

55. Werkzeuge für Wartung und Reengineering

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
<http://st.inf.tu-dresden.de>
Version 11-0.1, 29.12.11

- 1) Aufgaben
- 2) Vorgehen des Reengineering
- 3) Beispiele für Werkzeuge



55.1 Aufgaben von Wartung und Reengineering



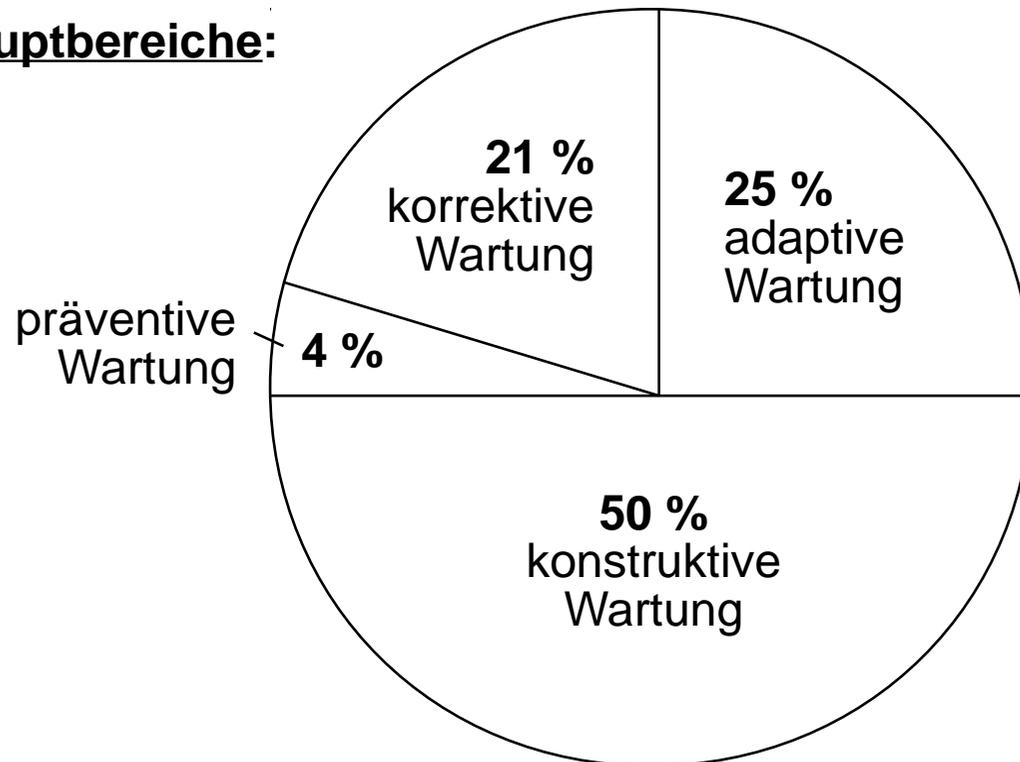
Hauptbereiche der Wartung

Def nition nach ANSI/IEEE Std. 610.12-1990:

Software-Wartung ist die Modifikation eines Software-Produktes oder einer Komponente nach der Auslieferung mit dem Zweck der:

- Fehlerlokalisierung und -korrektur,
- Verbesserung der Performance oder anderer Systemattribute,
- Adaptierung an eine geänderte Umgebung

