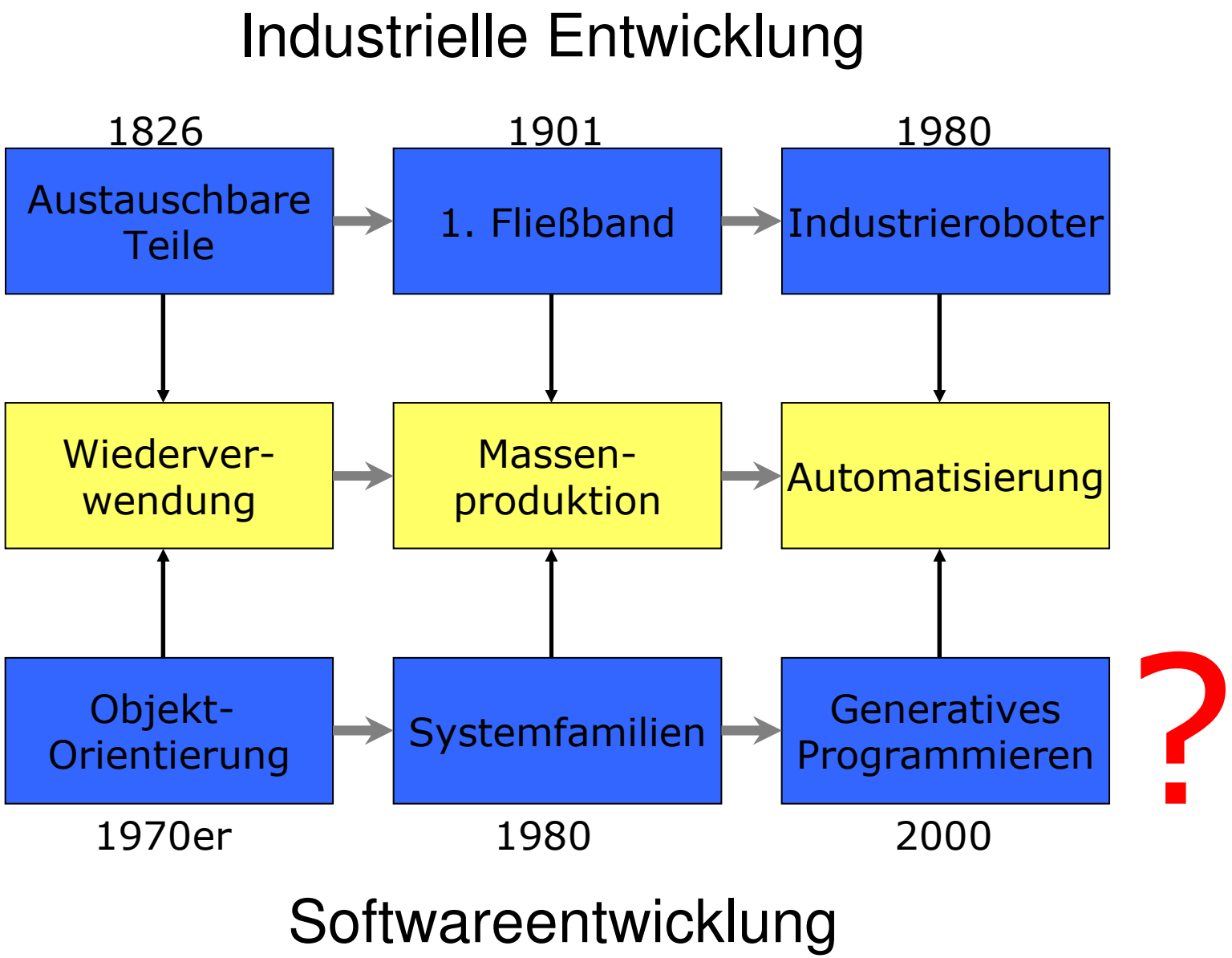
Einführung in Generatives Programmieren

Seminar
Moderne Softwareentwicklung
SS 2006

Bastian Molkenthin

Motivation


Industrielle Entwicklung
1826
1901
1980
Austauschbare Teile
1. Fließband
Industrieroboter
Wiederver-wendung
Massen-produktion
Automatisierung
Objekt-Orientierung
Systemfamilien
Generatives Programmieren
?
1970er
1980
2000
Softwareentwicklung

