

---

# Heute Neuentwicklung, morgen Legacy?

## Anwendungsfälle für Software Reengineering

Dr. Georg Molter

---



---

# Consulting Development Integration

---

---

Reengineering-Zielsetzung und –Ansätze

Besonderheiten von Reengineering-Projekten

Reengineering-Projektbeispiele

Lessons Learned und Fazit

---

## Beispiel 1

- Versicherungsbranche
- Quick-and-dirty Oracle Forms-Anwendung wird immer schlechter wartbar und erweiterbar, schlechte Usability
- Übernahme eines anderen Unternehmens ist Anlass für eine Neuentwicklung auf Grundlage der strategischen Plattform des Kunden
- Keine Spezifikation vorhanden, kaum Dokumentation

## Mich berührt das doch gar nicht!



### Beispiel 2

- Entwicklung einer VB6-basierten Software für die Versicherungsbranche durch ein Offshore-Unternehmen
- Insourcing wegen Qualitätsproblemen
- Herausforderungen:
  - Software instabil
  - Software unzureichend dokumentiert

Software Reengineering  
Slide 5  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Mich berührt das doch gar nicht!



### Beispiel 3

- Baumaschinenhersteller
- Mit der Entwicklung einer Steuersoftware für eine Maschinenserie beauftragter Zulieferer wird insolvent
- Zugriff auf den Sourcecode ist zwar vertraglich garantiert, die entsprechenden Schritte würden aber lange dauern und das Go Live der Gesamtlösung massiv verzögern

Software Reengineering  
Slide 6  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anlässe für Reengineering



### Erweiterung des Systems nicht oder nicht wirtschaftlich umsetzbar

- Funktionale Erweiterung
- Neue Schnittstellen
- Neue Zugangswege

### Systemeigenschaften

- Zu hohe Wartungskosten / Wartbarkeit nicht mehr gegeben
- Unzureichende Skalierbarkeit

### Wechsel der Hardwareplattform

- Aus Kostengründen
- Hardware nicht mehr verfügbar / wartbar

### Externe Anlässe

(Y2K-Umstellung, Euro-Umstellung, Basel II, BilMoG, ...)

Software Reengineering  
Slide 7  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Big Picture



**Software Reengineering ist die Analyse und strukturelle Veränderung eines Systems mit dem Ziel, es in neuer Form wiederherzustellen.**

Software Reengineering  
Slide 8  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

# Big Picture



Zusätzliche Informationen  
(Domänenwissen, Technologiewissen)

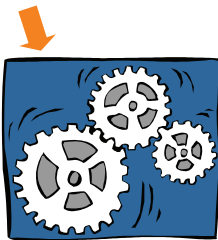
Ausgangssystem



Ergänzt Modell des  
Ausgangssystems



Reengineertes  
System



Analyse des  
Ausgangssystems

Software Reengineering  
Slide 9  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

# Erwartungen



## Verbesserungen bezüglich

- Wartbarkeit
- Erweiterbarkeit
- Wiederverwendbarkeit
- Portabilität
- Gebrauchstauglichkeit
- Skalierbarkeit
- Performance
- Offenheit
- ...