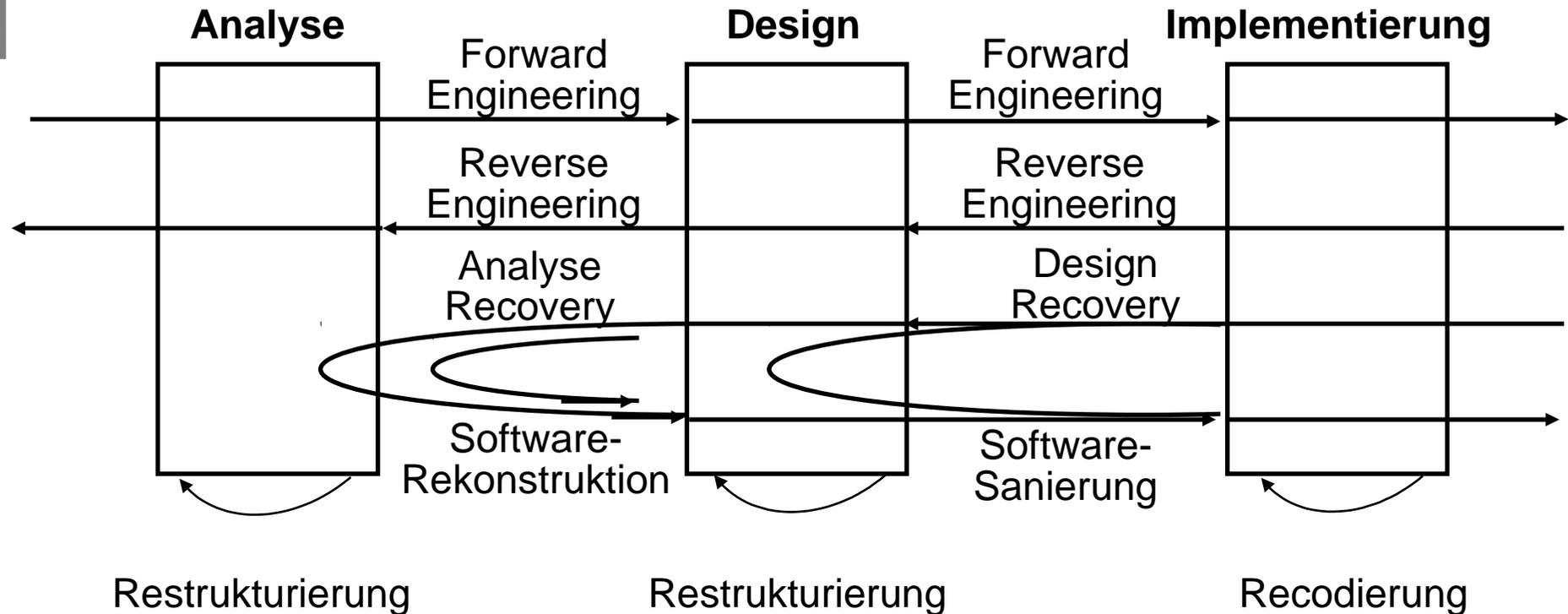
Hauptbereiche:



Zusätzlich unterscheidet man die **operative Wartung**.

Die **Wartungskosten** eines durchschnittlichen Anwendungsunternehmens liegen zwischen **50 - 70 %** des gesamten DV-Etats. [3, S.664].

Kategorien des Reengineering



Software Reengineering: Reverse Engineering oder *Design/Analyse Recovery*
in Verbindung mit *Restrukturierung/Redokumentation*
+ Forward Engineering

Quelle: nach Arnold, R.S. (Hrsg.): Software Reengineering; IEEE Computer Society Press 1994

Terminologie des Reengineering

- **Forward Engineering:** Anwenden von Methoden und Werkzeugen der Software-Entwicklung, um **Software-Dokumente** in eine implementierungsnähere Form zu bringen.
- **Reverse Engineering:** Identifizierung der einzelnen Komponenten eines **Software-Dokuments** (Quelltext, Klassen-, Modulstruktur, ERD,...) und ihrer Beziehungen durch eine methodische Analyse auf abstrakterem Niveau.
- **Design Recovery:** Wiedergewinnung einer vollständigen, abstrakten **Entwurfs-Beschreibung** eines vorliegenden **Software-Dokuments** unter Einbeziehung von **Domänenwissen** (relevante Informationsquellen, Anwenderwissen).
- **Analyse Recovery:** wie oben, aber Wiedergewinnung einer vollständigen, abstrakten **Analyse-Spezifikation**.
- **Restrukturierung:** Veränderungen/Umstrukturierungen von **Software-Dokumenten** (z.B. Modulchart, Spezifikation), um eine Vereinheitlichung oder Verbesserung der Programm- und Datenstruktur zu erhalten.
- **Recodierung:** **Restrukturierung** im Software-Dokument **Quelltext** (z.B. Entfernen von direkten Sprungbefehlen, unnötigem Code und Datenzugriffen usw.).
- **Redokumentation:** **Wiedergewinnung** oder Erzeugung von semantisch äquivalenten **Repräsentationen** innerhalb desselben Abstraktionsniveaus.

