# 74. 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 12-0.2, 31.01.13

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

In 2012/13 weggelassen
nur zur Info



# 74.1 Aufgaben von Wartung und Reengineering





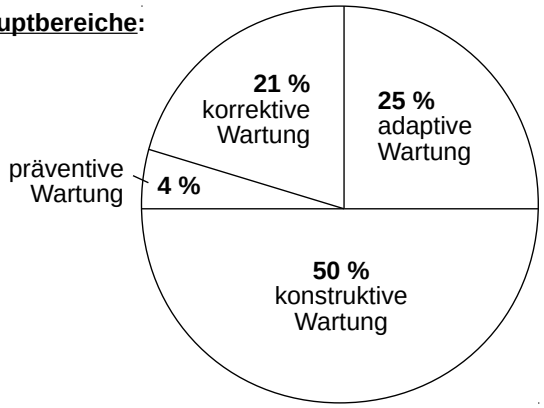
## Hauptbereiche der Wartung



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

**Software-Wartung** ist die Modif kation 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].



## Pro- und Kontra der Wartung



**Pro:**
- Programme werden robust und zuverlässig
- Wartung ist billiger als Neuentwicklung (Risiko)

**Kontra:**
- Fluktuation der Entwickler <-- *Wissensmonopol*
- unvollständige bzw. fehlende Dokumentation
- fehlende Spezif kation oder Entwurfsbeschreibung
- keine transparenten Programme (keine Verständlichkeit, Seiteneffekte)
- monolythische Programmstrukturen, zunehmende Probleme
- Erhaltung veralteter Programmiersprachen, veralteter Technologie
- fehlende Werkzeugunterstützung der Wartung
- Wartung ist teuer, Arbeiten sind unbeliebt und nicht attraktiv
- schlechte Planbarkeit und Managementprobleme

**Ausweg:**

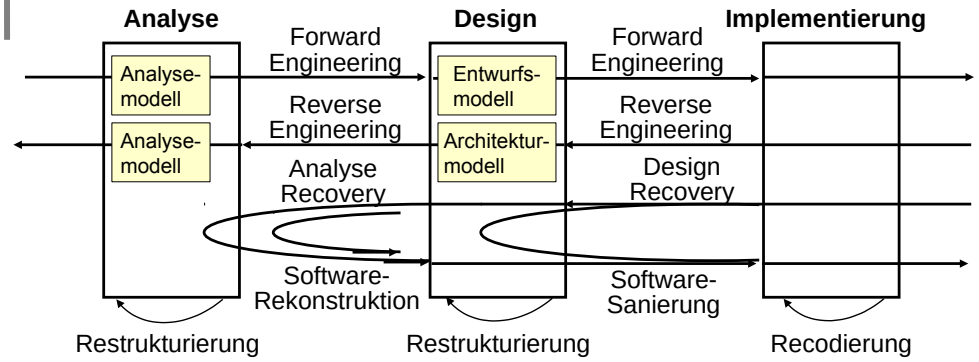
**(1) Software-Sanierung:** strukturell kontrollierbare Entfernung (Migration, Recodierung)