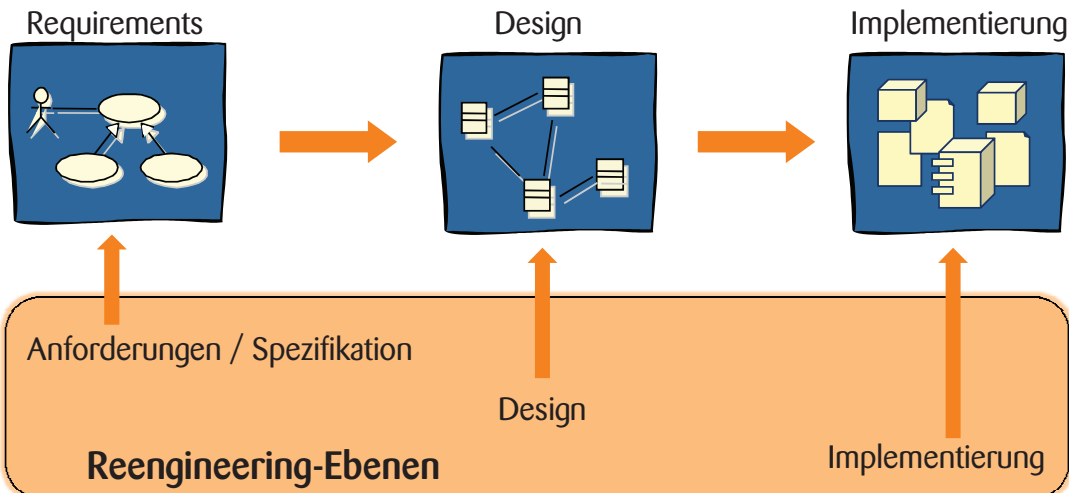
Software Reengineering  
Slide 10  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Ausgangspunkt: Traditionelle Softwareentwicklung



Umsetzung von Konzepten auf hoher Abstraktionsebene über logische, implementierungsunabhängige Designs zur physikalischen Systemimplementierung



Software Reengineering  
Slide 11  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

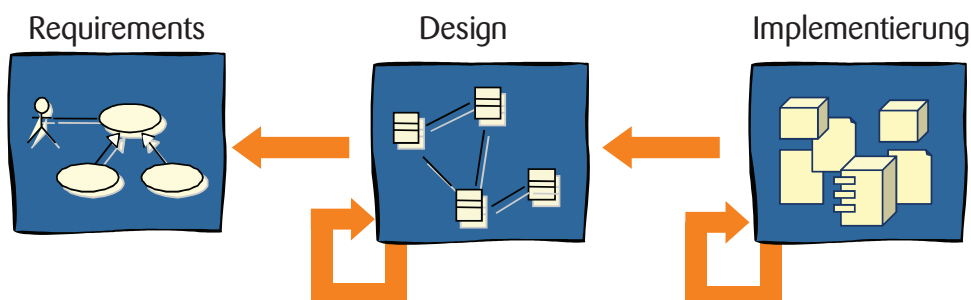
## Reverse Engineering



Wiedergewinnung verlorener Information über ein existierendes System

Präziser: Analyse eines Systems mit dem Ziel,

- die Systemkomponenten und ihre Beziehungen zu identifizieren;
- eine Beschreibung des Systems in einer anderen Form oder auf höherer Abstraktionsstufe zu gewinnen.



Software Reengineering  
Slide 12  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Redocumentation



### Analyse des Systems zur nachträglichen Erstellung von Dokumentation in unterschiedlicher Form

- Anwendungshandbücher
- Entwicklerdokumentation
- Reformatierter und kommentierter Sourcecode

#### Ziele:

- Verbesserte Benutzbarkeit
- Verbesserte Wartbarkeit und Wiederverwendbarkeit

#### Werkzeugunterstützung

- Formatierwerkzeuge (Pretty Printer)
- Dokumentationswerkzeuge

Software Reengineering  
Slide 13  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Respecification

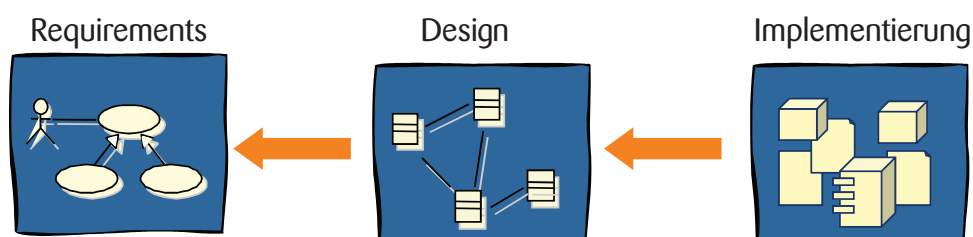


### Ableitung von Entwurfs- oder Anforderungsspezifikation auf Grundlage des Sourcecodes und der vorhandenen Dokumentation, sowie durch Analyse des Systemverhaltens

