserie k;
      Auto() { cout << "Auto " << endl; }
};

struct K1
{     enum { gaenge = 5 };
      typedef Verbrennungsmotor<K1> Motor;
      typedef Handschaltung<Motor> Schaltung;
      typedef Karosserie<Schaltung> Karosserie;
      typedef Auto<Karosserie> Auto;
};
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Manuelle Konfiguration



- 12 Konfigurationsrepositorys von Hand aufschreiben ist aufwendig und fehleranfällig
- Nur 8 Konfigurationsrepositorys beschreiben zulässige Konfigurationen
- Wie werden ausschließlich zulässige Konfigurationen erzeugt?

➡Konfigurationsgeneratoren

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Merkmalmodellierung
- Implementierungskomponenten
- Systemarchitektur
- ➢ DSL-Entwurf
- Generator

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Entwurfskriterien für DSLs



- Benutzbarkeit
  - Einfachheit: leicht verstehbar, erlernbar, anwendbar
  - Effizienz: mit möglichst wenig Sprachelementen möglichst viele Systeme "bestellen" können; für nicht-beschriebene Anteile gelten sinnvolle Standardvorgaben
  - Sicherheit: Die Wahrscheinlichkeit, ein ungültiges System "bestellen" zu können, ist möglichst klein
  - Offenheit: Falls nötig, sollten neue Sprachelemente leicht einzuführen sein
- Technische Realisierung
  - modularer Aufbau
  - leicht erweiterbar
  - leicht zu analysieren

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Grundlagen für den DSL-Entwurf

- Merkmaldiagramm
  - Wegfall von Merkmalen, die nicht spezifiziert werden müssen
  - Umwandlung mehrerer optionaler Merkmale in eine Dimension (besonders zur Vermeidung unzulässiger Spezifikationen)
  - Umordnung, so dass Merkmale, die häufiger spezifiziert werden vor denen stehen, die seltener spezifiziert werden
  - Einführen zusätzlicher abstrakter Merkmale
- Orientierung an vorhandenen DSLs, etwa Katalogen zum Bestellen von Autos
- Entwurf innovativer DSLs unter Einbeziehung der Anwender und Experten bei Berücksichtigung der Kriterien zur Benutzbarkeit

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Bestellung von Autos

	BaseLine	CityLine	EcoLine
Motor			
Otto	●		
Elektro		●	
Hybrid			●
Schaltung			
5-Gang-Handschaltung	●		
4-Gang-Automatik	❍	●	●
Zubehör			
Anhängerkupplung	❍	❍	❍

● Standardausstattung
❍ Zusatzausstattung (gegen Aufpreis)

Laut Prospekt kann kein Auto mit unzulässiger Konfiguration bestellt werden, also Elektromotor mit Handschaltung

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Domänenspezifische Sprache



```
// Umsetzung des Prospekts in eine domänenspezifische Sprache
// (Domain Specific Language, DSL)
enum Schaltung    { Fuenfgang, Viergangautomatik };
enum Modell       { baseline, cityline, ecoline }; // für interne Zwecke

template <int Schaltung_ = Fuenfgang>
struct BaseLine
{    enum   {    schaltung_ = Schaltung_,
                 modell_ = baseline
            };
};
// ...
enum Zubehoer     { ohne, Anhaengerkupplung };
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Merkmalmodellierung
- Implementierungskomponenten
- Systemarchitektur
- DSL-Entwurf
- ➢ Generator

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Konfigurationsgenerator



```
                // Spezifikation in domänenspezifischer Sprache
template <class Typ = BaseLine<>,int zubehoer_ = ohne>
struct AUTO_GENERATOR
{   // Kurznamen für Generator vereinbaren
    typedef AUTO_GENERATOR<Typ,zubehoer_> Generator;
    // Spezifikation parsen
    enum  {   modell_        = Typ::modell_,
              schaltung_ = Typ::schaltung_
          };
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Konfigurationsgenerator



```
// keine Baubarkeitsprüfung; DSL erlaubt keine falsche Spezifikation
// Komponenten montieren
enum { gaenge_ = (schaltung_ == Fuenfgang) ? 5 : 4 };
typedef SWITCH<modell_,
            CASE<baseline,Verbrennungsmotor<Generator>,
            CASE<cityline,Elektromotor<Generator>,
            CASE<ecoline,Hybridmotor<Generator>
          > > > >::RET Motor_;
typedef IF<(schaltung_ == Fuenfgang),
            Handschaltung<Motor_>,
            Automatikschaltung<Motor_> >::RET Schaltung_;
typedef IF<(zubehoer_ == Anhaengerkupplung),
            Karosserie_mit_AK<Karosserie<Schaltung_> >,
            Karosserie<Schaltung_> >::RET Karosserie_;
typedef Auto<Karosserie> RET; // fertiger Autotyp
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Template-Metaprogrammierung



- C++-Klassentemplates bilden eine Turing-vollständige, funktionale Programmiersprache (Unruh, Veldhuizen)
- C-ähnliche Kontrollstrukturen IF, SWITCH, DO, WHILE, FOR sowie EVAL_TABLE vorhanden
- Eisenecker, Czarnecki: Template-Metaprogrammierung - Eine Einführung, in OBJEKTspektrum 3/99
- Dies.: Template-Metaprogrammierung - Kontrollstrukturen, in OBJEKTspektrum 4/99
- Knaupp, Eisenecker, Czarnecki: Mit Tabellen zur Entscheidung, in OBJEKTspektrum 5/99
- Eisenecker, Czarnecki: Generatorbau in C++, in OBJEKTspektrum 6/99

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Konfigurationsgenerator



```
// Konfigurationsrepository erzeugen
struct Konfig
    {   typedef Motor_ Motor;
        typedef Karosserie_ Karosserie;
        // die folgende Information in Form von Konstanten
        enum  {   gaenge= gaenge_,
                  modell      = modell_,
                  schaltung   = schaltung_,
                  zubehoer    = zubehoer_,
              };
        typedef RET Auto;
    };
// Konfigurationsrepository ist Bestandteil von jedem Erzeugnis;
// kann z.B. von Generatoren wieder ausgelesen werden
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Nötige Änderungen



```
template <class Generator> // Generator statt Konfig
struct Verbrennungsmotor
{    typedef typename Generator::Konfig Konfig;
     Verbrennungsmotor()
     { cout << ”Verbrennungsmotor " << endl; }
};

// Elektromotor
...

// Hybridmotor
…
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Verwendung



```
typedef
    AUTO_GENERATOR<>::RET K1; // nur Standardvorgaben nutzen
typedef
    AUTO_GENERATOR<BaseLine<>,Anhaengerkupplung>::RET K2;
typedef
    AUTO_GENERATOR<BaseLine<Viergangautomatik> >::RET K3;

// ...

K1 a1; K2 a2; K3 a3;

// ...
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

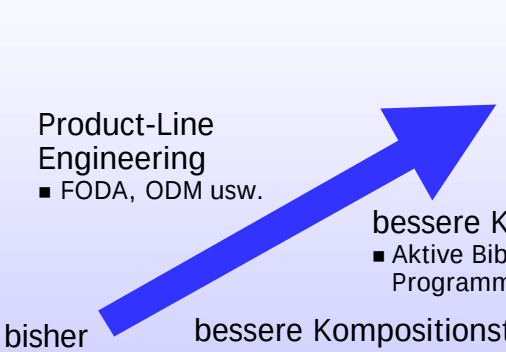


Übersicht

- ➢ Ausblick
- • Aktivitäten
- • Quellen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Industrielle Softwareproduktion



benötigte Komponente spezifizieren und automatisch montieren

generative Programmierung

Product-Line Engineering
- FODA, ODM usw.

bessere Kompositionsautomatisierung
- Aktive Biblitiotheken, erweiterbare Programmierungebungen usw.

bisher

bessere Kompositionstechniken
- AOP, SOP, APPC, CFs, Frames usw.

geeignete Komponente suchen, anpassen und von Hand montieren

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Thesen

- Eine Komponente ist stets Bestandteil eines wohldefinierten Fertigungsprozesses
- Werden Komponenten von Hand montiert, kann dies auch durch einen Generator automatisiert werden
- Generierungen sind statisch und dynamisch möglich
- Kriterien wie Binärformat, Interoperabilität, Sprachunabhängigkeit usw. sind nur relativ zum Fertigungsprozess relevant

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Ausblick
- ➢ Aktivitäten
- Quellen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Aktivitäten



- Generative Programming and Component Engineering (GPCE'03)
  22.-25.9.2003 in Erfurt
  in Verbindung mit NetObjectDays'03
  http://www.generative-programming.org
- Arbeitskreis "Generative und komponentenbasierte Softwareentwicklung"
  http://www.prakinf.tu-ilmenau.de/~czarn/generate

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Ausblick
- Aktivitäten
- ➢ Quellen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Informationen



- Bosch: Design and Use of Software Architecture: Adopting and evolving a product-line approach. Addison-Wesley, 2000
- Cleaveland: Program Generators with XML and Java. Prentice-Hall, 2001
- Clements, Northrop: Software Product Lines: Practices and Patterns. Addison-Wesley, 2001

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Informationen



- Czarnecki, Eisenecker: Generative Programming. Methods, Tools, and Applications, Addison-Wesley, 2000
- Donohoe (Hg.): Software Product Lines: Experience and Research Directions, Proceedings. Kluwer Academic Publishers, 2000
- Weiss, Lai: Software Product-Line Engineering. A Familiy-Based Software Development Process. Addison-Wesley, 1999

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Informationen im WWW



- *http://softwaregenerators.com*
- *www.aisto.com/roeder/active*
- *www.delta-software-technology.com*
- *www.generative-programming.org*
- *www.iese.fhg.de*
- *www.omniscium.com/nerdy/ip/*
- *www.sei.cmu.edu/plp/framework.html*
- *www.sei.cmu.edu/programs/pls/*