**(2)** Software-Umkonstruktion (**Reengineering)**: keine strukturelle Beziehung zum Original

**Quelle:** nach [3, S. 663 - 679]

## Kategorien und Modelle im Reengineering






**Software Reengineering: Reverse Engineering (***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



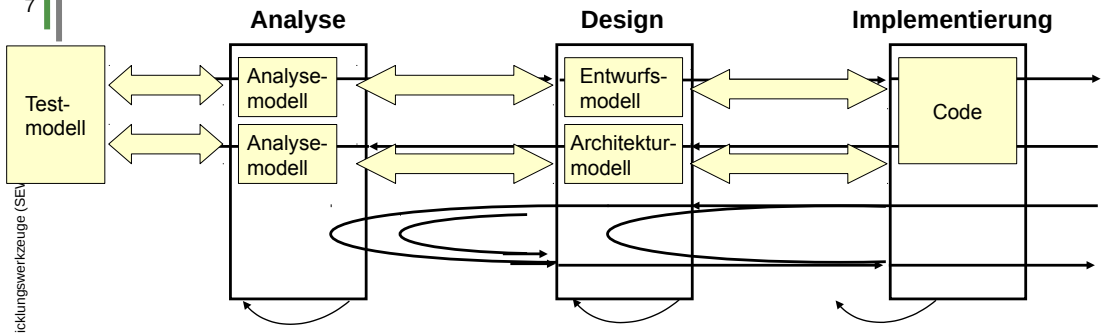
- ***Vorwärtskonstruktion (Forward Engineering):*** Anwenden von Methoden und Werkzeugen der Softwareentwicklung, um **Software-Dokumente** in eine implementierungsnähere Form zu bringen.
- ***Rückwärtskonstruktion (Reverse Engineering):*** Identif zierung der einzelnen Komponenten eines **Software-Dokuments** (Quelltext, Klassen-, Modulstruktur, ERD,...) und ihrer Beziehungen durch eine methodische Analyse auf abstrakterem Niveau.
- **Umkonstruktion (Reengineering)** verknüpft Rückwärts -mit Vorwärtskonstruktion.
- ***Design Recovery:*** Wiedergewinnung einer vollständigen, abstrakten **Entwurfs** bzw. **Architektur** eines Altsystems unter Einbeziehung von **Domänenwissen** (relevante Informationsquellen, Anwenderwissen).
- ***Analyse Recovery:*** wie oben, aber Wiedergewinnung einer vollständigen, abstrakten **Analyse-Modells**
- ***Restrukturierung:*** Veränderungen/Umstrukturierungen von Modellen, 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



## Reverse Engineering von Modellabbildungen






- Um ein Altsystem mit einem QM zu versehen, muss der Code, die Analyse, Architekturmodelle miteinander und mit den Testmdoellen verbunden werden (model mappings)
- QM wird erst möglich, wenn alle Modelle miteinander systemisch verknüpft sind

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



## 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
- ***Oberf ächenprobleme*** bei Übergang zu anderen Bildschirmen, Toolkits bzw. zu anderen ereignisgesteuerten GUIs
- ***Plattformprobleme*** bei unverträglichen Hardware- bzw. OS-Wechseln

| Problemgruppe \ Renovierungstyp | Design Recovery (Redesign) | Restrukturierung | Portierung |
|---|---|---|---|
| ***Systemstruktur*** | ++ | + | - |
| ***Datenstruktur*** | + | ++ | - |
| ***Oberf äche*** | + | + | + |
| ***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, systemspezif scher, DB-spezif scher- 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
  - Niveauanhebung 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



# 74.2 Vorgehen des Reengineering





# Schichtenmodell des Reengineering



- Sanierung auf Ebene des Anwendungs-Codes:
  - Herstellung von Strich- bzw. Einrückdiagrammen
  - Umsetzung in Zwischen-/Pseudocode (bzw. Struktogramme für Steuerfl.)
  - Elimination redundanter Codeteile bzw. wilder Sprünge (GOTO)
  - Extraktion von Automaten oder Statecharts
  - ↙ Auffinden anwendungsorientierter Code-Teile
- Sanierung auf Ebene der Programmsteuerung:
  - Trennung Definitionen von Anweisungen
  - Auffinden Programmrahmen, Abspaltung von (ext.) Dienstroutinen
  - Festlegen Aufrufhierarchie
  - ↙ Bestimmung Funktionen(Aktionen) mit Dekompositions-Teilen
- Sanierung auf Ebene der Daten:
  - Analyse der Definitionen 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

**Quelle:** nach Thurner, R.: Reengineering mit *Delta*; in Balzert, H. (Hrsg.):CASE - Systeme und Werkzeuge (2.Auf age); BI-Wissenschaftsverlag Mannheim 1990

