Reverse Engineering

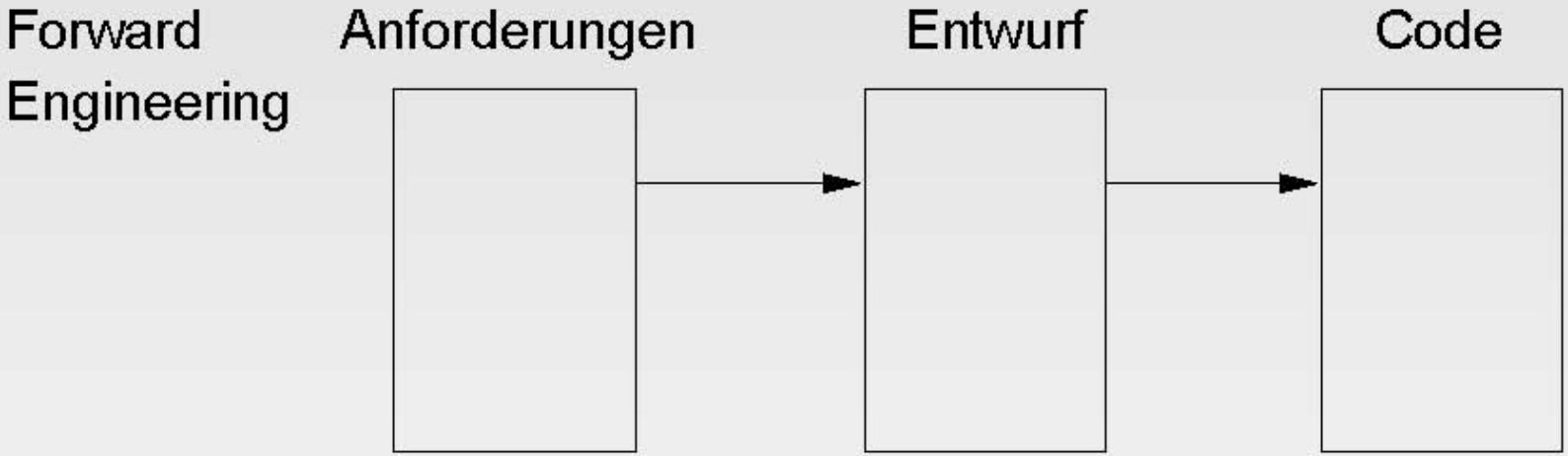
License Restrictions:

> *"Customer may not reverse engineer, disassemble, decompile, or translate the Software, or otherwise attempt to derive the source code of the Software."*

> *"To me the flow of time is irrelevant. You decide what you want. I then merely make sure that it has already happened."*
>
> *– The Hitch Hiker's Guide to the Galaxy*