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Übersicht



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Teil 1

- Beginn 15^{30}h

- Einführung
- Beispiel
- Ausblick & Quellen

- Pause 17^{00}h -17^{30}h

Teil 2

- Beginn 17^{30}h

- Merkmalmodellierung
- GUI-Generierung
- Fallstudie GP-WEB

- Ende 19^{00}h

Merkmalmodellierung: Neuerungen und Werkzeuge





Markus Lang
Fachhochschule Kaiserslautern, PoLITe
prinz-william@gmx.de

Übersicht



- ➢ Forderungen an die Merkmalmodellierung
- Der Merkmalmodelleditor AmiEddi2
- Der Metamodelleditor AmiEddi2
- Der Spezialisierungseditor AmiEddi2
- Hinweise und Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Forderungen an die Merkmalmodellierung



- Erstellung grafischer Merkmalmodelle
- Abbildung der Merkmalmodelle auf eine interne Datenstruktur
- Erstellung domänenspezifischer Metamodelle für Zusatzinformationen zu Merkmalen
- Erfassung der im Metamodell ausgewählten Zusatzinformationen
- Export der Merkmaldiagramme in ein weiterverwendbares Datenformat (XML)

(Fähigkeiten von Ami Eddi 1.3)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Forderungen an die Merkmalmodellierung



- Modularisierung komplexer Merkmaldiagramme
- Flexible Merkmal-Kardinalitäten und Gruppen
- Wertzuweisung an Merkmale bei der Spezialisierung
- Erweiterbarkeit des Metamodells durch den Benutzer
- Spezialisierung von Merkmaldiagrammen und Erstellung konkreter Systemkonfigurationen

(Vorgaben für Ami Eddi 2)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

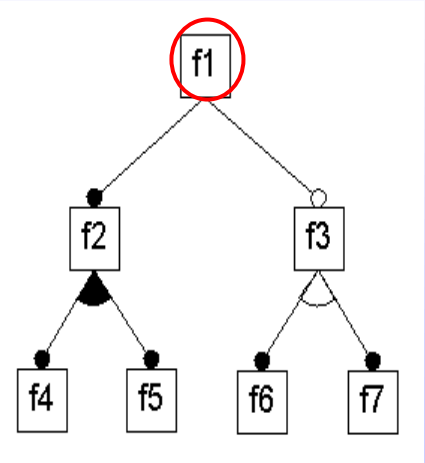
Übersicht



- Forderungen an einen Merkmalmodelleditor
- ➢ Der Merkmalmodelleditor AmiEddi2
  - Analyse eines Merkmaldiagramms
  - Das resultierende Fachkonzept
- Der Metamodelleditor AmiEddi2
- Der Spezialisierungseditor AmiEddi2
- Hinweise und Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Elemente eines Merkmaldiagramms



- Baumstruktur
- 1 Konzeptknoten
- Muss-Merkmale
- Kann-Merkmale
- Inclusiv-Oder-Gruppen
- Exclusiv-Oder-Gruppen
- zusätzl. Forderungen
  - flexible Gruppen
  - Merkmal-Kardinalitäten

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Das resultierende Fachkonzept in Prosa

- Merkmal
  - Beinhaltet alle Eigenschaften eines Merkmals
  - Wird selbst nicht in einem Diagramm verwendet
  - Wird in vielen Diagrammen referenziert
- Repräsentant
  - Eigentliches Merkmal in einem Diagramm
  - Zeigt auf ein Merkmal oder zur Modularisierung auf einen Konzeptknoten (Repräsentant)
  - Wertzuweisung bei Spezialisierung möglich

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Das resultierende Fachkonzept in Prosa