Quelle: nach Baumöl, U. u.a.: Einordnung und Terminologie des Software Reengineering; Informatik-Spektrum 19(1996) H.4 S. 191 - 195

Problemgruppen für Reengineering

- **Systemstrukturprobleme** äußern sich z. B. in hoher Komplexität der Komponenten, in hart codierter Logik, versteckter Semantik und nicht transparenter Mehrfachverwendung
- **Datenstrukturprobleme** treten bei unverträglichen Datentypen, Datenformaten (Jahreswechselprobleme), Parameterübergaben,... auf
- **Oberflächenprobleme** bei Übergang zu anderen Bildschirmen, Toolkits bzw. zu anderen ereignisgesteuerten GUIs
- **Plattformprobleme** bei unverträglichen Hardware- bzw. OS-Wechseln

Problem- gruppe \ Renovierungs- typ	Design Recovery (Redesign)	Restrukturierung	Portierung
Systemstruktur	++	+	-
Datenstruktur	+	++	-
Oberfläche	+	+	+
Plattform	+	++	++

Quelle: Keipinger, D.: Software-Renovierung; in Brössler, P., Siedersleben, J.(Hrsg.): Softwaretechnik; Hanser Verlag 2000

Ziele des Reengineering

- **Steigerung der Produktivität durch**
 - Einführung erprobter Technologien in bestehende Software
 - Übergang auf leistungsfähigere Programmiersprachen (Optimierung)
 - Verbesserung der Integrierbarkeit
 - leichtere Wartung und Motivation des Wartungspersonals
 - Verbesserung der Systemverwaltung
- **Verbesserung der Portabilität durch**
 - Plattformabtrennung: Trennung plattformunabhängiger, systemspezifischer, DB-spezifischer- und Anwendungs-Komponenten (transparente Dienststrukturen)
 - Einhaltung von Standards (Benutzungsoberflächen, API, SAA-Schnittstellen)
- **Erhöhung der Wiederverwendbarkeit durch**
 - Abbau der Personengebundenheit
 - Ermöglichung von Migration und Systemevolution
 - Erhalten und Verlängern der System-Lebensdauer
 - Niveaueanhebung und Wartung mit CASE-Werkzeugen auch zur Angleichung bzw. Kopplung mit bestehenden Software-Systemen

Quelle: nach McClure, C.: Software-Automatisierung - Reengineering - Repository - Wiederverwendbarkeit;
Carl Hanser Verlag 1993 S. 26 ff

55.2 Vorgehen des Reengineering



Schichtenmodell des Reengineering

Sanierung auf **Ebene des Anwendungs-Codes:**

- Herstellung von Strich- bzw. Einrückdiagrammen
- Umsetzung in Zwischen-/Pseudocode (bzw. Struktogramme für Steuerf.)
- Elimination redundanter Codeteile bzw. wilder Sprünge (GOTO)
- Extraktion von Automaten oder Statecharts
 - ⇒ *Auff nden anwendungsorientierter Code-Teile*

Sanierung auf **Ebene der Programmsteuerung:**

- Trennung Def nitionen von Anweisungen
- Auff nden Programmrahmen, Abspaltung von (ext.) Dienstrouinen
- Festlegen Aufrufhierarchie
 - ⇒ *Bestimmung Funktionen(Aktionen) mit Dekompositions-Teilen*

Sanierung auf **Ebene der Daten:**

- Analyse der Def nitionen und ihrer Zusammenhänge
- kontrollierte Erstellung der Datenstruktur
- Beschreibung von Datenhaltung/Dateien
 - ⇒ *Erstellung von Daten-Entwurfsobjekten*