#### Kombination von strukturellem und funktionalem Wissen

#### Grundlage für

- Forward Engineering
- Qualitätssicherung



Software Reengineering  
Slide 14  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

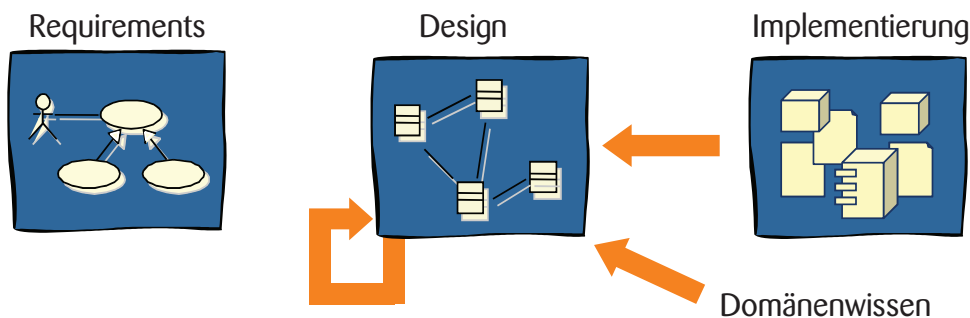
## Design Recovery



Kombination von über Reverse Engineering erhaltenen Informationen mit zusätzlichem Wissen (Domänenwissen, technologisches Wissen)...

... mit dem Ziel, eine umfassendere oder abstraktere Beschreibung des Systems zu erhalten

Beispiel: explizite Formulierung von im System implizit umgesetztem Business Rules



Software Reengineering  
Slide 15  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

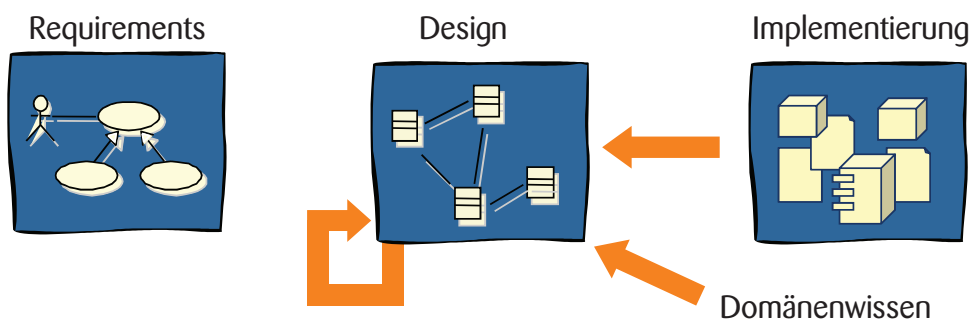
## Reverse Architecting, Architektur-Wiedergewinnung



Reverse Engineering mit dem Ziel, eine Beschreibung der Architektur des Systems zu erstellen.

- Komponenten und Konnektoren
- Architekturelle Sichten
- Architekturelle Mechanismen

Grundlage für tiefgreifende Änderungen oder Erweiterungen



Software Reengineering  
Slide 16  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Restructuring



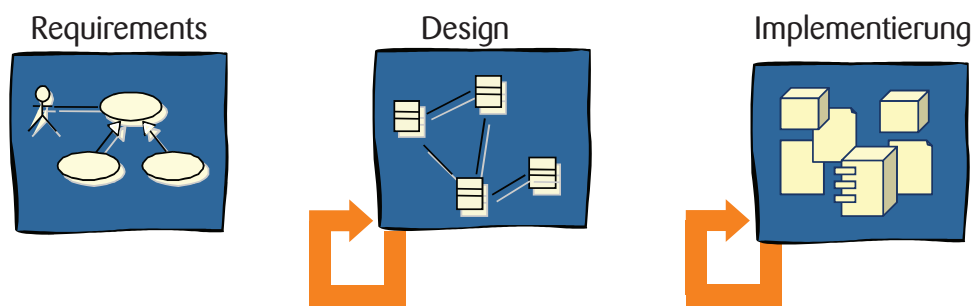
Transformation von einer Repräsentation in eine andere auf derselben Abstraktionsebene