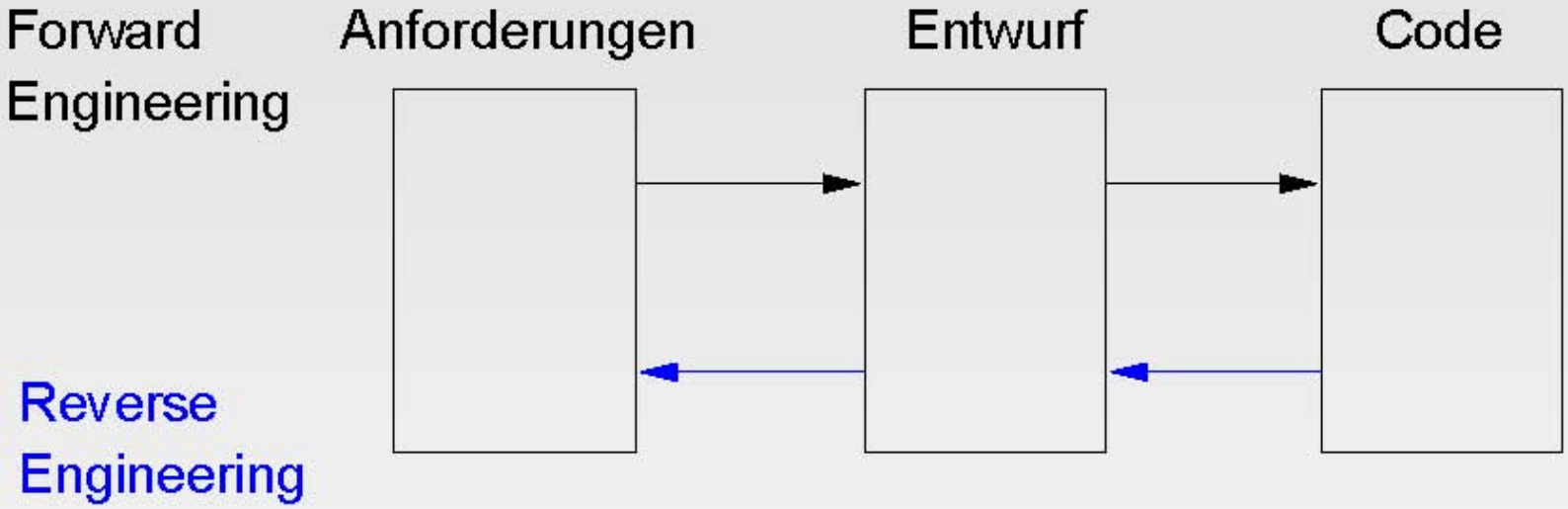
Reverse Engineering (Chikofsky und Cross II. 1990)

Identifikation der Systemkomponenten und deren Beziehungen, mit dem Ziel das System in einer anderen Form oder auf höherem Abstraktionsniveau zu beschreiben.



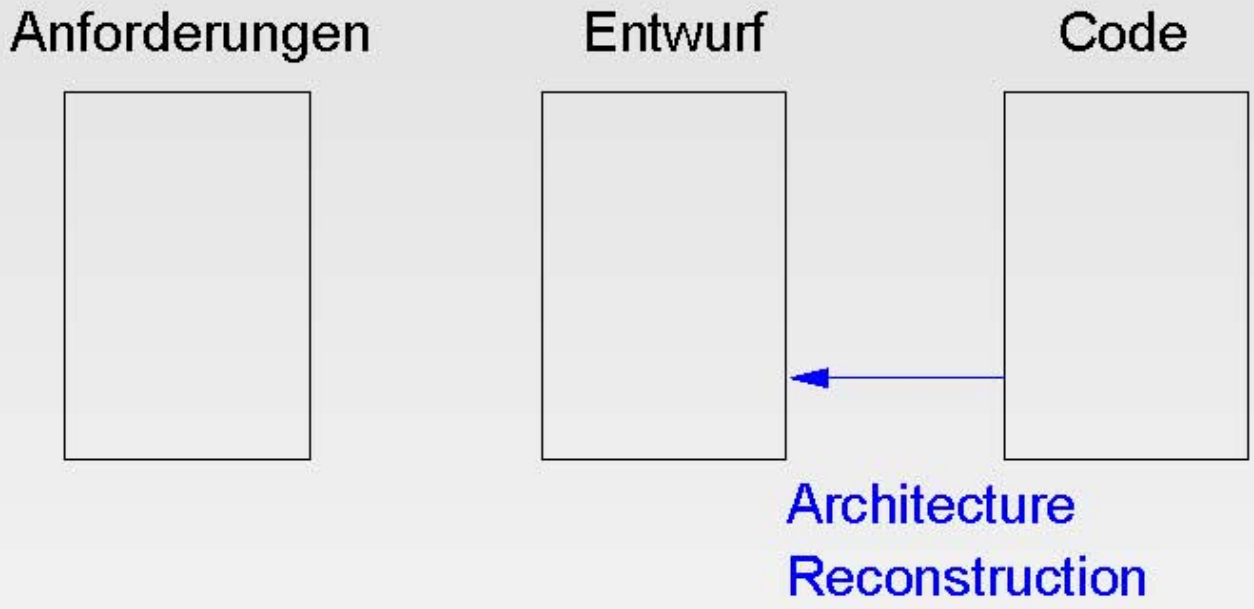
Reverse Engineering (Chikofsky und Cross II. 1990)

Identifikation der Systemkomponenten und deren Beziehungen, mit dem Ziel das System in einer anderen Form oder auf höherem Abstraktionsniveau zu beschreiben.