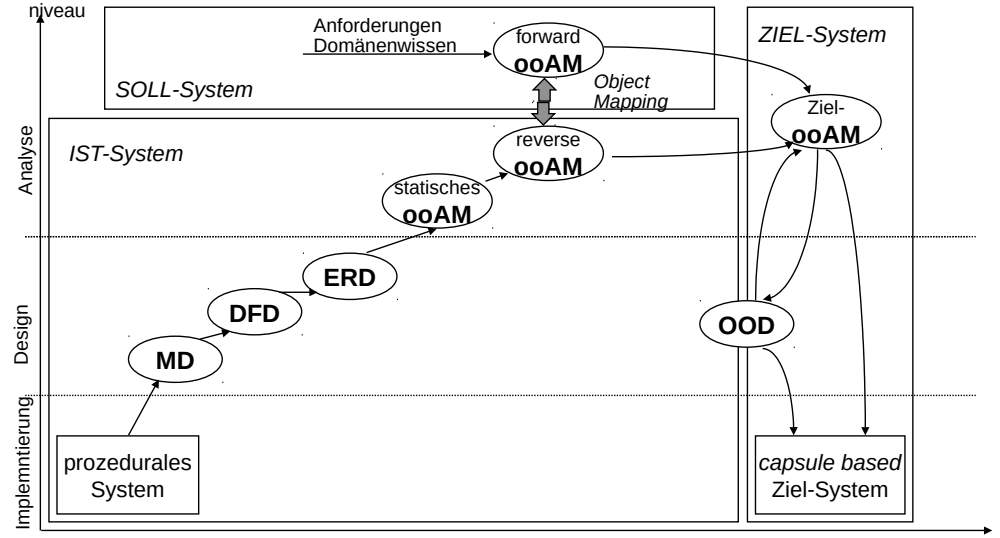


# Bsp: Reengineering mit *COREM*

## *- Von klassischer zu objektorientierter Software -*



***COREM::**= Capsule Oriented Reverse Engineering Method*




**Quelle:** Klösch, R., Gall, H.: Objektorientiertes Reverse Engineering; Springer Verlag 1995

## Vorgehensweise von *COREM*



1. Prozeduraler Quell-Code wird unter Zuhilfenahme der **Design-Recovery-Methoden**
   - Modular Design (MD, Structure Charts)
   - Datenf uss-Diagramme (DFD, Funktionsmodellierung)
   - Entity-Relationship-Diagramme (Datenstrukturmodellierung)
   - statische Klassen- und Objektdiagramme

   überführt 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.



## Reengineering ist modellbasiert



Da das Reverse Engineering mit Architektur- und Analysemodellen endet,
und die Vorwärtskonstruktion mit solchen beginnt,
ist das **Reengineering** ein modellbasierter Entwicklungsprozess.