← Code → Code

Design → Design

ohne Änderung von Semantik und nach außen sichtbarem Verhalten

aber mit unter Umständen massiven Veränderungen,  
z.B. Änderung der zugrundeliegenden Paradigmen (strukturierter Code, objektorientiertes Design, komponentenbasierte Architektur)



Software Reengineering  
Slide 17  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Restructuring - andere Interpretation



Übertragung eines Programms von einer Darstellung in eine andere, ohne Änderung am Grad der Abstraktion oder am funktionalen Verhalten

### Beispiele:

- Entfernung von GOTO's
- Formatierung
- Entfernung von totem Code

Software Reengineering  
Slide 18  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

Umschließen des unveränderten alten Systems oder eines Teils davon mit einer neuen Schnittstelle

Beispiel: Web-Schnittstelle für hostbasierte Anwendung

- Problematik des Altsystems bleibt bestehen (Wartbarkeit, Skalierbarkeit, ...)
- Parallelbetrieb beider Systemumgebungen
- + Voraussetzung für schrittweise Ablösung des Altsystems
- + Organisatorisch vorteilhaft

Transformation, Hosting oder Portierung eines vorhandenen Systems für seinen Einsatz in einer neuen Umgebung

- Neues Betriebssystem
- Neue Hardware-Plattform
- Neue Entwicklungsumgebung

### Techniken

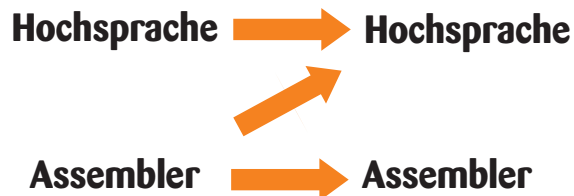
- Einsatz von Retargeting-Werkzeugen
- Virtualisierung / Emulation
- Kompatibilitäts-Bibliotheken

? Retargeting oder Neuentwicklung?

# Programmtransformation



Sourcecode-Transformation von einer Sprache in eine andere, oder zu einer anderen Version der ursprünglichen Sprache



Software Reengineering  
Slide 21  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

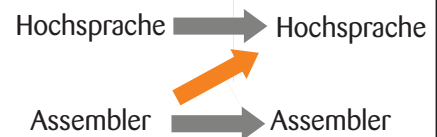
# Programmtransformation



## Assembler nach Hochsprache

Schwierigkeiten:

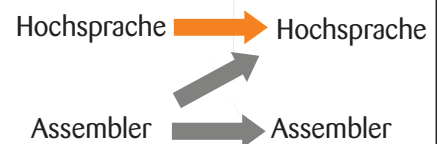
- Erkennen von Hardware-Spezifika und hochsprachlichen Konstrukten
- Qualität des Zielcodes



## Hochsprache nach Hochsprache

Schwierigkeiten:

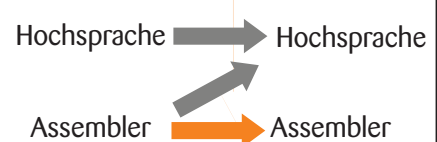
- Kontrollfluss, Datenstrukturen und Sprachmechanismen der Quellsprache bleiben größtenteils erhalten



## Assembler nach Assembler

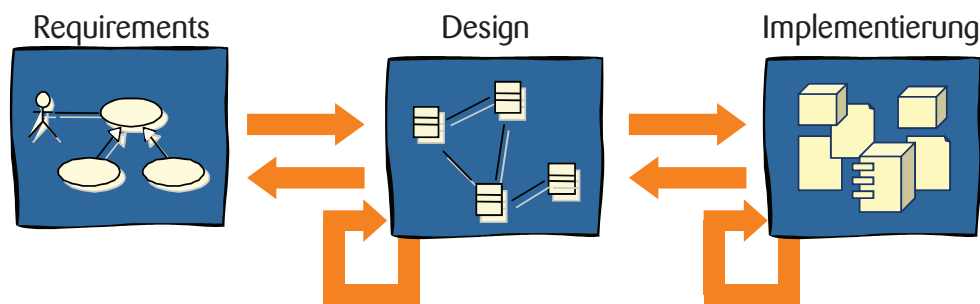
Voraussetzungen:

- Ähnliche Prozessorstruktur bezüglich Speicherverwaltung, Registerstruktur, I/O-Implementierung



Realisierung eines neuen Systems unter Rückgriff auf vom Altsystem abgeleitete Produkte und Artefakte

Neurealisierung fehlender oder nicht verwertbarer Teile des Altsystems



Software Reengineering  
Slide 23  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

Realisierung eines neuen Systems unter Rückgriff auf vom Altsystem abgeleitete Produkte und Artefakte

Neurealisierung fehlender oder nicht verwertbarer Teile des Altsystems

### Randbedingungen:

- ✓ Problem weitgehend bekannt
- ✓ Wissen über das vorhandene System erleichtert die Projektdurchführung (Randbedingungen, Aufwandsschätzung)
- ! Gefahr versteckter Abhängigkeiten / Seiteneffekte
- ! Lösung durch vorhandenes System eingeschränkt

Software Reengineering  
Slide 24  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

# Durchführung von Reengineering-Projekten

Software Reengineering  
Slide 25  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Arten von Reengineering-Projekten

### Reengineering eines einzelnen Systems

### Reengineering einer Produktlinie

- Identifikation des ursprünglichen Domänenmodells und der daran vorgenommenen Erweiterungen
- Problematik: Integration von Seitenästen (abgeleitete Individualsysteme)

### Ausbau eines Produktes zur Produktlinie

- Nicht nur Respezifikation des Systems, sondern zusätzlich Erarbeiten eines Domänenmodells und Betrachtung der Systemeigenschaften vor diesem Hintergrund

Software Reengineering  
Slide 26  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Reengineering-Vorgehensweisen: Komplett-Umstellung



### Bereitstellung der vollen Systemfunktionalität in einem Release

#### Erforderliche Ressourcen

- Wartung des Altsystems bis zum Übergang
- Ressourcen für die Neusystementwicklung

#### Auswirkungen für die Anwender

- Hoher Supportbedarf bei der Umstellung
- Keine parallele Benutzung von zwei Systemen

Software Reengineering  
Slide 27  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Reengineering-Vorgehensweisen: Sukzessive Migration



### Parallelbetrieb von Alt- und Neusystem, sukzessive Verlagerung von Funktionalität vom Alt- ins Neusystem

#### Technische Voraussetzungen

- Unterstützung für zentrale Datenhaltung oder Synchronisation doppelter Datenhaltung

#### Erforderliche Ressourcen

- Schritthaltende Verlagerung des Wartungsaufwandes vom Alt- ins Neusystem
- Ressourcen für die Neusystementwicklung

#### Auswirkungen für die Anwender

- Parallele Benutzung zweier Systemumgebungen: erhöhter Aufwand, ineffizientere Arbeitsabläufe
- Schrittweise Einführung in die neue Umgebung

Software Reengineering  
Slide 28  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Reengineering-Vorgehensweisen: Sukzessive Umstellung auf Subsystem- /Modulebene



- Sukzessives Bereitstellen, Integrieren und Testen neuer Subsysteme
- Architektur als Skelett für die Integration
- Wrapping von Teilen des Altsystems
- Voraussetzung: Altsystem muss sinnvoll partitionierbar sein
- Architekturelle Mechanismen zur Unterstützung des Mischbetriebs erforderlich

### Erforderliche Ressourcen

- Beginn der Arbeiten am Neusystem mit einem Kernteam
- Schritthaltende Verlagerung von Wartungs- und Entwicklungsaufwand aufwandes vom Alt- ins Neusystem

### Auswirkungen für die Anwender: Ähnlich wie bei Komplett- Umstellung

Software Reengineering  
Slide 29  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Projektdurchführung