- Kardinalität
  - Gibt an, wie oft ein Repräsentant in einer Konfiguration erscheinen darf
  - früher nur 0-1, 1-1 (Kann-,Mussmerkmal)
  - jetzt flexible Kardinalitäten (Beispiel: 2-4)
- Gruppe
  - Fasst Kardinalitäten zu Gruppen zusammen
  - Gibt an, wie viele Kardinalitäten aus der Gruppe in einer Konfiguration erscheinen dürfen
  - früher nur Exklusiv- bzw. Inklusiv-Oder Gruppen
  - jetzt flexible Gruppen (Beispiel: 6 aus 49)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Das formalisierte Fachkonzept in EBNF



```
<merkmal>             := NAME DESCRIPTION [OPEN] [PREMATURE]
                         [<spezialisierungstyp>]
<spezialisierungstyp> := (BOOL | INT | FLOAT | STRING | POINTER)

<repräsentant>        := POINTER [<kardinalität> | <gruppe>]*

<kardinalität>        := MINIMUM [MAXIMUM] <repräsentant>

<gruppe>              := MINIMUM [MAXIMUM] <kardinalität>
                         <kardinalität>+
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Rückblick Forderungen

- Modularisierung komplexer Merkmaldiagramme ✓
- Flexible Merkmal-Kardinalitäten und -Gruppen ✓
- Wertzuweisung an Merkmale bei der Spezialisierung ✓
- Erweiterbarkeit des Metamodells durch den Benutzer ✗
- Spezialisierung von Merkmaldiagrammen und Erstellung konkreter Systemkonfigurationen ✗

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht

- Forderungen an die Merkmalmodellierung
- Der Merkmalmodelleditor AmiEddi2
- ➢ Der Metamodelleditor AmiEddi2
  - Metamodell durch Merkmaldiagramme
  - Erweiterung des Metamodells
- Der Spezialisierungseditor AmiEddi2
- Hinweise und Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

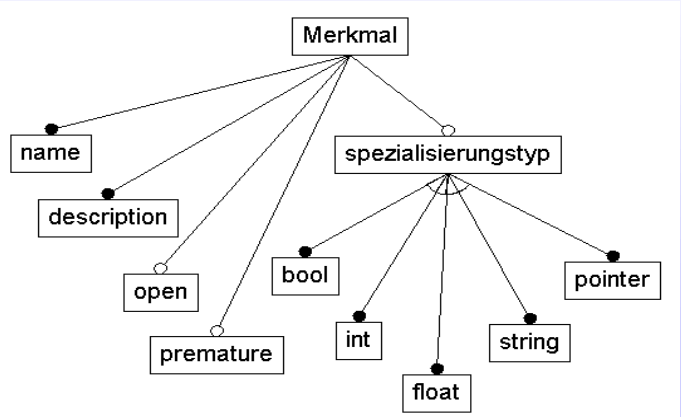
Metamodell durch Merkmaldiagramme

- Idee
  - Die Bestandteile des Fachkonzept des Merkmalmodelleditors können selbst durch Merkmaldiagramme beschrieben werden
  - Sie bilden das Grundgerüst des Metamodells
- Folge
  - Der Benutzer kann mit dem Merkmalmodelleditor das Metamodell um Zusatzinformationen erweitern
  - Dies macht die Merkmalmodellierung zu einem in sich geschlossenen und orthogonalen System

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Metamodell durch Merkmaldiagramme

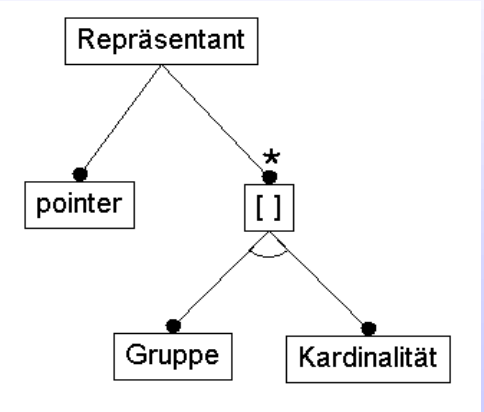
<merkmal> := NAME DESCRIPTION [OPEN] [PREMATURE] [<spezialisierungstyp>]



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Metamodell durch Merkmaldiagramme

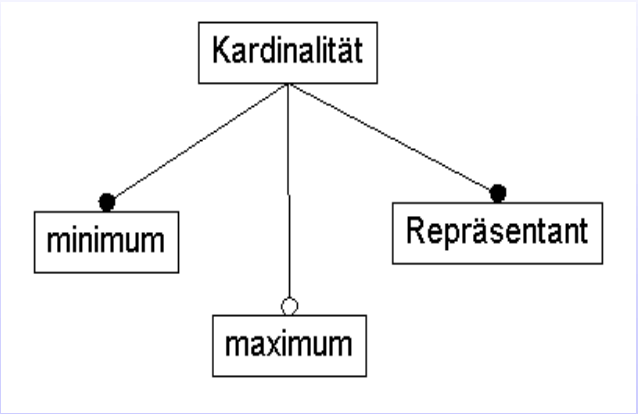
<repräsentant> := POINTER [<kardinalität> | <gruppe>]*



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Metamodell durch Merkmaldiagramme

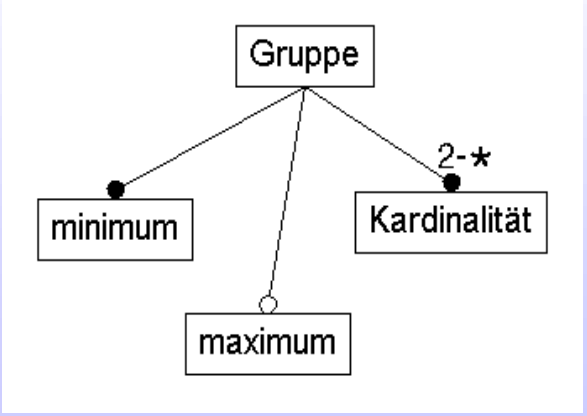
<kardinalität> := MINIMUM [MAXIMUM] <repräsentant>



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

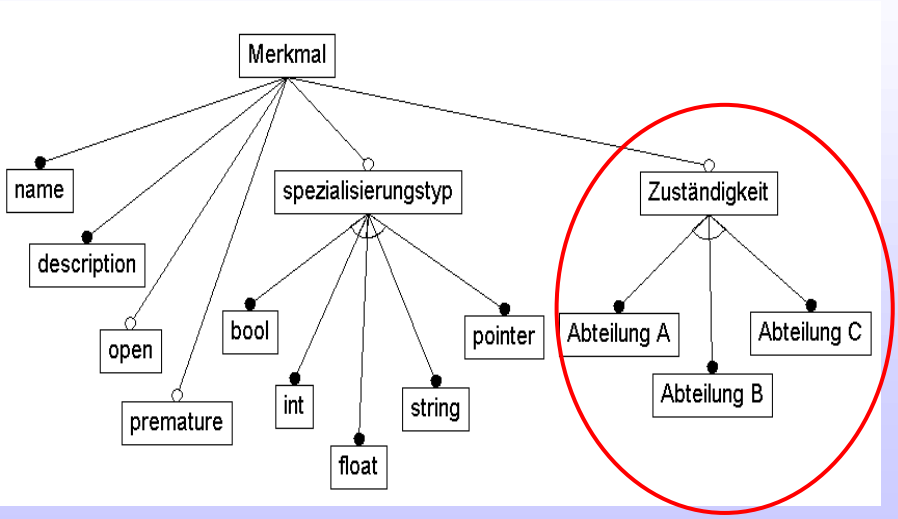
Metamodell durch Merkmaldiagramme

<gruppe> := MINIMUM [MAXIMUM] <kardinalität><kardinalität>+



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Erweiterung des Metamodells (Beispiel)



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Forderungen an die Merkmalmodellierung
- Der Merkmalmodelleditor AmiEddi2
- Der Metamodelleditor AmiEddi2
- ➢ Der Spezialisierungseditor AmiEddi2
  - Was geschieht bei der Spezialisierung?
  - Spezialisierung des Metamodells
  - Spezialisierung im Überblick
- Hinweise und Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

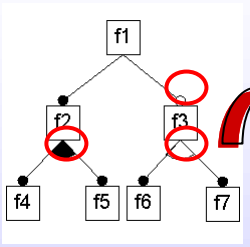
Was geschieht bei der Spezialisierung?



- Einschränkung der Variabilität eines Merkmaldiagrams
  - Gruppen werden aufgelöst
  - Kardinalitäten werden exakt bestimmt
  - Merkmale erhalten bei Bedarf Werte
- Spezialisierung eines Merkmaldiagramms führt zu einem neuen Merkmaldiagramm
- Besitzt ein Merkmaldiagramm keine Variabilität mehr, so ist es vollständig spezialisiert (Konfiguration)

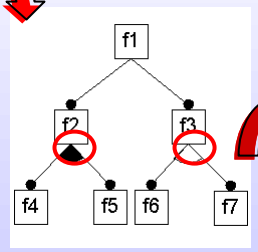
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Was geschieht bei der Spezialisierung? (Beispiel)



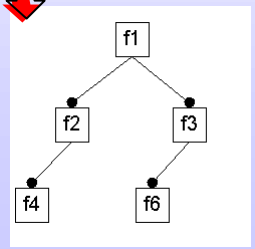
Variabilität:
- f3 optional
- Inklusiv-Oder-Gruppe f4/f5
- Exklusiv-Oder-Gruppe f6/f7

Teilweise Spezialisierung



Variabilität:
- Inklusiv-Oder-Gruppe f4/f5
- Exklusiv-Oder-Gruppe f6/f7

Vollständige Spezialisierung (Konfiguration)



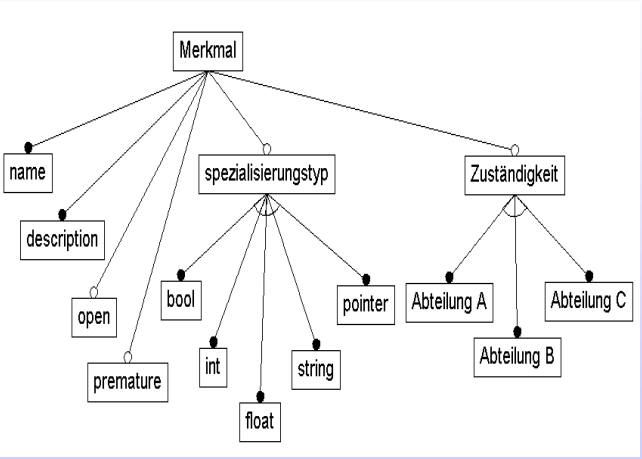
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Spezialisierung des Metamodells

- Da das Metamodell selbst durch Merkmaldiagramme beschrieben, ist kann auch hier die Spezialisierung angewandt werden
- Die Spezialisierung der Systemdiagramme führt zu den Bausteinen, aus denen sich ein Merkmaldiagramm zusammensetzt

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

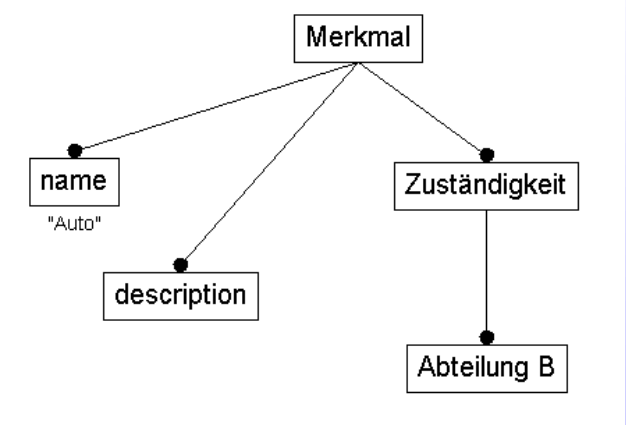
Spezialisierung des Metamodells (Beispiel)



Bei der Spezialisierung dieses System-diagramms könnte zum Beispiel ein Merkmal mit dem Namen „Motor“, entstehen, für das Abteilung B zuständig ist

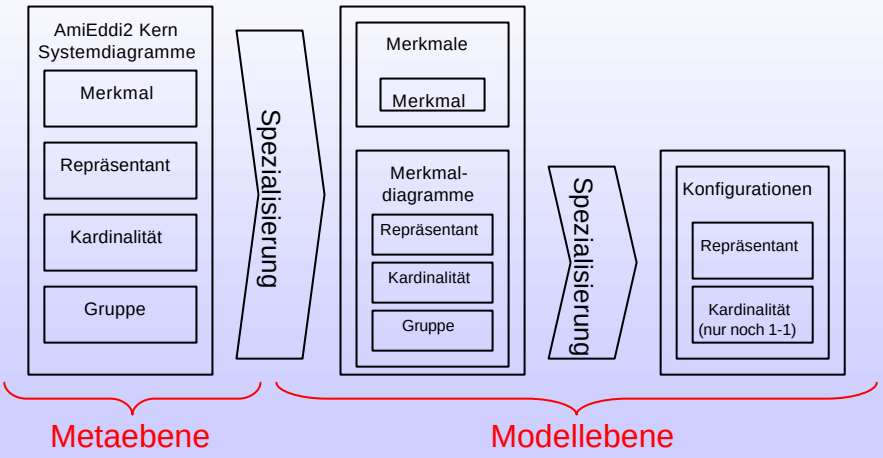
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Spezialisierung des Metamodells (Beispiel)



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Spezialisierung im Überblick



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Rückblick Forderungen

- Modularisierung komplexer Merkmaldiagramme ✓
- Flexible Merkmal-Kardinalitäten und -Gruppen ✓
- Wertzuweisung an Merkmale bei der Spezialisierung ✓
- Erweiterbarkeit des Metamodells durch den Benutzer ✓
- Spezialisierung von Merkmaldiagrammen und Erstellung konkreter Systemkonfigurationen ✓

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Forderungen an die Merkmalmodellierung
- Der Merkmalmodelleditor AmiEddi2
- Der Metamodelleditor AmiEddi2
- Der Spezialisierungseditor AmiEddi2
  - Was geschieht bei der Spezialisierung?
  - Spezialisierung des Metamodells
  - Spezialisierung im Überblick

➢ Hinweise und Informationen

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Hinweise



- Die Entwicklung von **AmiEddi-XML** wurde durch die Delta Software Technology GmbH (*www.d-s-t-g.com*) gefördert (Bearbeiter: Markus Lang)
- Die Entwicklung von **AmiEddi2** (Arbeits-bezeichnung) ist ein Gemeinschaftsprojekt der DaimlerChrysler AG, Research and Technology, Ulm und der FH Kaiserslautern, Zweibrücken (Projektgruppe: Thomas Bednasch, Markus Lang, Henrik Becker)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Weiterführende Informationen



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

- M. Selbig: A Feature Diagram-Editor - Analysis, Design, and Implementation of its Core Functionality. Diplomarbeit Fachbereich I/MST, FH Kaiserslautern, Zweibrücken 2000
- F. Blinn: Entwurf und Implementierung eines Generators für Merkmalmetamodelle. Diplomarbeit Fachbereich I/MST, FH Kaiserslautern, Zweibrücken 2001
- U. W. Eisenecker, M. Selbig, F, Blinn, K. Czarnecki: Merkmalmodellierung für Softwaresystemfamilien. In: OBJEKTspektrum 5/2001
- Th. Bednasch: Konzept und Implementierung eines konfigurierbaren Metamodells für die Merkmalmodellierung. Diplomarbeit Fachbereich I/MST, FH Kaiserslautern, Zweibrücken, in Vorbereitung

Merkmalmodellbasierte GUI-Generierung





Max Schlee

Fachhochschule Kaiserslautern, PoLITe

schlee2@web.de

Überblick



- ➢ GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ABA-Details
- Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

GUI-Generator wofür?



- Steigerung der Produktivität
- Berücksichtigung ergonomischer Richtlinien
- Einheitlichkeit von generiertem Code
- Vermeidung von Fehlern
- Kostenminimierung
- Schnelle Prototyperstellung

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Überblick



- GUI-Generator wofür?
- ➢ Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ABA-Details
- Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Stand der Technik



Modellbasierte GUI-Generatoren

- Beispiele
  - JANUS (OO-Modell)
  - GENIUS (ER-Modell)
- Veränderungen am Code bleiben erhalten
- Wissen über das verwendete Modell ist erforderlich

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Stand der Technik



Nicht modellbasierte GUI-Generatoren

- Beispiele
  - AppWizard
  - AppExpert
- Sehr einfach in der Handhabung
- Code wird durch neue Generierung überschrieben

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Überblick



- GUI-Generator wofür?
- Stand der Technik
- ➢ ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ABA-Details
- Quellen
- Demonstration

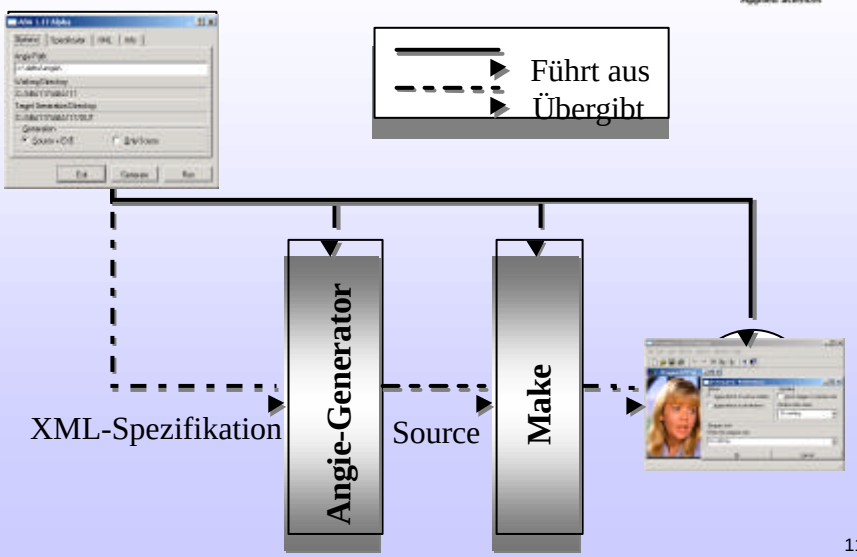
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

ABA

- Modellbasiert (Merkmalmodell)
- Es wird nur das generiert, was von dem GUI-Anteil und Funktionen benötigt ist
- Die verwendete Komponenten können von dem Designer entworfen werden
- Sehr einfache Handhabung
- Veränderungen am Code bleiben erhalten (zur Zeit nicht automatisiert)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Generierungsprozess in ABA



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

ABA-Demonstration 1

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

ABA-Systemvorausetzungen

- Windows 9.x, NT, 2k, XP
- ANGIE R2.1
- Qt 2.3.0
- Visual C++ 6.0

Überblick



- GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- ➢ Frameprozessor ANGIE
- Qt
- ABA-Details
- Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

ANGIE



- Frame Prozessor
- Delta Software Technology GmbH, *http://angie.d-s-t-g.com*
- Visual Basic ähnlich
- Grosse Auswahl von Skriptfunktionen
- Unabhängig von Programmiersprachen
- Erzeugung von nicht binären nicht-Code-Artefakten ist möglich, etwa Hilfe-Dateien, Ressourcen usw.
- Plattformabhängig (Windows)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Wichtige Elemente



- Modularisierung mit Deklarations- und Code-Modulen
- Skript-Funktionen
- Variablen-Container
- Frame-Deklarationen
- Datendeklarationen global und lokal
- Arrays, Collections (mit Schlüssel)
- Alle notwendigen imperativen Anweisungen, Zuweisungen und Funktionsaufrufe
- Intrinsische Funktionen, zum Beispiel für Strings, Arrays, Collection usw.

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Frame



- Repräsentation eines Frames