Sanierung auf **Ebene der Präsentationsschicht:**

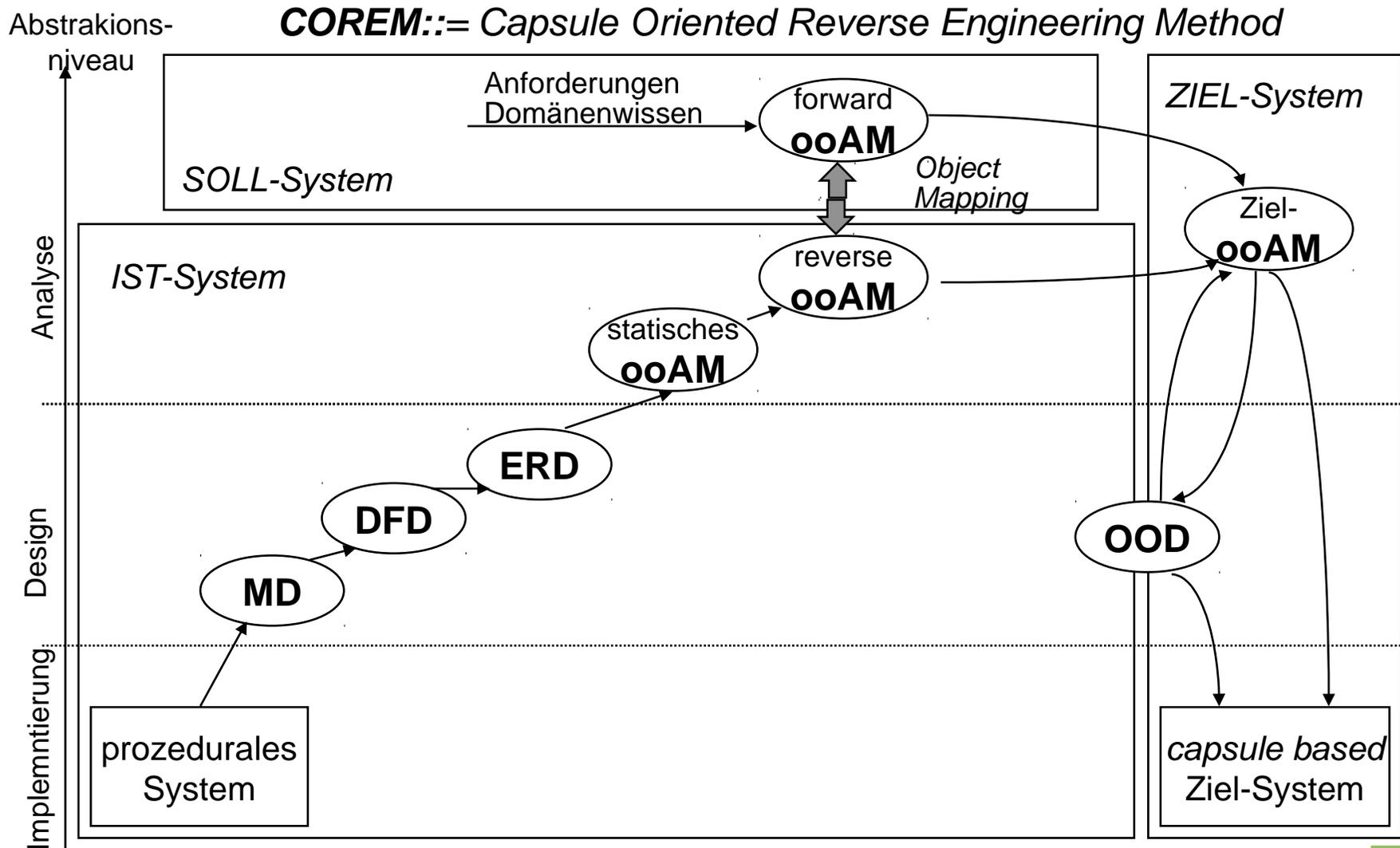
- Datenaufbereitung für Listen, Masken bzw. alle Präsentationsobjekte

Sanierung auf **Ebene der Dialog-/Hauptsteuerung**

- Hauptsteuerung der Programmablauffolge
 - ⇒ *Die beiden letzten Ebenen sind in CASE oft ungenügend unterstützt*