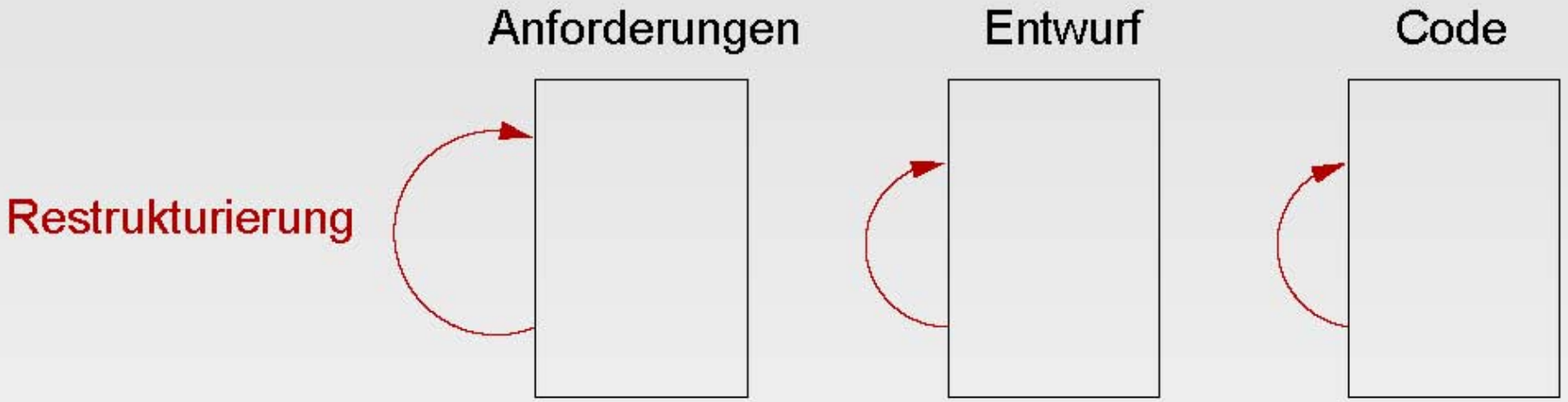
Architekturrekonstruktion

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



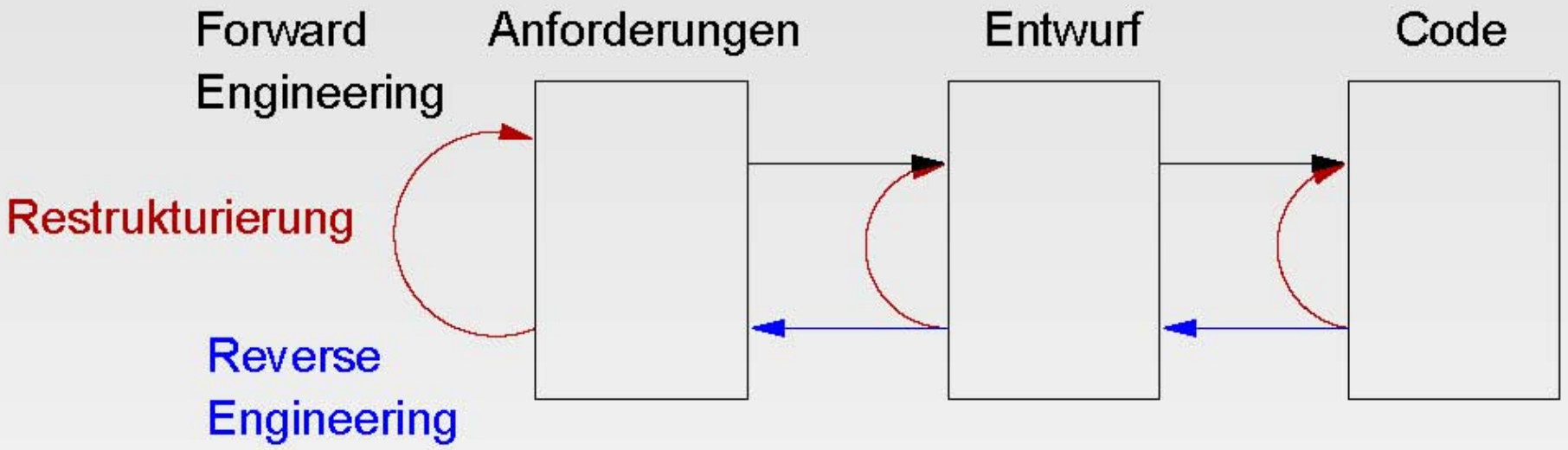
Restrukturierung (Chikofsky und Cross II. 1990)