Rückblick auf Objektorientierung

Objektorientierte Softwareentwicklung soll in der Theorie Qualität und Produktivität erhöhen durch:

- Klassen
- Vererbung
- Polymorphie und dynamischen Binden
- Kapselung / Information Hiding
- Entwurfsmuster

Aber in der Realität …

Probleme

- Wiederverwendbarkeit
  - Klassen als Einheit zu klein
  - Frameworks schwer zu integrieren
- Komplexität
  - Änderungen und nachträglicher Einbau von Features implizieren Fragmentierung des Codes
- Verlust von Informationen
  - Wissen liegt nicht als Code vor
  - semantische Lücke zwischen Realität und Code

Ursache: Ausrichtung auf Einzelsystementwicklung

Hauptziel: bessere Wiederverwendbarkeit

Ziele des Generativen Programmierens

- Hohe Wiederverwendbarkeit
  - Automatische Generierung von Softwarekomponenten
  - Einfache Erstellung versch. Varianten eines Produkts
  - Kürzere Entwicklungszeit bei niedrigeren Kosten
- Optimale Anpassung an Einsatzkontext
  - Höhere Produktqualität
  - Optimierung im Hinblick auf Speicher und Geschwindigkeit

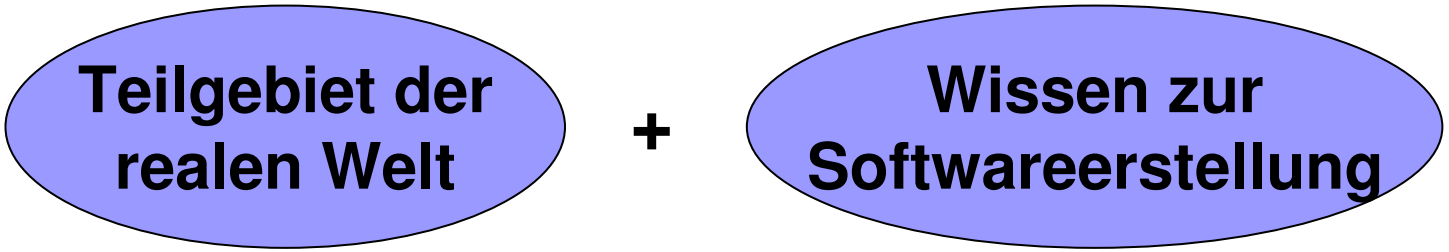
Definition Generatives Programmierens

„Generatives Programmieren (GP) modelliert Softwarefamilien so, dass ausgehend von einer Anforderungsspezifikation mittels Konfigurationswissen aus elementaren, wiederverwendbaren Implementierungskomponenten ein hochangepasstes und optimiertes Zwischen- oder Endprodukt nach Bedarf automatisch erzeugt werden kann.“

[Quelle: „Generative Programming“, Czarnecki/Eisenecker]

Begriffe

Domäne:



Umfasst das Wissen über Konzepte, Merkmalen und Abläufe der realen Welt in einem bestimmten Bereich, inklusive dem Wissen zur Erstellung von Software- systemen in diesem Bereich