### Ausgangspunkt: Bestandsaufnahme des Altsystems

- Identifikation von Subsystemen
- Bewertung von Wartbarkeit, Wartungssituation, Qualität etc. auf Subsystemebene

### Scoping und Priorisierung des Reengineering-Projekts

### Entscheidung über die Abstraktionsebene der Reengineering-Aktivitäten:

Code - Design / Architektur - Spezifikation

### Planung von

- Reverse Engineering (Respecification, Redocumentation, Architecture Recovery etc.)
- Forward Engineering - Rekonstruktion der Systemfunktionalität in der Zielumgebung

Software Reengineering  
Slide 30  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Technische Planung der Migration: Roadmap für die Ablösung von Altsystemkomponenten

- Datenbankmigration
- Migration externer Schnittstellen
- Migration der IT-Infrastruktur

## Organisatorische Aspekte

- Anwender
- Entwicklungsabteilung
- Systemadministration und Benutzersupport

## Benutzerschulung und -support

## Reine Reengineering-Projekte rechtfertigen sich nur durch Verbesserung in nichtfunktionalen Eigenschaften wie

- Wartbarkeit
- Wiederverwendbarkeit
- Erweiterbarkeit
- Portabilität
- Verfügbarkeit / Stabilität

Abschätzen der bezüglich dieser Qualitätsmerkmale erreichbaren Verbesserungen bei Projektbeginn

Etablieren geeigneter Metriken zur Erfolgskontrolle

Kombinierte Betrachtung bei Bereitstellung zusätzlicher Funktionalität

## Testen in Reengineering-Projekten



**Qualitätssicherung abhängig von Vollständigkeit und Qualität der Anforderungsspezifikation**

**Im Vergleich zu Neuentwicklungen überproportional hoher Aufwand für das Erstellen von Testfällen**

**Direkter Vergleich mit dem Altsystem technisch aufwendig und nicht aussagekräftig**

- Verhalten des Altsystems muss nicht korrekt sein!
- Problematik: Bestimmung der Testabdeckung

Software Reengineering  
Slide 33  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Risiken für Reengineering-Projekte



**Performance der Altsysteme zunächst nur schwer zu erreichen**

**Unzureichende Qualitätssicherung**

**Ungenügende Qualität des Zielsystems**

- Alte und neue Fehler kombiniert
- Transformation nicht wartbaren Codes in nicht wartbaren Code

**Vernachlässigen von Informationsquellen**

- Code
  - „Wir kennen unsere Domäne.“
  - Aber oft: Quellcode nicht verfügbar / nicht verständlich
- Mitarbeiter

Software Reengineering  
Slide 34  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

... Mehr Risiken...



---

**Blindes Vertrauen auf Tools**

**Hohe organisatorische Komplexität**

**1:1-Abbildung des Altsystems mit seinen Schwächen und Fehlern  
(Nichtergreifen einer Chance)**

**Kosten/Nutzen-Verhältnis des Reengineering-Projekts nicht bekannt**

**Keine Änderung der vom System unterstützten Prozesse**

**Vernachlässigen der Bedeutung des Faktors Mensch**

---

Software Reengineering  
Slide 35  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

---

**Reengineering-  
Anwendungsfälle**



Software Reengineering  
Slide 36  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anwendungsfall: Reengineering einer Produktlinie



### Ausgangssituation:

- Familie von ähnlichen, aber individuell angepassten Systemen
- Keine durchgängige gemeinsame Codebasis als Kern

### Reengineering-Zielsetzung

- Bessere Schätz- und Planbarkeit der Ableitung neuer Systeme
- Nutzung von Cross-Leverage-Effekten
- Geringerer Aufwand und kürzere Time-to-market für die Entwicklung neuer Systeme
- Verbesserte Wartbarkeit

Software Reengineering  
Slide 37  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anwendungsfall: Reengineering einer Produktlinie



### Vorgehensweise

- Aufarbeiten und Konsolidierung der Variantenvielfalt
- Reverse Engineering des zugrundeliegenden Domänenmodells (Abstraktionen, Gemeinsamkeiten und Variationspunkte in der Domäne)