Transformation einer Repräsentation in eine andere auf derselben Abstraktionsebene, ohne Änderung der Funktionalität des Systems.



Reengineering (Chikofsky und Cross II. 1990)

Untersuchung (Reverse Engineering) und Änderung des Systems, um es in neuer Form zu implementieren.
Synonyme: Renovation, Reclamation.



Reengineering-Varianten

Reines Reengineering:

- das System soll lediglich restrukturiert werden
- keine Funktionalität kommt hinzu / wird geändert

Erweiterndes Reengineering:

- System wird zunächst analysiert und/oder restrukturiert, um dann Funktionalität zu ändern oder hinzuzufügen

Wrapping

Das System erhält eine neue Schnittstelle, bleibt aber ansonsten unangetastet.

- Interimslösung, wenn System bald ausgewechselt werden soll.
- Notwendig, wenn das System Subsystem per Subsystem geändert werden muss.
- Oft eingesetzt, um zeichenorientierte Anwendungen mit einer graphischen Benutzerschnittstelle zu versehen.
- Organisatorische Gründe:
  - „altes“ Wartungspersonal behält Kontrolle über Wartung „ihres“ Systems
  - „junges“ Wartungspersonal hat „moderne“ Sicht

Business Process Reengineering

"Business process reengineering is the search for, and the implementation of, radical change in business process to achieve breakthrough results."

– T.A. Stewart, Fortune Magazine'93.

Business Process Reengineering

Etwas sachlicher:

- Wiedergewinnung der tatsächlichen Abläufe der Geschäftsprozesse (Workflow) (z.B. Bestellungswesen, Auftragsabwicklung, etc.)
- Überarbeitung und Neudefinition der Abläufe

Ziele des Reverse Engineerings

- Kontrolle der Komplexität
- Gewinnung alternativer Sichten
- Wiedergewinnung verlorener Information
- Erkennung von Seiteneffekten
- Schaffung höherer Abstraktionen
- Unterstützung von Wiederverwendung

Häufige Reengineering-Aufgaben

- Plattformanpassung
- Änderung der Programmiersprache
  - neuer Standard, neue Sprache, neues Paradigma
- Benutzerschnittstelle zeichenorientiert hin zu graphisch orientiert
- Mainframe hin zu Client-Server-Architektur
- Datenbankumstellung

Präventive Maßnahmen, wie z.B. Verbesserung des Information Hidings, Remodularisierung, sind eher selten.

Mass Changes

Änderungen, die weite Teile des Codes bzw. sehr viele Systeme betreffen.