Begriffe

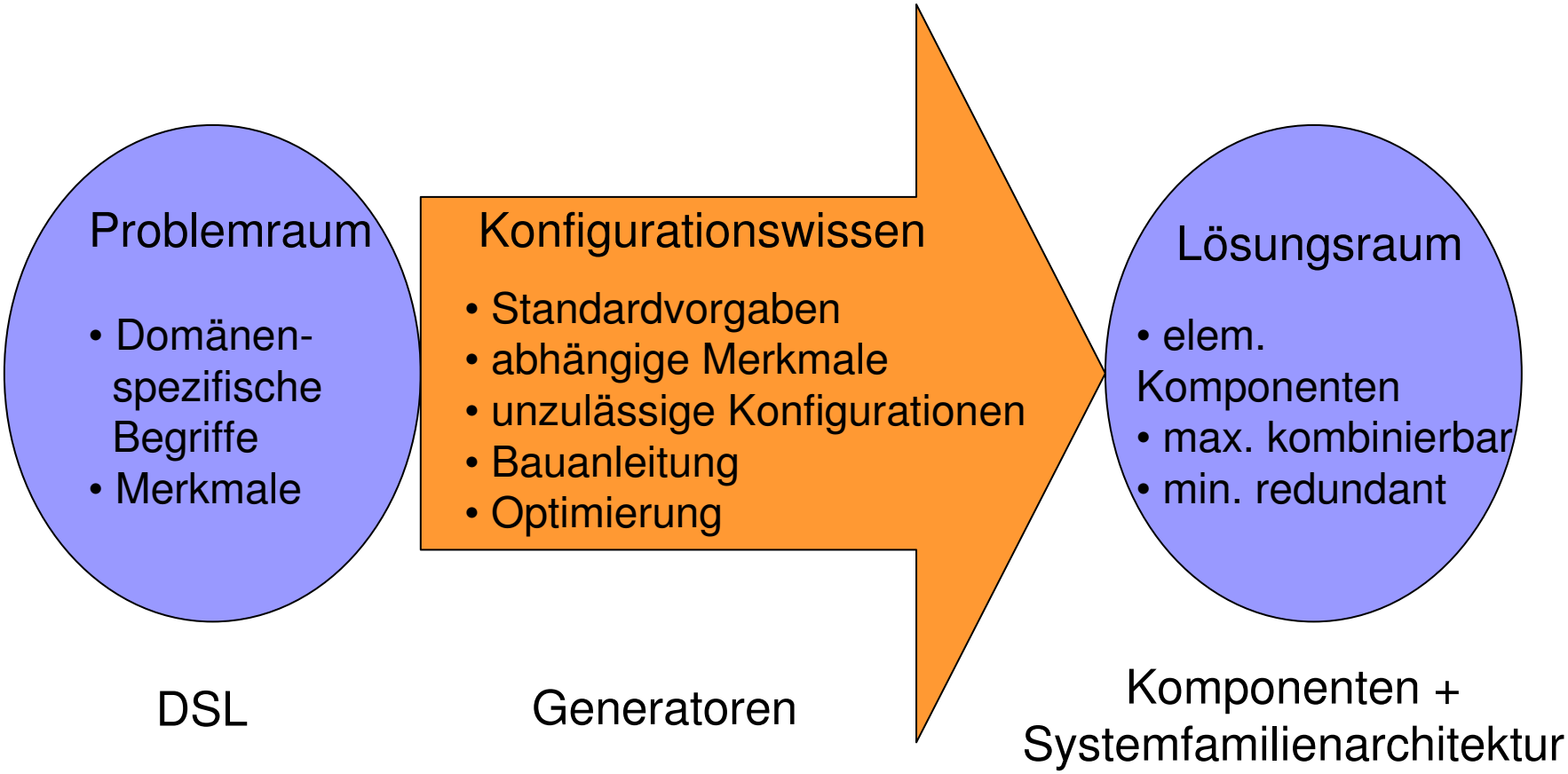
Domänenspezifische Sprache (DSL)
[Domain Specific Language]

- Problemorientierte Sprache einer bestimmten Domäne
- Spiegelt Begriffe und Denkweise der Experten wider
- Wird für die Spezifikation der zu erstellenden Software benutzt
- Einzige Beschreibungsmöglichkeit des Entwicklers
- Eingabe für den Generator
- Form wählbar: textuell, graphisch, …