```
.Frame MyFrame (myParameter)
…
.End Frame
```

- Erzeugen einer Frameinstanz

```
.frMyFrame = createFrame(„MyFrame“, “parameter”)
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Slot (Einfügepunkt)



- Einbettung des Slots
  (<!!> vom Benutzer definierbar)

```
...
<!mySlot!>
...
```

- Optionale Codeblöcke
  (<??> vom Benutzer definierbar)

```
...
<? CodeBlock <!mySlot!>?>
...
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Kontrollstrukturen



- Verzweigungen

```
.If  a = b Then
…
.Else
```

- Wiederholungen

```
.For i = 0 To 10
…
.Next
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Collections



- Syntax
  `.Dim myCollection = createCollection()`
- Collection Funktionen
  - createCollection
  - Add
  - Remove
  - RemoveAll

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Weiteres



- Variabelendefinition:
  `.Dim myVariable As Typ`
- Skript-Funktionen
  `.Function myFunction`
  `…`
  `.End Function`
- Export einer Frameinstanz in eine Datei
  `.export(frMyFrame, „FileName“, „FileType“)`

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Projekte



- Backend für Pattern by Example
- Backend für SCORE
- GP-WEB

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Überblick



- GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE

➢ Qt

- ABA-Details
- Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Qt (nur als Beispiel)



- Entwickelt von Trolltech, *www.troll.no/*
- Basis für KDE (Linux)
- Multiplattformfähig
- XML-Unterstützung
- Mehr als 200 Klassen
- Qt 2.3.0 für Windows freiverfügbar
- Für die freiverfügbare Version ist nur Visual C++ geeignet

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

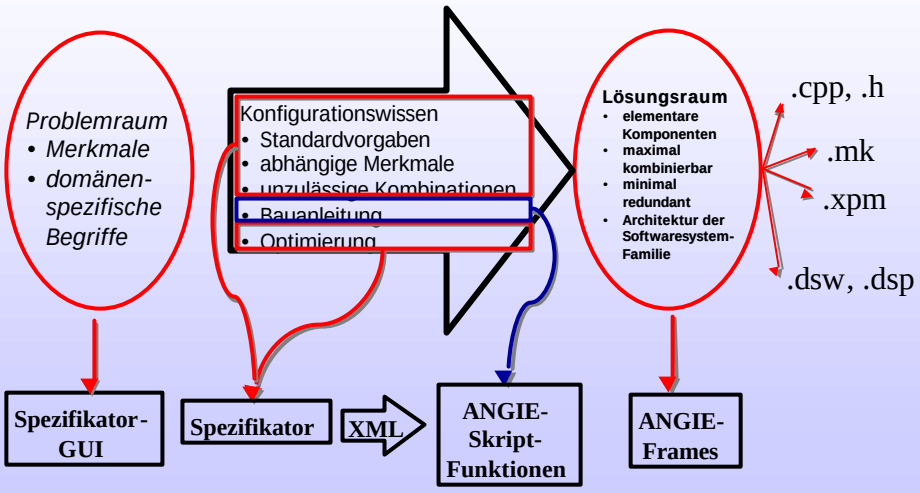
Überblick



- GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ➢ ABA-Details
- Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Generatives Domänenmodell



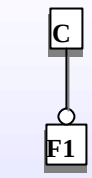
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

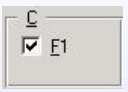
Abbildung des Merkmalmodells auf GUI und Spezifikator

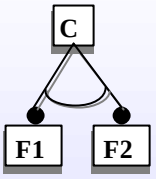
- Automatisierung der Spezifikationserstellung
- Verwaltung von Abhängigkeiten
- Verwaltung unsinniger Eingaben
- Bessere Wahrnehmung als Text
- Keine XML-Kenntnisse erforderlich
- Keine Kenntnisse über Logik und getroffenen Vereinbarungen erforderlich
- Keine Baubarkeitsprüfung nötig
- Lernen durch Ausprobieren

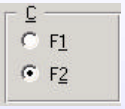
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

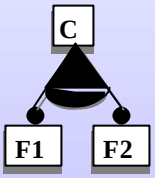
Abbildung des Merkmalmodells auf GUI und Spezifikator

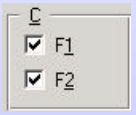










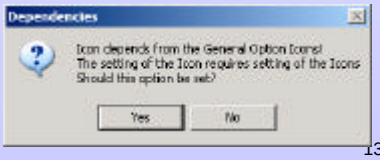


Dependencies

1.

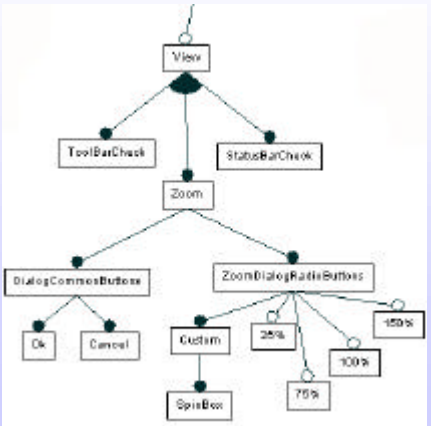


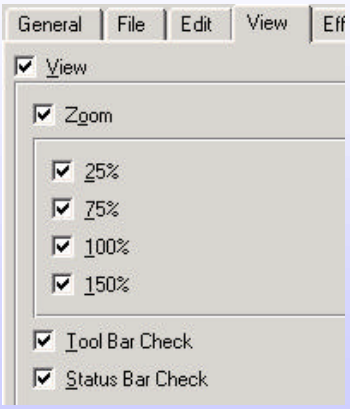
2.