- Einführung des Euros
- Legacy to Internet Interoperability (Electronic Commerce)
- Änderung von Repräsentationsformen:
  - Y2K Problem
  - Erweiterung des Bar-Codes
  - Unix-Datum

Reengineering in der Praxis

Gegenwärtig zur Verfügung stehende Werkzeuge:

- grep
- symbolische Debugger
- Cross-Reference-Tools (fertige Parser, die Basisinformationen extrahieren; z.T. mit Visualisierung durch Graphen)
- UML-CASE-Tools, die Klassendiagramme extrahieren
- programmierbare Analyse- und Transformationsumgebungen basierend auf abstrakten Syntaxbäumen (z.B. Refine von Reasoning Systems, DMS von Semantic Designs, RainCode)

Übersicht über diese Vorlesung

- Programmanalysen und -repräsentationen
- Program-Slicing
- Refactoring und Transformationen
- Software-Produkt-Metriken
- Erkennung duplizierten Codes
- Mustersuche
- Software-Visualisierung
- Begriffsanalyse
- Analyse und Restrukturierung von Vererbungshierarchien
- Merkmalsuche
- Software-Clustering, Architekturrekonstruktion und -validierung
- Reengineering-Projekte

"Software Engineering is reengineering on the empty system."

"Software Engineering is reengineering on the empty system."

"Is it?"

Unterschiede Forward Eng. / Reengineering

Forward Engineering auf grüner Wiese

- Problem noch unklar
- Aussagen über Aufwand, Dauer, Zuverlässigkeit etc. sind schwierig
- System existiert nicht
  - Entwurf hat viele Freiheiten
  - im sauberen Entwurf gibt es keine versteckten Abhängigkeiten

Reengineering

- Problem weitgehend klar
- Idealerweise: Daten aus der Vergangenheit existieren, die Grundlage für Schätzungen darstellen
- System existiert
  - Genaue Struktur/Qualität bekannt?
  - Lösung ist durch existierendes System beschränkt
  - Änderungen können globale Auswirkungen haben (viele versteckte Abhängigkeiten)

Software Engineering & Reengineering

Reengineering beginnt oft bereits während der Erstentwicklung:

- neue Anforderungen treffen ein
- Missverständnisse und Unklarheiten werden sichtbar
- der Entwurf hat sich als unzureichend erwiesen
- Integration anderer Komponenten macht Umstrukturierungen notwendig

Weiterführende Literatur

- Chikofsky und Cross II. (1990): "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software
  definiert Terminologie; ist die begriffliche Grundlage
- Baumöl u. a. (1996): "Einordnung und Terminologie des Reengineering"
  führt z.T. deutsche Begriffe ein

Bücher I

- Demeyer u. a. (2002) stellen eine Reihe von Vorgehensweisen bei typischen Problemen des Reengineerings vor
- Müller (1997) bietet eine Einführung in verschiedene Aspekte des Reengineerings (Programmverstehen, Metriken, Sprachkonversion, Restrukturierung, Wiederverwendung, Migration zu objektorientierten Systemen, Managementaspekte)
  - obwohl meine Vorlesung 1999 in völliger Unkenntnis dieses Buches entstanden ist, ist doch eine große Überlappung der Inhalte festzustellen (das Buch beschreibt aber weniger die konkreten Techniken)
- Fowler (2000) beschreibt so genannte *Bad Smells* (Code-Anomalien) und zugehörige Refactorings, um sie zu beseitigen
- Seacord u. a. (2003) beschreiben Methoden zur Modernisierung von Anwendungssystemen; in erster Linie Prozess- und Managementfragen werden erläutert

Bücher II

- Simon u. a. (2006) beschreiben, wie man die Wartbarkeit von Systemen messen kann; das Ergebnis ist eine Einstufung in Analogie zu CMMI jedoch für die innere Produktqualität
- Sneed u. a. (2005) beschreiben organisatorische Aspekte und verschiedene Prozesse für die Wartung und Weiterentwicklung von Software
- Masak (2006) diskutiert verschiedene Aspekte der Wartung und Evolution, insbesondere Beobachtungen, Metriken, Anti-Patterns und Management
- Pigoski (1996) behandelt Probleme und Managementlösungen der Software-Wartung
- Lehner (1989) beschreibt Probleme und Managementlösungen der Software-Wartung

