
Software-Krise 2.0?

Late Afternoon Talks 23./24. März 2010

„MDSD: effizienter, günstiger, besser“

Folie 1 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010

Was geschah im Jahr 1968?

NATO Software Engineering Conference
in Garmisch Partenkirchen



Friedrich Ludwig Bauer

„Software-Krise“

→ Schwierig, korrekten, verständlichen und überprüfbaren Code zu erstellen

- Schnelles Wachstum der Rechnerleistung
- Möglichkeit, komplexere Probleme zu lösen

Software-Krise 2.0?
Folie 2 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010

Herausforderungen in der Applikationsentwicklung von heute



Anwendungen



Technologie



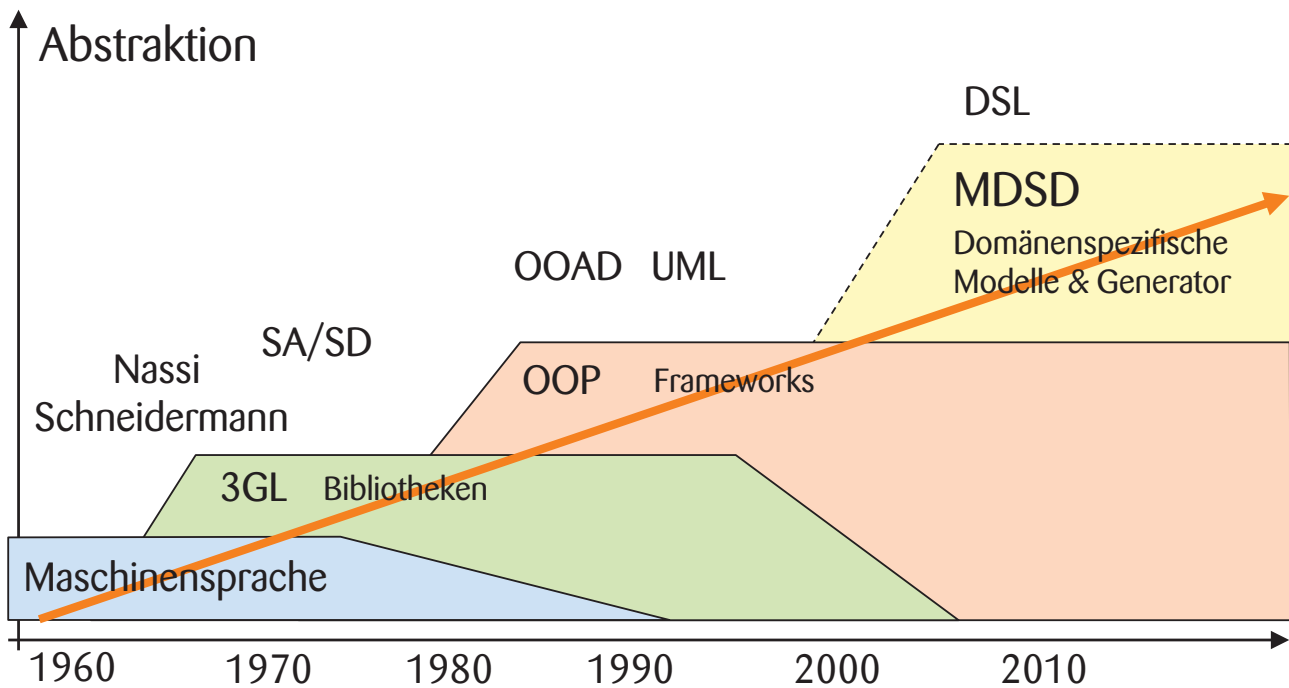
Spezialisten



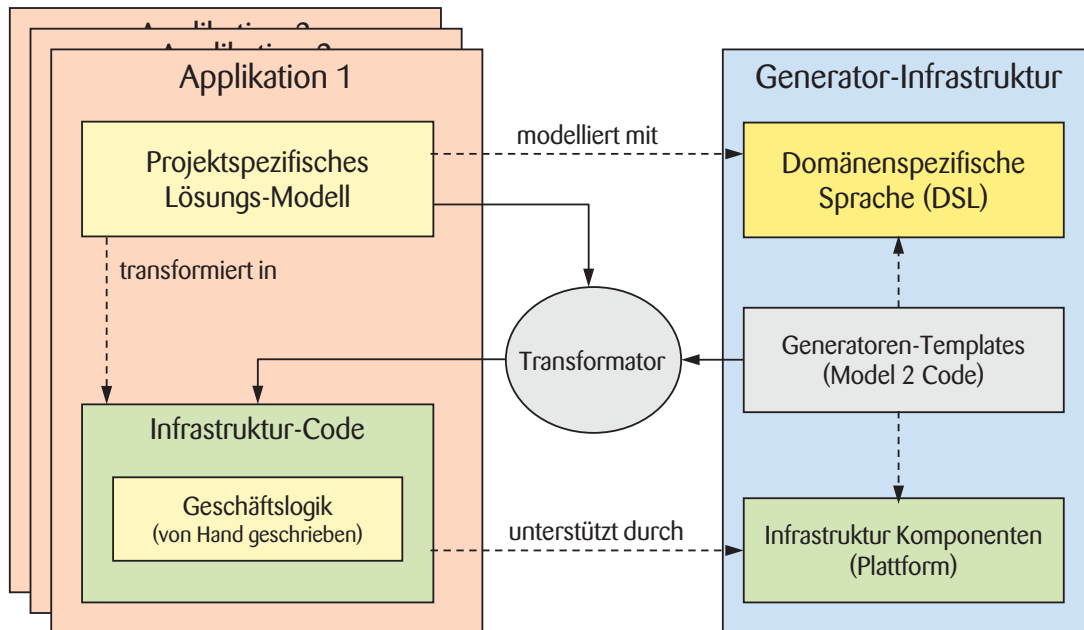
Software-Krise 2.0?
Folie 3 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010