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

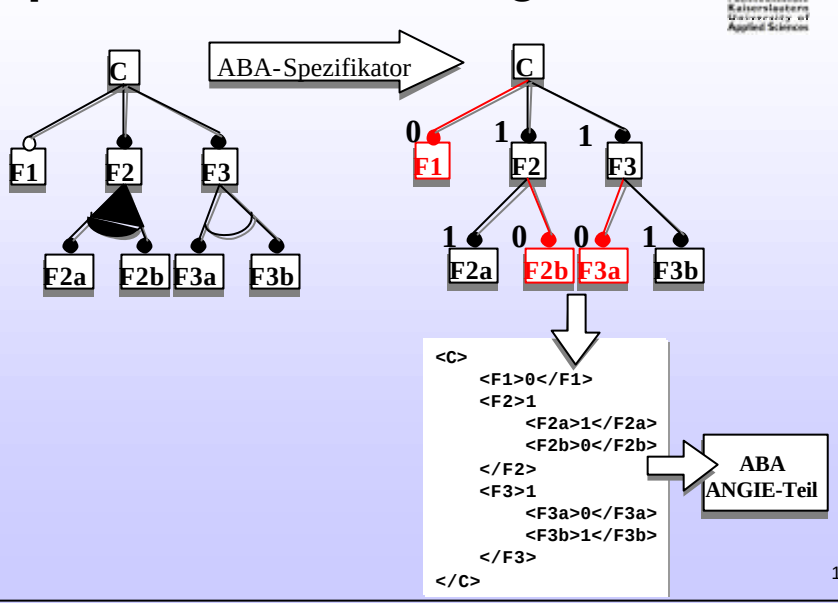
Abbildung des Merkmalmodells (Beispiel)





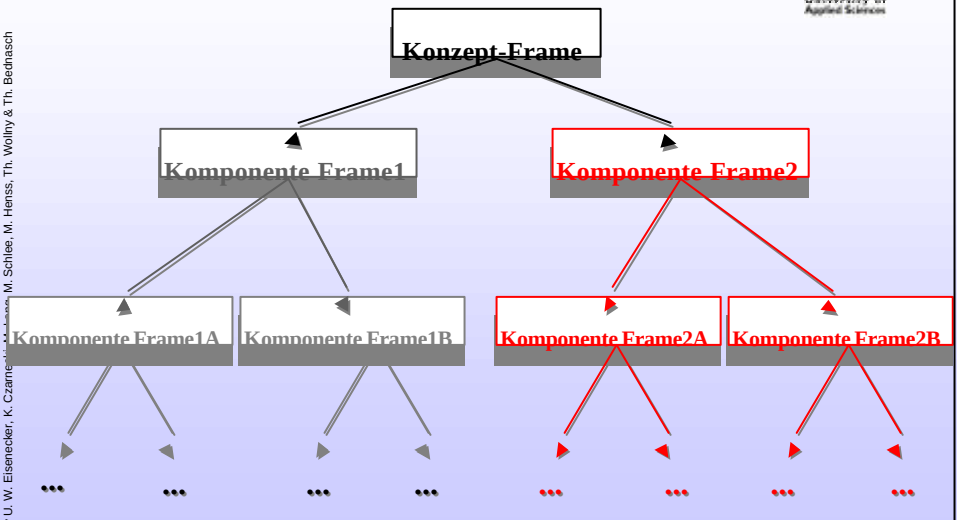
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Spezifikationserstellung in ABA



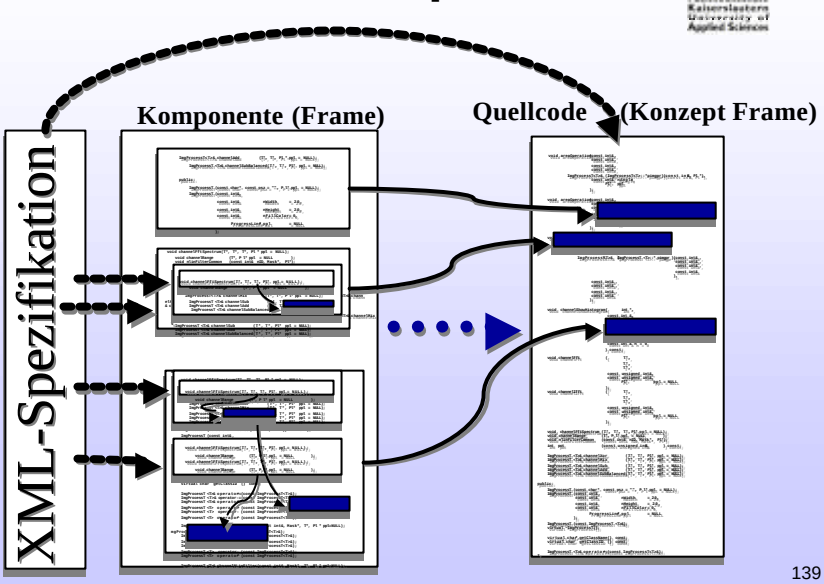
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Erzeugen von Frames