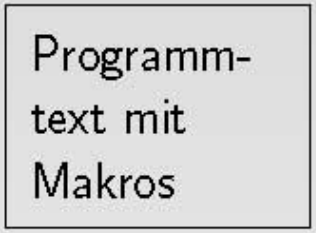
Statische Programmanalyse

2. Statische Programmanalyse
   - Compiler versus Reengineering-Werkzeug
   - Lexikalische Ebene
   - Syntaktische Ebene
   - Statische Semantik
   - Analysemöglichkeiten
   - Quellennahe Repräsentation
   - Kontrollfluss
   - Kontrollabhängigkeit
   - Datenabhängigkeit
   - Aliasing

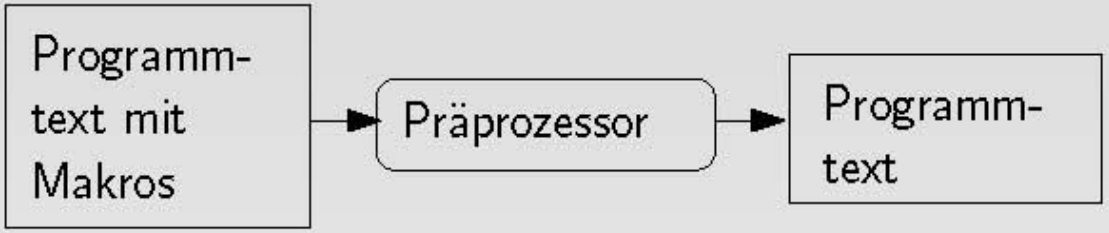
Statische Programmanalyse für das Reengineering (Compilerbau-Technologie)

- Lernziele
  - Grundlagen aus Compilerbau für grundlegende Programmanalysen
  - Unterschiedliche Anforderungen innerhalb des Reengineerings
  - Verständnis der Abstraktionsebenen von Programmdarstellungen
- Kontext
  - Grundlegende Programmanalysen sind Basis aller weiteren Analysen