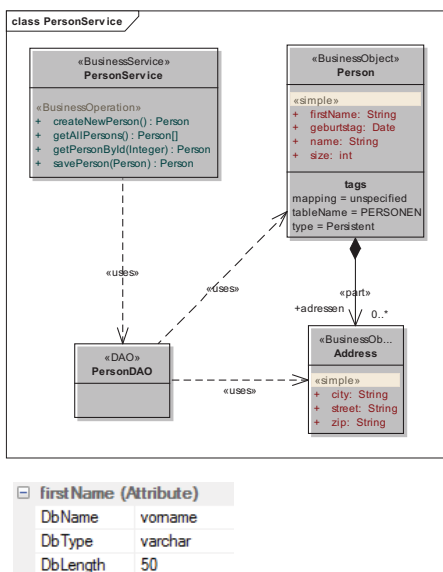
Umgang mit der Komplexität in der Geschichte



Modellgetriebene Software-Entwicklung (MDS) kurz vorgestellt



MDS an einem kleinen Beispiel



Generiert werden:

- Entity Objekte (Java)
- Datenbankstruktur (SQL)
- Mapping-Definition (XML)
- DAO mit Default-Verhalten für Person und Address (Java)
- Business-Service (Java)
 - Infrastruktur-Methoden
 - Verbindung zu DAO
 - Leere Methoden für Geschäftsfunktionen
- Zugriffsberechtigung (XML)
- Zudem (Java)
 - Factory-Methoden
 - Dependency Injection

MDSD steigert Effizienz, erhöht Qualität, reduziert Risiken/Kosten



Einfachere Modelle durch Abstraktion

- Leichter lernen, leichter anwenden
- Aktuelle Dokumentation

Technologie-Einsatz in Generator gekapselt

- Architekturvorgaben einhalten
- Architekturänderungen durch Generator

Höhere Qualität durch generierten Code

- Richtlinien einhalten
- Immer richtig / immer falsch

Weniger Spezialisten notwendig

- Generator Infrastruktur mehrfach einsetzen

Aufwand Realisierung einer mittelgrossen Web-Applikation mit J2EE Architektur



Aus „Model Driven Software Development“ T.Stahl, M. Völter

Einsatz von MDSD zu empfehlen bei:



Wiederholungen

- Applikationen mit gleicher Architektur (n-Tier, n-Layer)
- Ähnliche Elemente in der gleichen Applikation (Entities)

Redundanzen

- Business Layer → Presentation Layer (Transfer Objects)
- Business Layer → Persistenz (DB-Strukturen)

Anforderung mit sehr hoher Abstraktion

- RESTfull an Stelle von SOAP
- Synchron / Asynchron

Komplexe technische Strukturen

- Verwenden von Frameworks und Services (Spring, SOA)
- Technologie-Mappings (Hibernate, JPA, Middleware)

Diese Herausforderungen ergeben sich



Weitere Komplexität

- Generative Architektur, DSL und Generatoren

Spezialisten notwendig

- Architektur, DSL, Modelle, Generatoren

Generator Infrastruktur

- Referenzimplementation
- Modellprüfungen, Transformationen

Management Support

- Investition
- Mehrfachverwendung (ROI)

Software-Krise 2.0?
Folie 9 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010

Nutzen aus MDS



Effizienter

- Time to Market für neue Funktionen
- Schnelleres Feedback vom Fachbereich

Günstiger

- Geringere Kosten für Realisierung und Wartung
- Weniger Druck für Outsourcing/Offshoring

Besser

- Verifizierbare Modelle → geringere Fehlerkosten
- Bessere Kommunikation mit Fachbereich
- Spezialistenwissen wird automatisch angewendet

Software-Krise 2.0?
Folie 10 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010

Vielen Dank!



Software-Krise 2.0?
Folie 11 von 11
23./24. März 2010

Ulrich Brawand
© Zühlke 2010