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Einbauen einer Komponente



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Realisierung (ANGIE)

```
.Function main() As Variant
    .Dim colSpec = createCollection()

    .Dim xmlhandle = xmlLoad("spec", "SpecXML")
    .Dim i = 0
    .For i = 0 To xmlCount(xmlhandle)
        .add(colSpec,                                _
             xmlGetElementText(xmlhandle, i), _
             xmlGetElementName(xmlhandle, i)  _
            )
    .Next i
    .xmlClose(xmlhandle)
    .createApplicationH(colSpec)
...
...
.End Function
```

```
.Frame ApplicationH(colSpec As Collection), Column Off
    .Dim nTempValue1  = CInt(colSpec("effects"))
    .Dim frEffectsH1  = (nTempValue1 = 1)                    _
                        ? createFrame("EffectsH1", colSpec) _
                        : EMPTY
...
...
<!frEffectsH1!>
...
...
.End Frame
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Merkmale des erzeugten Systems



- Für das bestellte System gilt:
  - Freischaltung von den Funktionen nicht möglich
  - Ungeeignet für den anderen Benutzer

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Überblick



- GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ABA-Details
- ➢ Quellen
- Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Quellen



- Krzysztof Czarnecki, Ulrich W. Eisenecker; Generative Programming. Methods, Tools, and Applications. Addison-Wesley, 2000
- Sven Macke; Generative Programmierung mit AspectJ. Diploma thesis, University of Applied Sciences Kaiserslautern, Zweibrücken 2001
- ANGIE Online Help, Copyright (c) 2000 - 2002 Delta Software Technology GmbH

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Überblick



- GUI-Generator wofür?
- Stand der Technik
- ABA-Überlick
- Frameprozessor ANGIE
- Qt
- ABA-Details
- Quellen

➢ Demonstration

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

ABA-Demonstration 2

Fallstudie GP-WEB

Eine E-Learning-Systemfamilie

Mathias Henss

Fachhochschule Kaiserslautern, GP-WEB

mat_tes@web.de

Übersicht Design und Entwurf



- Einführung
- Domänenanalyse
- Domänenentwurf
- Klassifizierung von Merkmalen
- Technikanalyse

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung



- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- ➢ Einführung
- Domänenanalyse
- Domänenentwurf
- Klassifizierung von Merkmalen
- Technikanalyse

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Einführung



- „GP für web-orientierte Softwaresystemfamilien“
  - IDEE: Funktionsfähiges web-orientiertes System automatisch erstellen
  - ZIEL: Technische und methodische Voraussetzungen hierfür schaffen
- Die Projektumsetzung erfolgt anhand der Beispieldomäne E-Learning/Web-Learning

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Übersicht



- Einführung
- ➢ Domänenanalyse
- Domänenentwurf
- Klassifizierung von Merkmalen
- Technikanalyse

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Domänenanalyse



- Brainstorming
- Literaturstudien
  - diverse Skripte
  - Selektions- und Entscheidungskriterien für die Auswahl von Lernplattformen und Autorenwerkzeuge (Autor: Prof. Dr. Rolf Schulmeister)
- Interviews
  - mit Firmen
  - mit Interessenbeteiligten
- Internetrecherche
  - allgemein zum Thema E-Learning
  - produktive E-Learning-Systeme (WebCT, academy-now ...)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Domänenanalyse



- Die Ergebnisse wurden schriftlich fixiert
- Strukturierung der Informationen
- Informationen mit Quellenangaben archiviert
  - Internetlinks
  - Mails
  - Skripte
  - Artikel aus Fachzeitschriften

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Einführung
- Domänenanalyse
- ➢ Domänenentwurf
- Klassifizierung von Merkmalen
- Technikanalyse

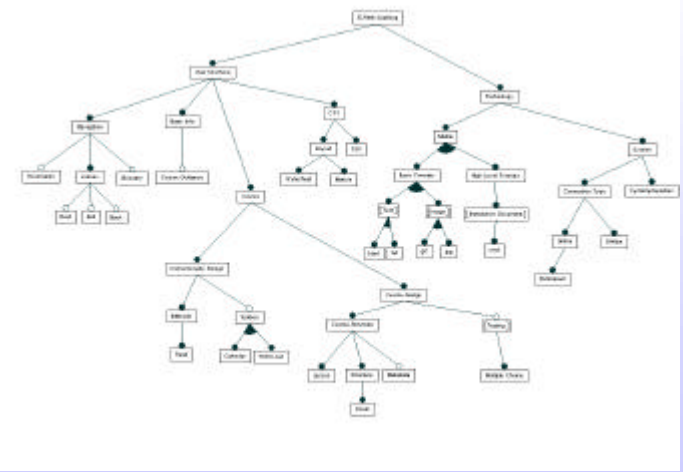
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Domänenentwurf

- Merkmalmodellierung
- Kurze Erläuterungen zu den Merkmaldiagrammen
- Vorstellung der gesamten Softwaresystemfamilie anhand des Merkmalmodells
- Erläuterungen zum Minimalsystem

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Merkmaldiagramm



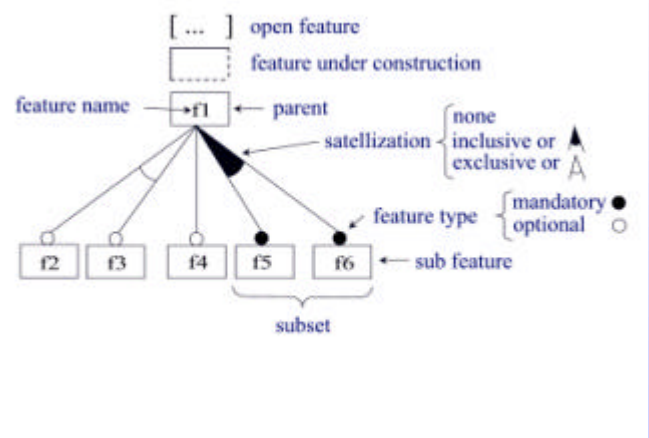
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Merkmalmodellierung

- Gruppieren der Informationen zur Merkmalsfindung
- Abhängigkeitsanalyse
- Klassifizierung
- Aufstellen des Merkmaldiagramms

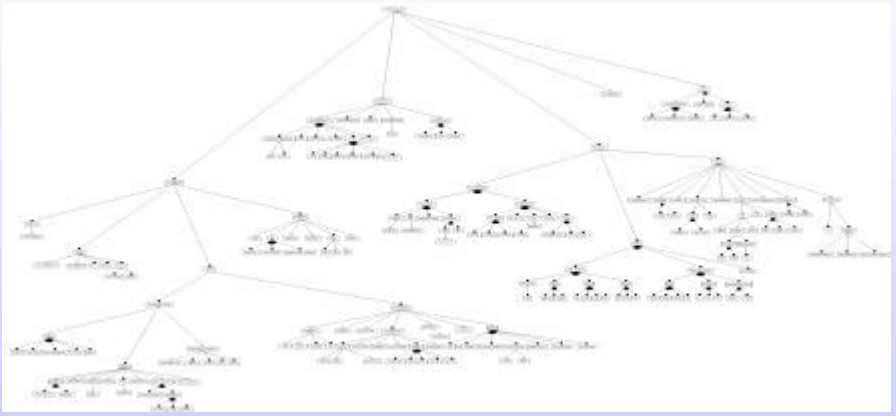
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Zeichenerklärung für das Merkmaldiagramm



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

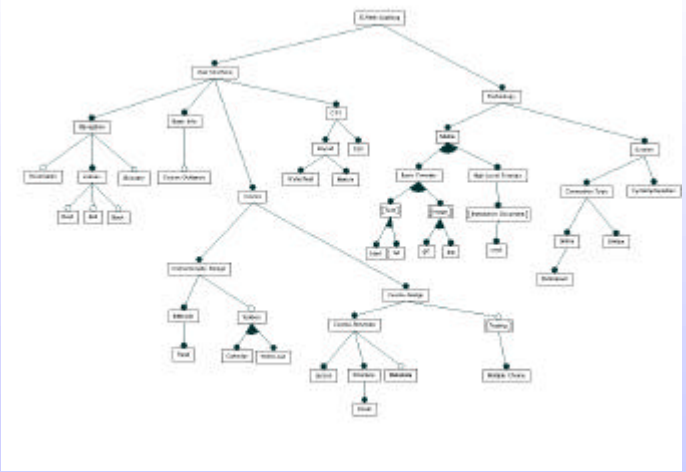
Merkmaldiagramm E-Learning-System



Originalbild zu finden unter *www.gp-web.org*

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Merkmaldiagramm Minimalanforderung



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht



- Einführung
- Domänenanalyse
- Domänenentwurf
- ➢ Klassifizierung von Merkmalen
- Technikanalyse

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Klassifizierung von Merkmalen



- Konkretes Merkmal
  - wird direkt als Komponente implementiert
- Aspektmerkmal
  - wird mit anderen Merkmalen verbunden und besitzt eine oder mehrere Implementierungen
- Abstraktes Merkmal
  - besitzt keine eigene Implementierung, wirkt sich aber auf die Konfiguration anderer Merkmale aus

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Übersicht



- Einführung
- E-Learning-System
- Domänenanalyse
- Domänenentwurf
- Klassifizierung von Merkmalen
- ➢ Technikanalyse

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Technikanalyse



- Plattform (UNIX, Win2000/XP, IIS5.0, Apache)
- Gegenüberstellung unterschiedlicher Implementierungssprachen und Konzepte sowie deren Bewertung
  - Laufzeitumgebung (MS.NET oder J2EE)
  - Skripting (VBscript, Javascript)
  - Programmiersprache (C#, Java, VB.NET, J#)
- Auswahl einer geeigneten Generatorsprache und einer geeigneten Zielsprache
  - Frameprozessortyp (ANGIE, XVCL)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Übersicht Implementierung



- ➢ Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Generatoranforderungen



- Flexibles Zusammenspiel der einzelnen Komponenten
  - Script-Integration und Komposition
- Wartung der einzelnen Merkmale soll auch für Betreuer später leicht möglich sein
  - Auslagerung einzelner Implementierungskomponenten in eine externe XML-Datei
  - Systemvariablen mit Hilfe von Wildcards in Komponenten einbinden
- Leichte Erweiterung des Generators
  - interne Struktur flexibel gestalten

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung



- Generatoranforderungen
- ➢ E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Anforderungen



- Übersichtliche, einheitliche Einteilung für das Web-Frontend
- Intuitive Bedienung
- Clientseitige Plattform-Unabhängigkeit
- Allgemeine Browserunterstützung
  - Berücksichtigung alter wie aktueller Browser
  - Unabhängigkeit von einzelnen Browsern wie Netscape Navigator, Opera, Mozilla oder MS Internet Explorer
- Ohne Internetanbindung lauffähig

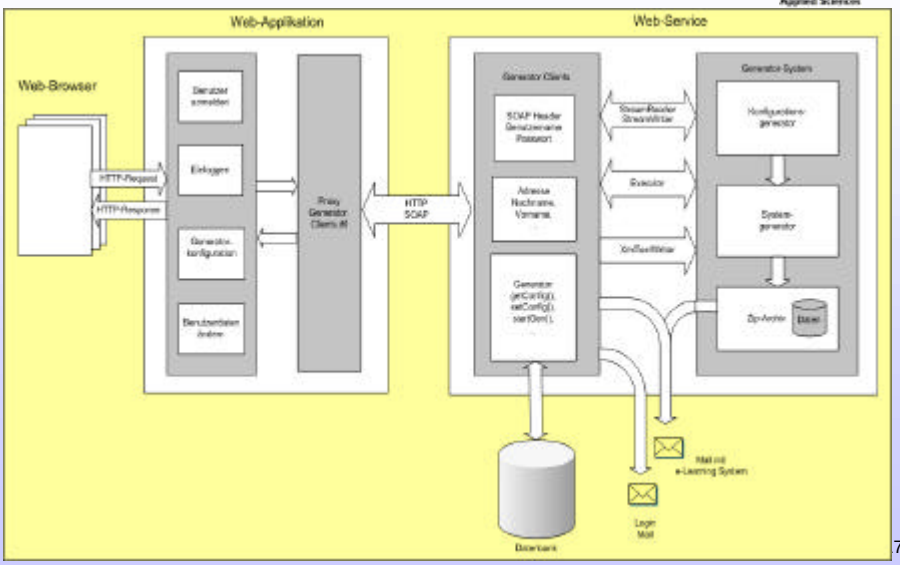
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung

- Generatoranforderungen
- E-Learning-System Anforderungen
- ➢ Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service

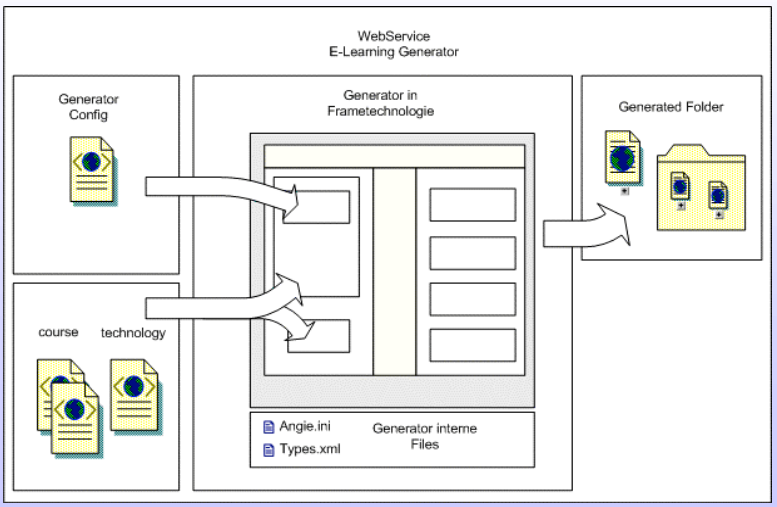
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Gesamtstruktur



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Generatorstruktur im Überblick



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung

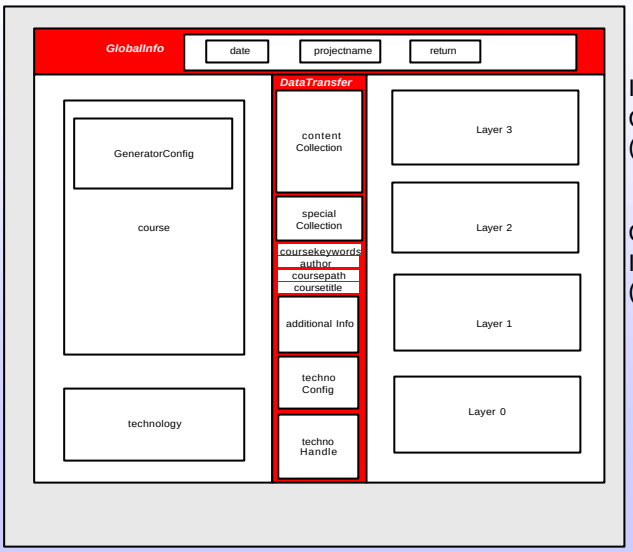
- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick

➢ Generatorarchitektur

- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Generatorarchitektur



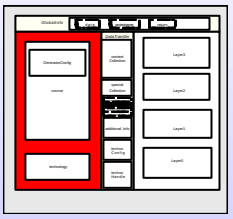
Interne Trennung der Generatorkomponenten (DataTransfer)

Global zugängliche Informationen (GlobalInfo)

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

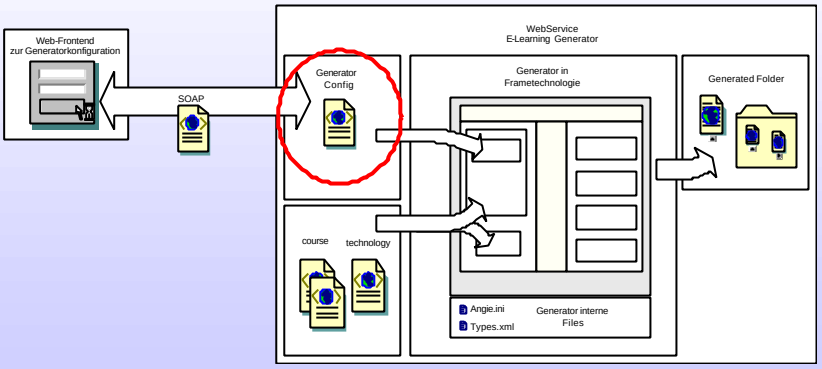
Übersicht Implementierung

- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- ➢ XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zwischenübersicht Gesamtstruktur



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

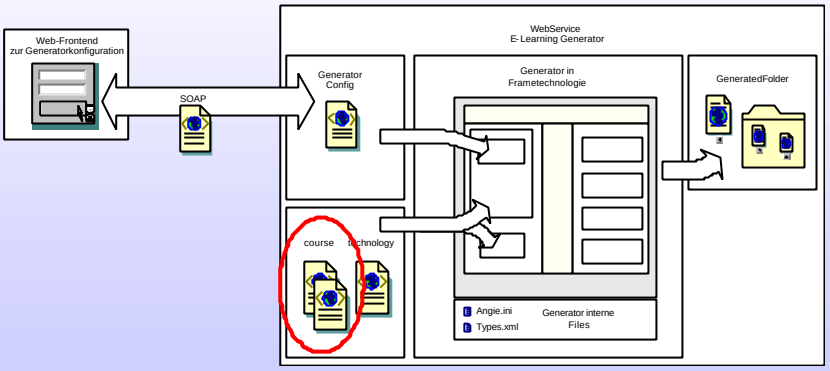
XML-Komponente

- Einlesen der Generator-konfiguration
  - technoconfig
  - contentconfig
  - additional

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<regulation>
  <config>
    <technoconfig>
      <feature gp="gn">toc</feature>
      <feature gp="ln">next</feature>
      <feature gp="gn">exit</feature>
    </technoconfig>
    <contentconfig>
      <extlink/>
    </contentconfig>
  </config>
  <additional>
    <course>C++</course>
    <orderer>Mattes</orderer>
  </additional>
</regulation>
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zwischenübersicht Gesamtstruktur



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

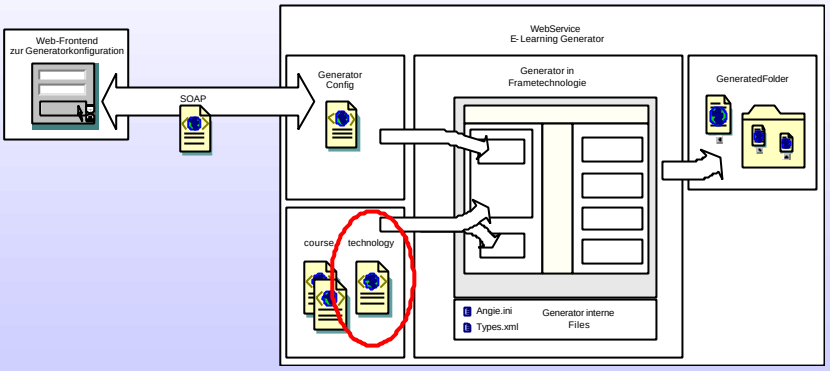
XML-Komponente

- Selektieren der Lerninhalte
  - LMML-Spezifikation (Uni-Passau)
  - Label
    - Kurs
    - Kapitel
    - Abschnitt