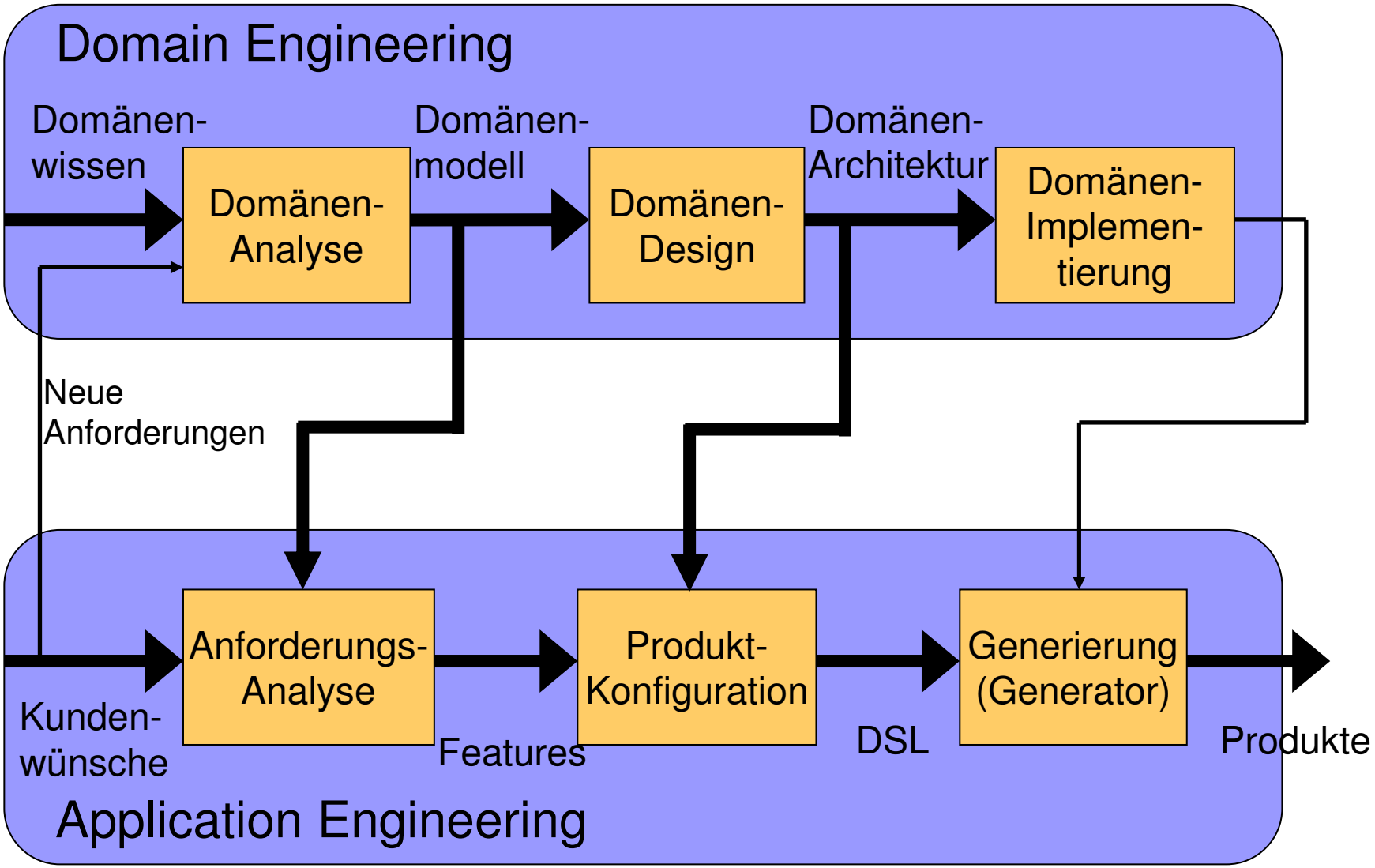
Begriffe

Generatives Domänenmodell

- Bildet Grundlage für Automatisierung der Softwareherstellung



Methoden des GP



Methoden des GP

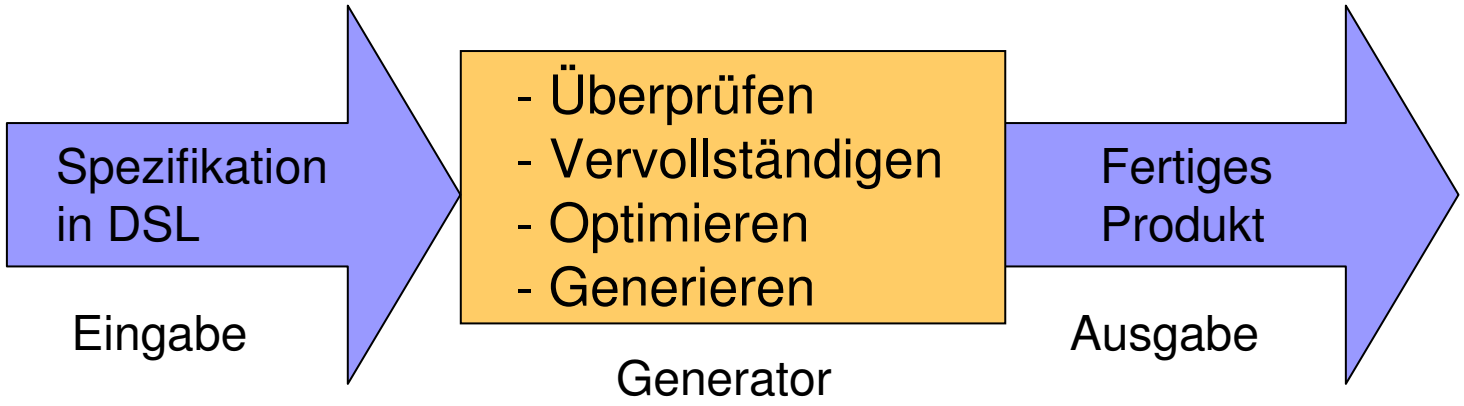
Domain Engineering…

- … ist das Sammeln, Organisieren und Speichern bisheriger Erfahrungen […] in bestimmten Domänen in Form von wiederverwendbaren Komponenten.
- Zeitaufwendige Vorarbeit des GP
- Ergebnis: wiederverwendbare Komponenten bzw. Generator
- „development for reuse“

Application Engineering…

- … ist der Prozess der Erstellung kompletter Systeme.
- Spezifikation in DSL + Generator = Softwareprodukt
- Mehrfache Wiederholung zur Erstellung verschiedener Varianten einer Software
- „development with reuse“

Generatoren



- Wunsch: Komplexe Programme, die auf Knopfdruck aus der Spezifikation das Softwareprodukt automatisch generieren
- Meist modularer Aufbau
- Erkennen fehlerhafte Eingaben
- Füllen Lücken in der Spezifikation mit Standardwerten
- Wählen selbstständig am besten passendste Komponenten

Beispiele für Generatoren

- Bekanntestes Beispiel: Compiler
- Entwicklungsumgebungen, die aus UML Diagrammen Javacode erzeugen
- „Generative Matrix Computation Library“ (GMCL)
  - Matrix-Konfigurations-DSL unterstützt 1840 unterschiedliche Arten von Matrizen
  - C++ Implementierung nur ca. 7500 Zeilen
- „Generative Matrix Factorization Library“ (GMFL)
  - generiert Algorithmen für LU-Zerlegungen
  - basiert auf GMCL