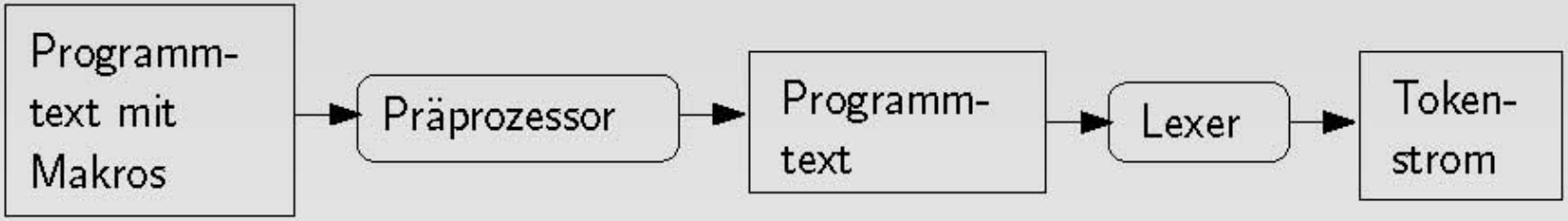
Compiler-Struktur



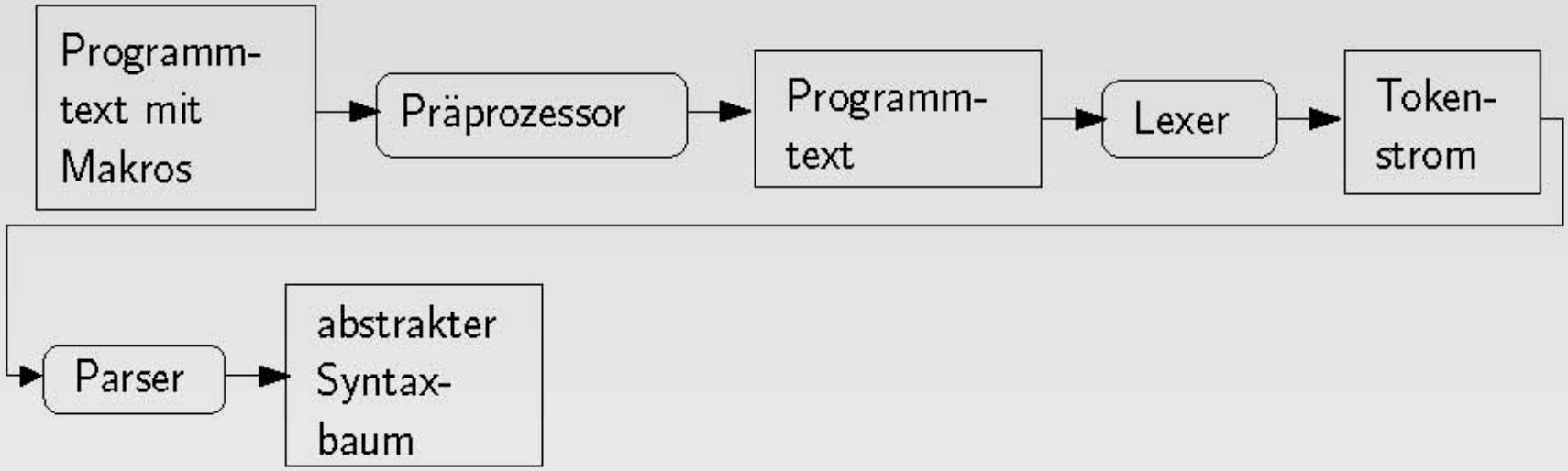
Compiler-Struktur