Bsp: Reengineering mit COREM

- Von klassischer zu objektorientierter Software -



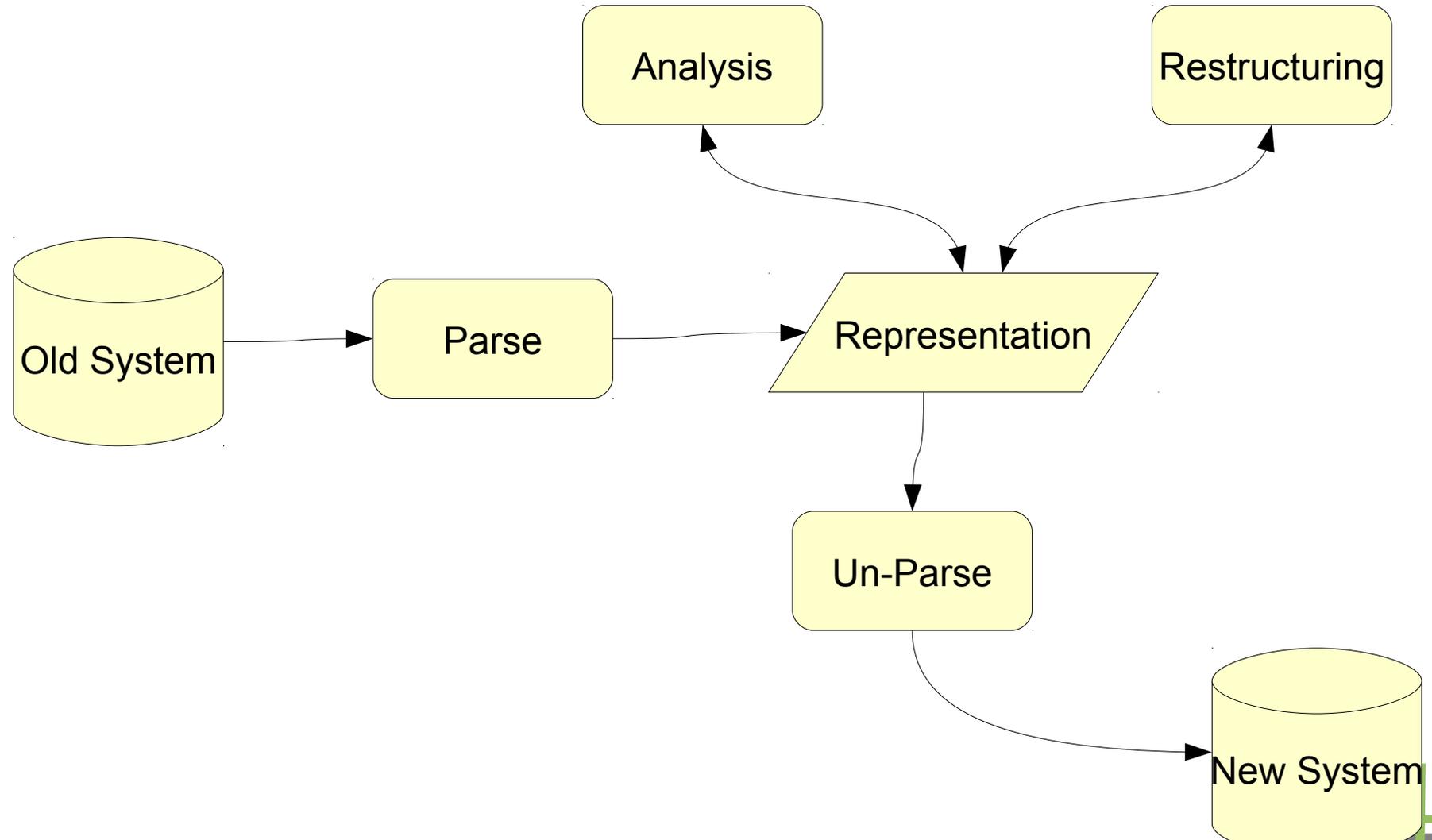
Schritte:

- ① Prozeduraler Quell-Code wird unter Zuhilfenahme der **Design-Recovery-Methoden**
 - Modular Design (MD, Structure Charts)
 - Datenfluss-Diagramme (DFD, Funktionsmodellierung)
 - Entity-Relationship-Diagramme (Datenstrukturmodellierung)
 - statische Klassen- und Objektdiagrammeüberführt in *reverse* generiertes, objektorientiertes Anwendungsmodell, **reverse ooAM**
- ② Auf anderem unabhängigen Weg wird über die Anforderungsanalyse mittels des *Reuse Engineer* ein *forward* generiertes, objektorientiertes Anwendungsmodell, das **forward ooAM** erzeugt.
- ③ Abbildung der Objektkandidaten des *reverse ooAM (Anwendungsmodell)* auf die Objekte des *forward ooAM*. Als Ergebnis des Vergleichs wird ein objektorientiertes Ziel-Anwendungsmodell synthetisiert, das **Ziel-ooAM**.
- ④ Damit weitere Objekte zwischen Ziel-ooAM und Quellcode zugeordnet werden können, wird über einen zusätzlichen OOD-Schritt ein objektorientierter **Ziel-Entwurf** erzeugt.
- ⑤ Auf Basis des Ziel-ooAM kann eine ReTransformation auf die Quell-Code-Ebene durchgeführt werden.