- Bordcomputer-Software für Satelliten
  - Spezifikation in XML
  - Erzeugung von Ada83-Code mit Hilfe einer JSP-ähnlichen Templatesprache

Programmiertechniken (für Lösungsraum)

Generisches Programmieren

- Abstrakte, generalisierte Beschreibung von Klassen, Funktionen, Datenstrukturen und Algorithmen zur Maximierung von Anpassungsfähigkeit und Kompatibilität
- Wichtige Konzepte: Typ-Parametrisierung & Polymorphie
- .NET Sprachen, C++ mit Templates, Java ab 1.5
- Generativ ≠ Generisch

Aspektorientiertes Programmieren (AOP)

- Verbessert Lokalität von Quellcode eines bestimmten Problems
- Aufteilung von Probleme in funktionale Komponenten als auch in Aspekte
- Kaum vorhanden in heutigen Programmiersprachen

Bewertung

Vorteile :

- Kürzere Entwicklungszeit ➔ geringere Kosten
- Spezifikation in DSL ermöglicht schnelle Änderungen
- Weniger Fehler im Produkt durch automatische Generierung
- Automatische Optimierung
- Modularer Aufbau des Generators ermöglicht teilweise Wiederverwendung in anderen Projekten
- Wissen liegt explizit vor

Bewertung

Nachteile :

- Entwicklung wiederverwendbarer Komponenten ist komplexer und zeitaufwändiger als Lösung eines speziellen Problems
- Herstellung und Wartung von Generatoren teuer
- Domain Engineering wichtig und aufwändig; stellt Fundament für DSL dar
- Abgrenzung der Domäne schwierig
- Domäne muss gut verstanden sein

Aufwand lohnt nicht für Einzelsystementwicklung

Fragen ?

Danke für ihre Aufmerksamkeit !