```
<lmml>
<section author="GPweb" label="kurs_cpp" title="C++">
  <section label="kapitel00" title="Kapitel 0" topics="">
    <section label="abschnitt00_01" title="Vorwort">
      <paragraph>
        <LMMLtext size="">Hier steht noch kein Vorwort</LMMLtext>
      </paragraph>
    </section>
    <section label="abschnitt00_03" title="Einleitung">
      <paragraph title="Zielsetzung">
        <LMMLtext size="">Der vorliegende Kurs führt in die
          Softwareentwicklung mit C und C++ ein. Wenn Sie ....
        </LMMLtext>
      </paragraph> ...
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Zwischenübersicht Gesamtstruktur



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

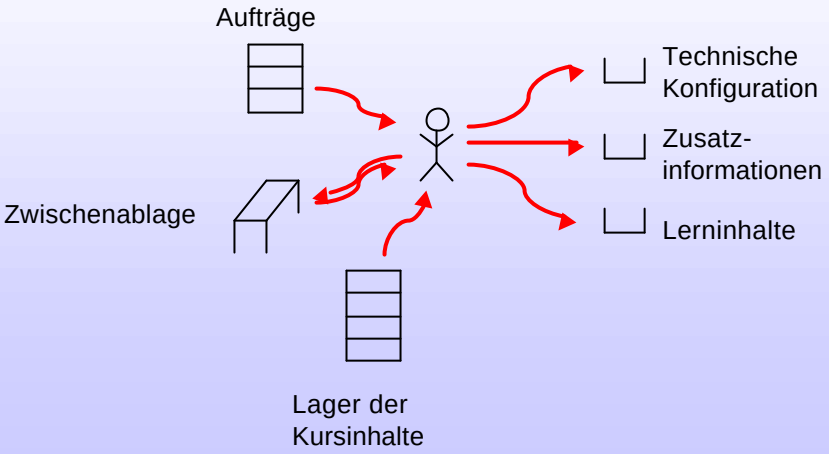
XML-Komponente

- Scriptbestellungen über XML-Komponente

```
<toc Host="script_code" MIME="text/javascript" Kommentar="<!-- /*TOC*/-->" >
<![CDATA[ function zeigeTOC(){
  var dateiname = "../course/content/toc.html";
  parent.intro.content.location.href=dateiname;} ]]>
</toc>
<toc Host="mediaCopy" >
<![CDATA[ inhaltsverzeichnis.gif ]]>
</toc>
<toc Host="integration_code" >
<![CDATA[ <img src="../layout/css1/images/inhaltsverzeichnis.gif" border="0"
alt="Inhaltsverzeichnis" onclick="zeigeTOC()"> ]]>
</toc>
```

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

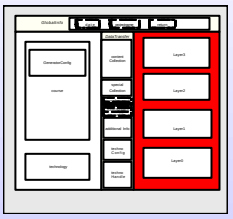
Fertigungsablauf in der XML-Komponente



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung

- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- ➢ Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- Web-Service



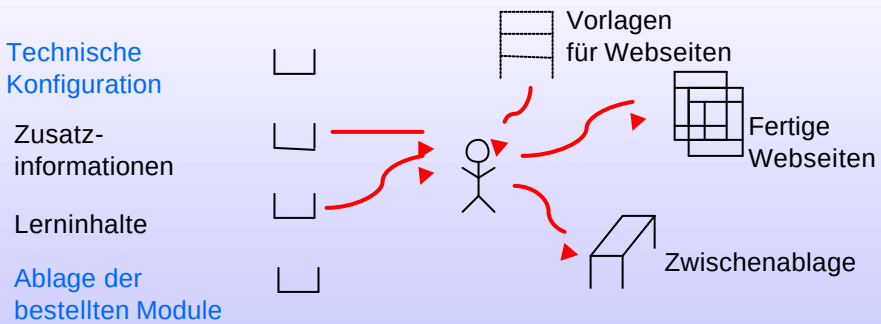
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Web-Komponente

- Schichten 0 bis 3
  - Schicht 0: Funktionen zur Slotinitialisierung
  - Schicht 1: Deklarationsebene
  - Schicht 2: Seitendefinitionen
  - Schicht 3: Script-Integration
- Vorgehensweise
  - Übernahme der Webinhalte
  - HMTL - Konvertierung ➞ Schicht 0
  - Montage der Webseiten ➞ Schicht 1
  - Seitendefinitionen ➞ Schicht 2
  - Seitenexport in Schicht 2 bzw. Ablage in Collection
  - Weitergabe in Schicht 3 zur Script-Integration
  - ➞ Export

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

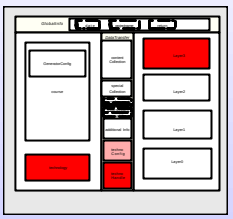
Fertigungsablauf in der Web-Komponente



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Übersicht Implementierung

- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- ➢ Script-Integration
- Web-Service