55.3 Werkzeuge für das Reengineering

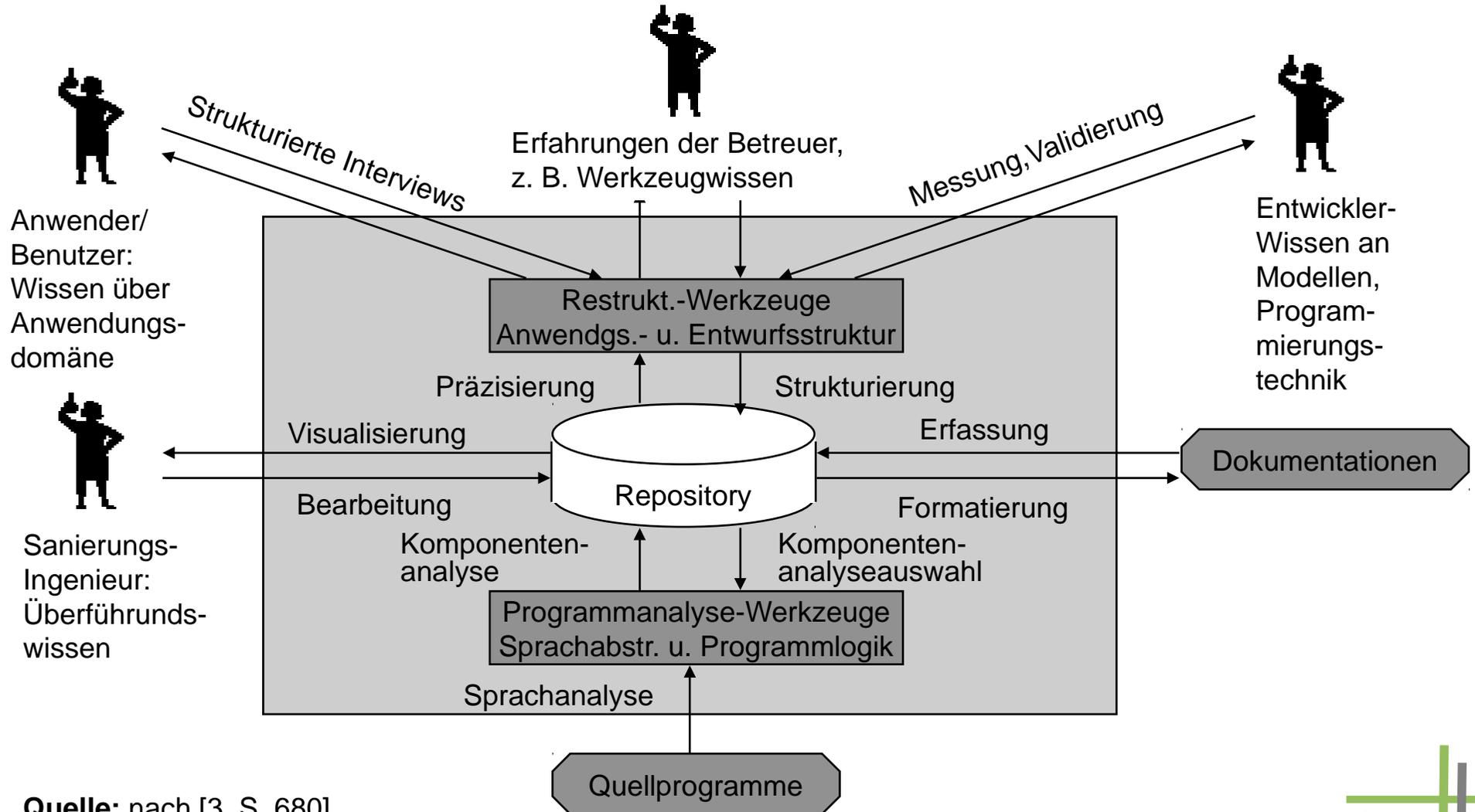


Struktur eines Reengineering-Werkzeuges



Zusammenwirken in CARE-Umgebung

CARE: Computer aided Reverse Engineering oder auch **Reengineering**



Quelle: nach [3, S. 680]