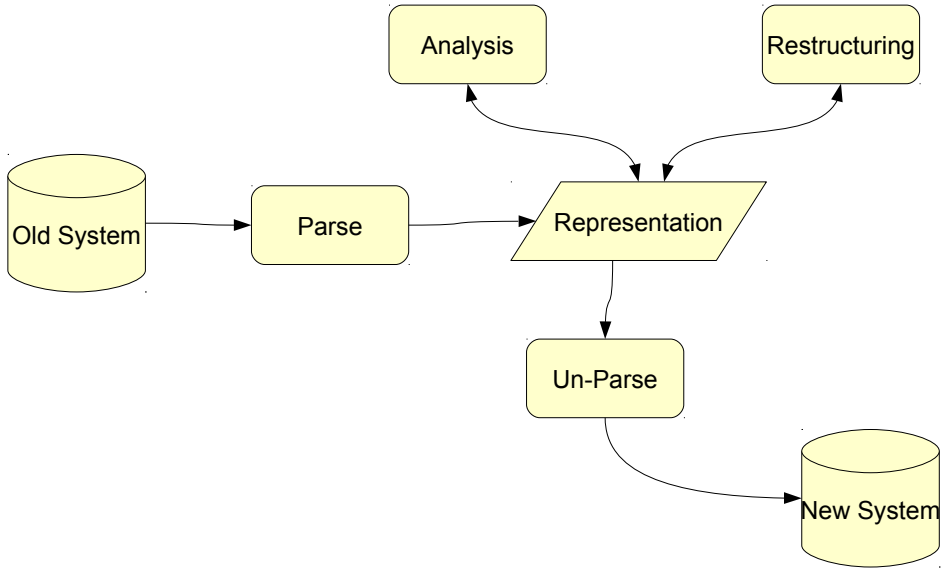
# 74.3 Werkzeuge für das Reengineering





## Struktur eines Reengineering-Werkzeuges

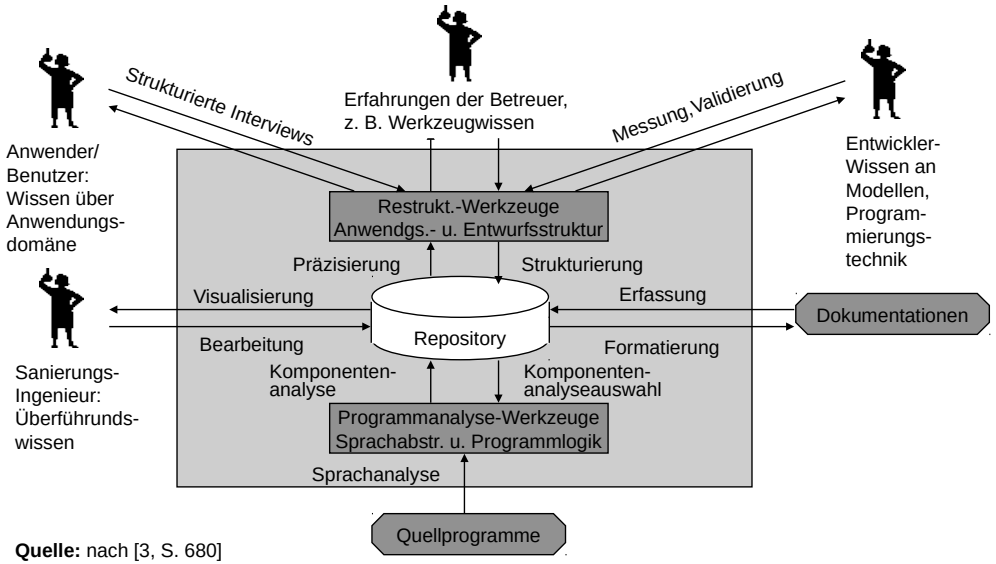




## Zusammenwirken in CARE-Umgebung

17

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




**Quelle:** nach [3, S. 680]



## Reengineering-Werkzeuge

18

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

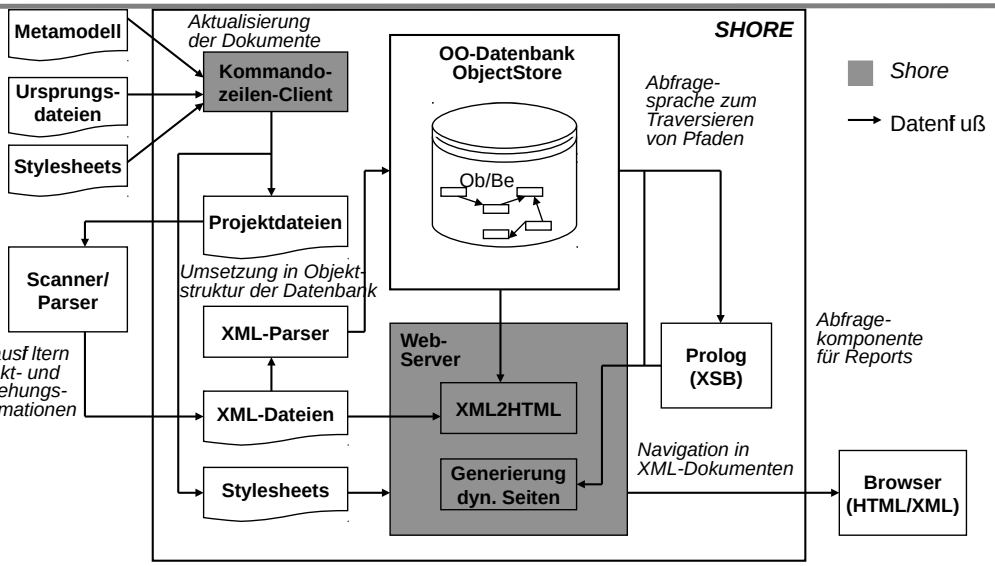
- **Werkzeuge zur Programmanalyse**
  - Datenf ussanalyse
  - Aufrufgraphanalyse
  - Daten-/Programmlogik-Tracer
  - Cross-Referenzer
- **Messwerkzeuge**
  - Metrik-Analysatoren
  - Qualitätsanalysatoren
  - Überwachungs-Monitore von Programmstandards
- **Restrukturierungswerkzeuge**
  - für Verarbeitungslogik und Namenskonventionen
  - Reformatierungswerkzeuge/Beautif er
- **Decomposer (Mustererkenner, Parser, Analysatoren)**
  - Mustersuche mit *grep, awk* oder *perl*
  - Syntaxanalyse mit Scanner & Parser (u.a. *lexx/yacc*) } für Logik- und Datenrekonstruktion
  - semantische Analysatoren

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



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

19




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



## DMS commercial toolkit

20

- 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

```
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



## Ergebnisse des Reenginering



- 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