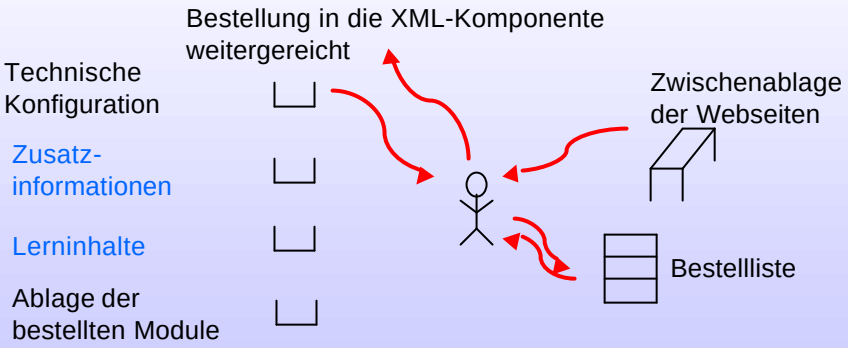
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Script-Integration

- Scriptvorbestellungen aus Schicht 2
  - Internes Script
- Ermittlung relevanter Einträge in der technischen Generatorkonfiguration
  - Externe Scripte
- Anfordern der Scripte
  - Sammeln benötigter Bestellinformationen
  - Bestellungen an die XML-Komponente weiterreichen
- Bearbeitung der scriptinternen Wildcards
- Script-Export, Script-Integration in die Webseiten

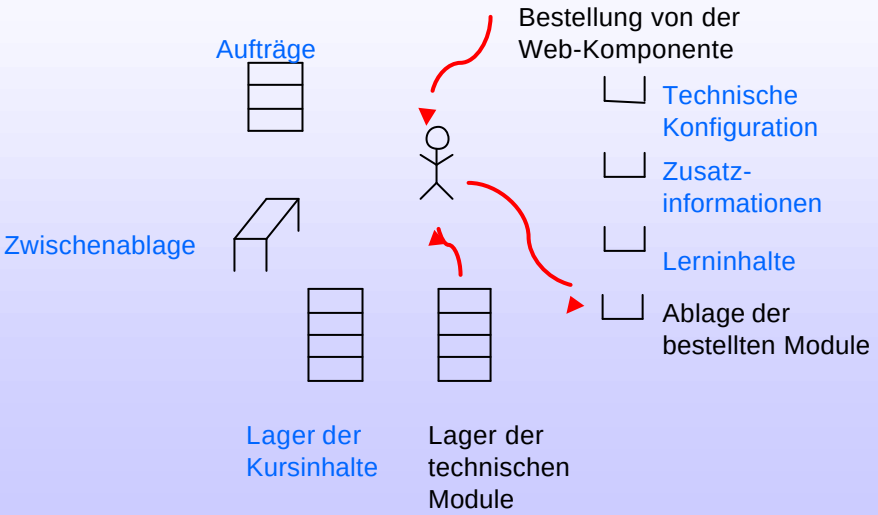
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Fertigungsablauf in der Web-Komponente



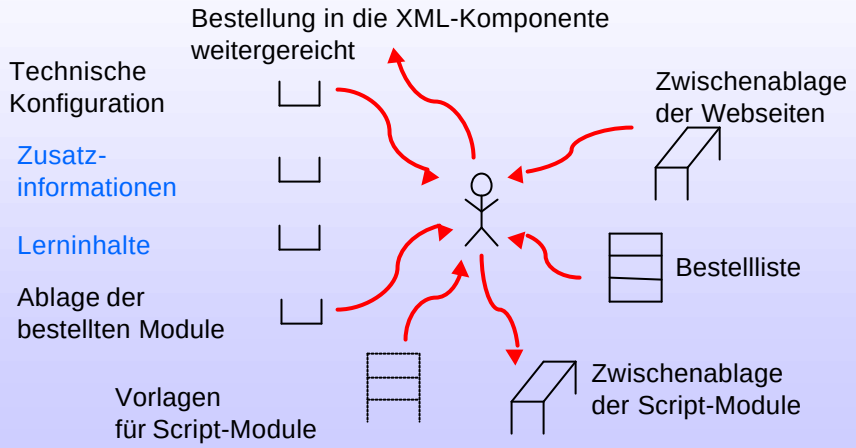
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Fertigungsablauf in der XML-Komponente



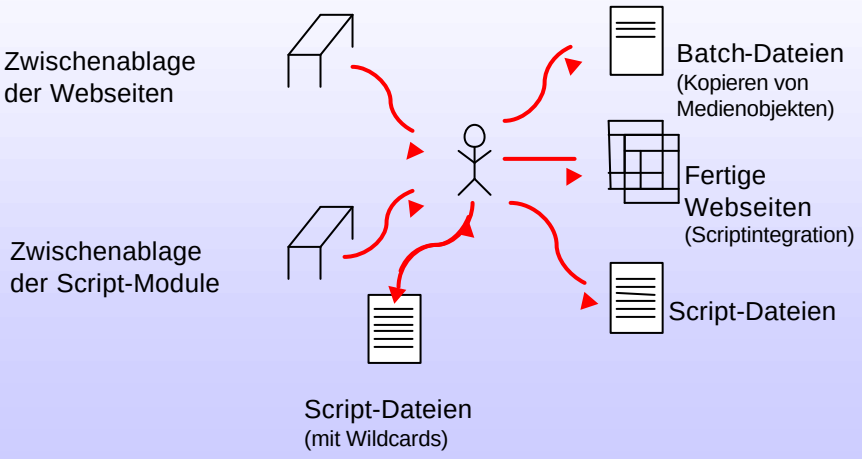
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Fertigungsablauf in der Web-Komponente



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Fertigungsablauf in der Web-Komponente



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wolny & Th. Bednasch

Übersicht Implementierung



- Generatoranforderungen
- E-Learning-System Anforderungen
- Gesamtstruktur
  - Generatorstruktur im Überblick
- Generatorarchitektur
- XML-Komponenten
- Web-Komponenten
  - Schichten 0 bis 3
- Script-Integration
- ➢ Web-Service

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Web-Service



- Implementiert mit VB.NET
- Zugang der Methodenaufrufe gesichert mit SOAP-Header
- Web-Service übernimmt die Steuerung und Überwachung des Generators
- Web-Frontend verbindet sich über einen Proxy mit dem Web-Service
- Web-Frontend zur Konfiguration des Generators und Web-Service mit Generator können auf verschiedenen Servern betrieben werden

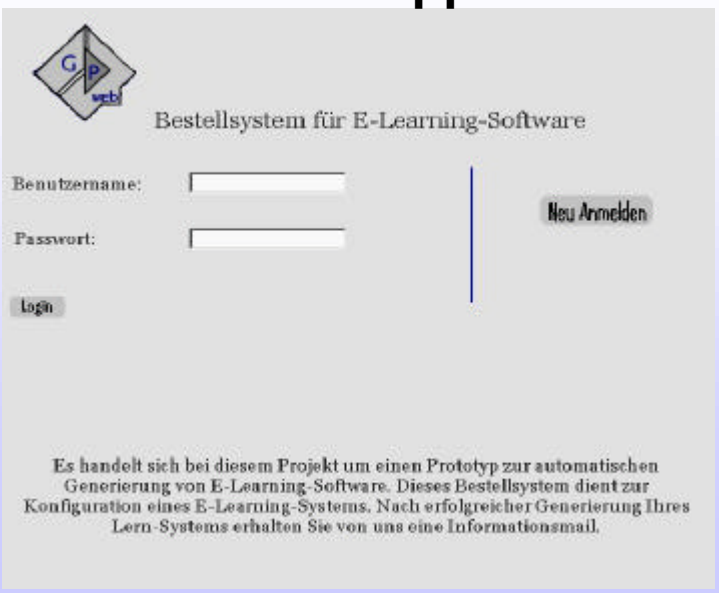
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Web-Service

- Versendet generiertes Passwort via Mail an den Endanwender
- Web-Service speichert die User-Daten in einer Datenbank
- Web-Service liefert Konfigurationstemplate und nimmt ausgeprägte Konfigurationsdaten entgegen
- Fertiges End-System wird via Mail an Endanwender versendet

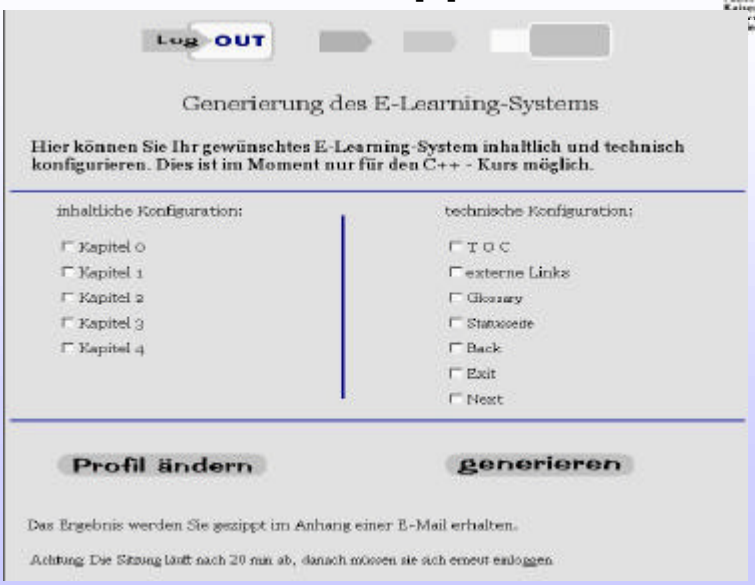
© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Ansicht der Webapplikation



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Ansicht der Webapplikation



© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch

Besten Dank ...

- ... für Ihre Aufmerksamkeit
- ... für Ihre Geduld
- ... für Ihre Fragen

Nicht vergessen, besuchen Sie diese Seiten

- *www.gp-web.org*
- *www.generative-programming.org*

© 2002 by U. W. Eisenecker, K. Czarnecki, M. Lang, M. Schlee, M. Henss, Th. Wollny & Th. Bednasch