Softwarewerkzeuge sind für das Reengineering unerlässlich, weil die Programmanalyse und anschließende Synthese manuell enorm aufwendig und unzuverlässig wäre:

▶ **Werkzeuge zur Programmanalyse**

- Datenflussanalyse
- Aufrufgraphanalyse
- Daten-/Programmlogik-Tracer
- Cross-Referenzer

▶ **Messwerkzeuge**

- Metrik-Analysatoren
- Qualitätsanalysatoren
- Überwachungs-Monitore von Programmstandards

▶ **Restrukturierungswerkzeuge**

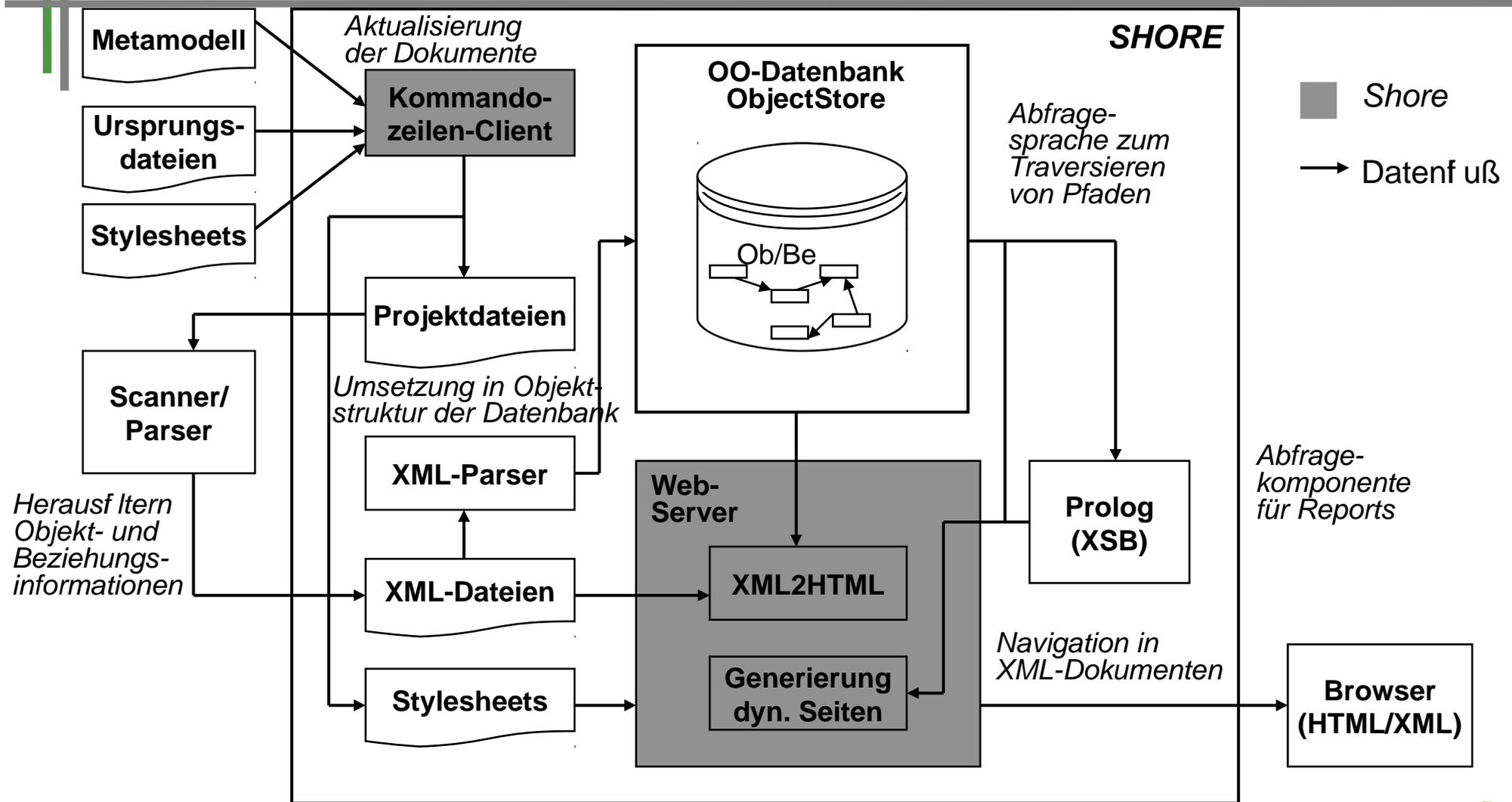
- für Verarbeitungslogik und Namenskonventionen
- Reformatierungswerkzeuge/Beautif er