### Alternativen

- Reengineering der Implementierung zu Core Assets, Kapselung der Spezifika einzelner Systeme
- Neuentwicklung von Core Assets auf Grundlage des reengineerten Domänenmodells

Software Reengineering  
Slide 38  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anwendungsfall: Reengineering einer Produktlinie



### Erfahrungen

- Ursprünglich favorisierter teilautomatisierter Reengineering-Ansatz schnell als unzulänglich erkannt
  - Teilautomatisiert gewonnener Code nicht als Grundlage für etliche Produktvarianten einsetzbar
- Reverse Engineering des Domänenmodells führte zu zahlreichen Änderungen / Erweiterungen / Vereinfachungen

Software Reengineering  
Slide 39  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anwendungsfall: Migration Host → Client/Server-Umgebung



### Ausgangssituation

- EDV-Landschaft mit umfassender Unterstützung der Geschäftsprozesse
- Hohe fachliche Komplexität
- Enge Vernetzung der Anwendungen

### Reengineering-Zielsetzung

- Ablösung der Host-Plattform
- Unterstützung für unterschiedliche Zugangswege
- Bereitstellung zusätzlicher Schnittstellen

Software Reengineering  
Slide 40  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Anwendungsfall: Migration Host → Client/Server-Umgebung



### Vorgehensweise

- Realisierung einer geeigneten Infrastrukturplattform
  - technische Infrastruktur für die Realisierung des Client/Server-Systems
  - Unterstützung für Wiederverwendung auf unterschiedlichen Abstraktionsebenen
- Komplettes Reverse Engineering vorab nicht möglich
  - ➔ sukzessive Migration der einzelnen Anwendungen

### Erfahrungen

- Migrationsprojekt von der IT getrieben;  
Nutzenpotenzial seitens der Fachabteilung zu spät erkannt, um den möglichen Nutzen komplett abzuholen

Software Reengineering  
Slide 41  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009



## Lessons Learned und Fazit

Software Reengineering  
Slide 42  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

### Explizite Berücksichtigung des Reengineering-Projektcharakters

- bei der Definition der Vorgehensweise,
- bei der Projektplanung,
- bei Schätzungen etc.

**verbessert Vorhersagbarkeit und Effizienz der Projektdurchführung.**

### Reverse Engineering des Domänenmodells ist extrem wertvoll.

- Grundlage für Forward Engineering, Klärung von Use Cases und externen Schnittstellen
- Grundlage für die Migration bzw. Integration mit dem Altsystem

**In den meisten Fällen wird – und sollte – mehr geschehen als Nachbildung mit identischer Funktionalität.**

- Change Requests gibt es auch bei Reengineering-Projekten...
- Pay-Off des Reengineering-Projektes entsteht oft nur durch begleitende Optimierung von Prozessen.

www.zuehlke.com

gmo@zuehlke.com



## Quellen



- Taxonomy Project of the IEEE-CS Technical Council on Software Engineering (TCSE) - Committee on Reverse Engineering
- Chikofsky, E.J., Cross II, J. H., 'Reverse Engineering and Design Recovery: A Taxonomy', IEEE Software, January (1990).
- DOD policy workshop SB-1, Santa Barbara, California, 1992
- WCRE Conference Series (Working Conference on Reverse Engineering)

Software Reengineering  
Slide 46  
23. Juni 2009

Dr. Georg Molter, Zühlke  
© Zühlke 2009

## Browsing und Strukturierung

- Hypertext
- Struktur-Visualisierung

## Formatierung, Pretty-Printing

## Generierung von Dokumentation,

## Code- und Designmetriken

## Werkzeuge und Toolkits für

- Sprachwechsel
- Datenbankwechsel
- Wechsel der Benutzerschnittstelle

# Ansätze in Abhängigkeit von der Reengineering-Ebene

## Code-Ebene

- Program Slicing
- Code-Transformation
- ...

## Design-Ebene

- Remodularisierung
- Subsystemerkennung
- Architekturerkennung
- ...

## Spezifikationsebene

- Business Rules
- ...