▶ **Decomposer (Mustererkenner, Parser, Analysatoren)**

- Mustersuche mit *grep*, *awk* oder *perl*
 - Syntaxanalyse mit Scanner & Parser (u.a. *lexx/yacc*)
 - semantische Analysatoren
- } für Logik- und Datenrekonstruktion

Quelle: nach McClure, C.: Software-Automatisierung - Reengineering - Repository - Wiederverwendbarkeit,
Carl Hanser Verlag 1993 S. 26 ff

Bsp: Dokumenten-Renovierungs-Werkzeug SHORE von sd&m



Quelle: Keipinger, D.: Software-Renovierung; in Brössler, P., Siedersleben, J.(Hrsg.): Softwaretechnik; Hanser Verlag 2000

DMS commercial toolkit

- ▶ <http://www.semanticdesigns.com/Products/DMS/WhyDMSForSoftwareQuality.pdf>
- ▶ <http://www.semanticdesigns.com/Products/DMS/SimpleDMSDomainExample.html>
- ▶ <http://www.semanticdesigns.com/Company/Publications/DMS-for-ICSE2004-reprint.pdf>
- ▶ Employs
 - a graph rewriting language to change the code
 - attribute grammar evaluators for computing custom analyses over ASTs, such as metrics
 - control- and data- flow analysis
 - local and global pointer analysis
 - whole-program call graph extraction
 - symbolic range analysis by abstract interpretation.

```
rule
  simplify_conditional_assignment(v:left_hand_side,e1:expression,e2:expression)
    :statement->statement
  = " if (\e1) \v=\e2; else \v=e3; "
->
  " \v=\e1:?\e2:\e3; "

if no_side_effects(v);
```

- ▶ http://en.wikipedia.org/wiki/DMS_Software_Reengineering_Toolkit

Reengineering-Werkzeuge

GUPRO

Universität Koblenz

Querywerkzeuge, Metriken, Analysen

Bauhaus

Universität Bremen,
Universität Stuttgart

Grössere Analyse- und Metriksuite

DMS Design
Maintenance
System

<http://www.semanticdesigns.com/Products/DMS/DMSToolkit.html>

http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

http://en.wikipedia.org/wiki/Dynamic_code_analysis

- ▶ CARE-Werkzeuge verbessern Programmlesbarkeit und vereinfachen Programm-logik
 - Durch Verringerung von Test-/Fehlersuchzeiten Senkung des Wartungsaufwandes um 20 bis 25 %
 - Gute Unterstützung für Reformatierung und damit Senkung der Programmkomplexität
 - Verbesserung der Nachdokumentation
- ▶ CARE-Werkzeuge steigern die Anzahl der restrukturierten, konvertierten und redokumentierten Anweisungen von 70 auf 2000 Statements pro Tag
 - Vollautomatische Rekonstruktion mit Redefinition und Programmsanierung nur sehr eingeschränkt möglich
- ▶ Verlängerung der Lebensdauer von Altsystemen, damit Sicherung der Software-vermögenswerte
- ▶ Oftmals geben wegen der hohen Kosten und fehlenden Schnittstellen die Vorlieben zu Neuentwicklung und Standardsoftware den Ausschlag.

Quelle: nach Stahlknecht, P., Drasdo; A.: Methoden und Werkzeuge zur Programmsanierung; Wirtschaftsinformatik, 37(1995), S. 160-174



The End