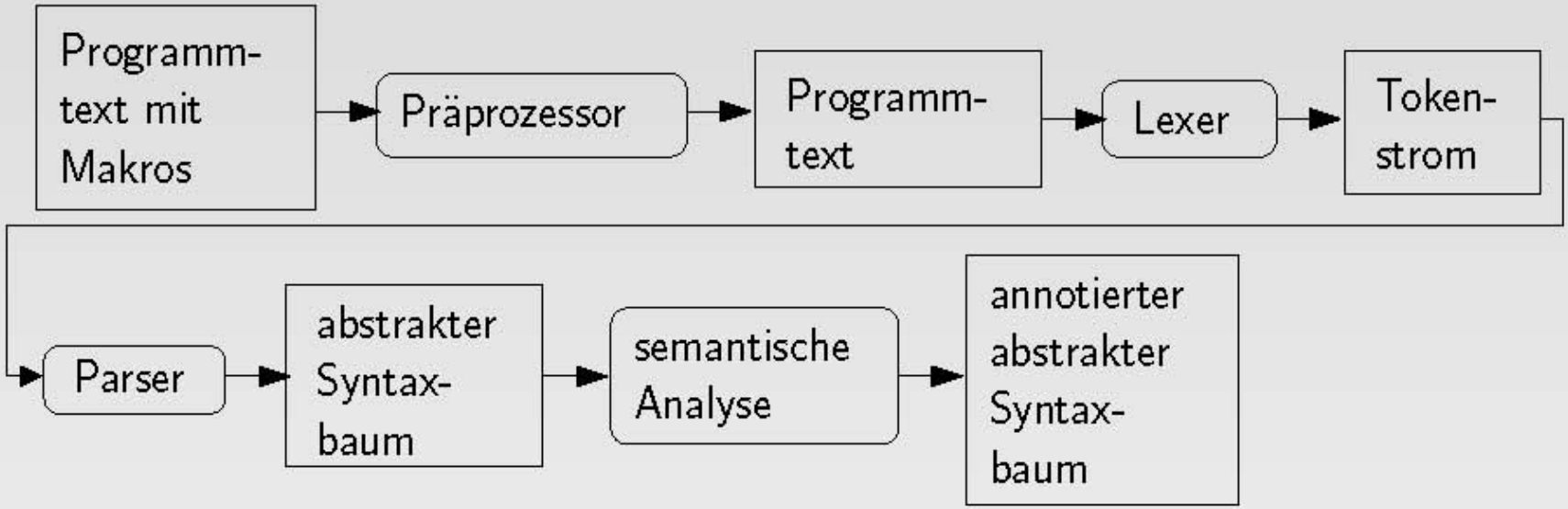
Compiler-Struktur



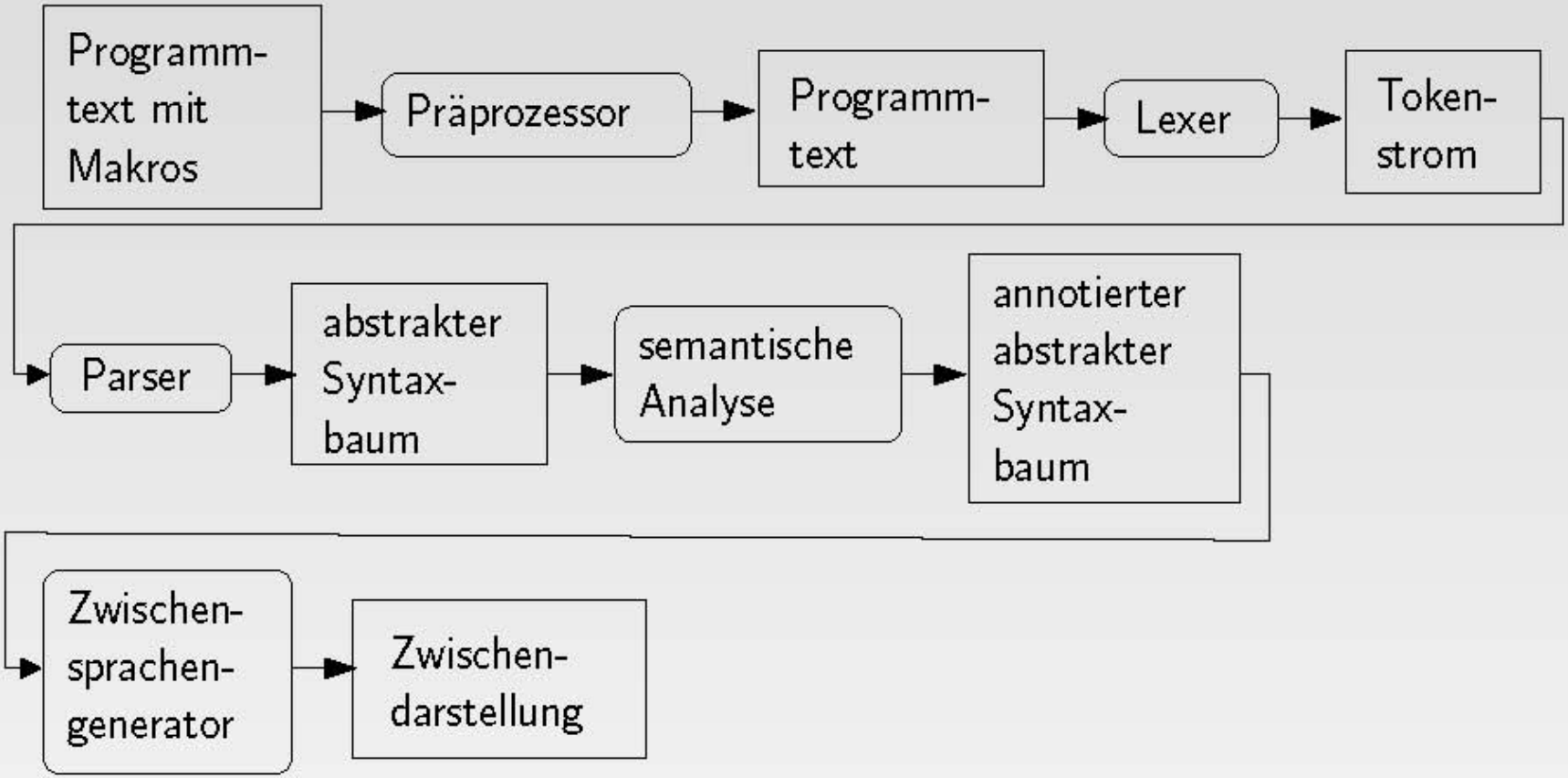
Compiler-Struktur



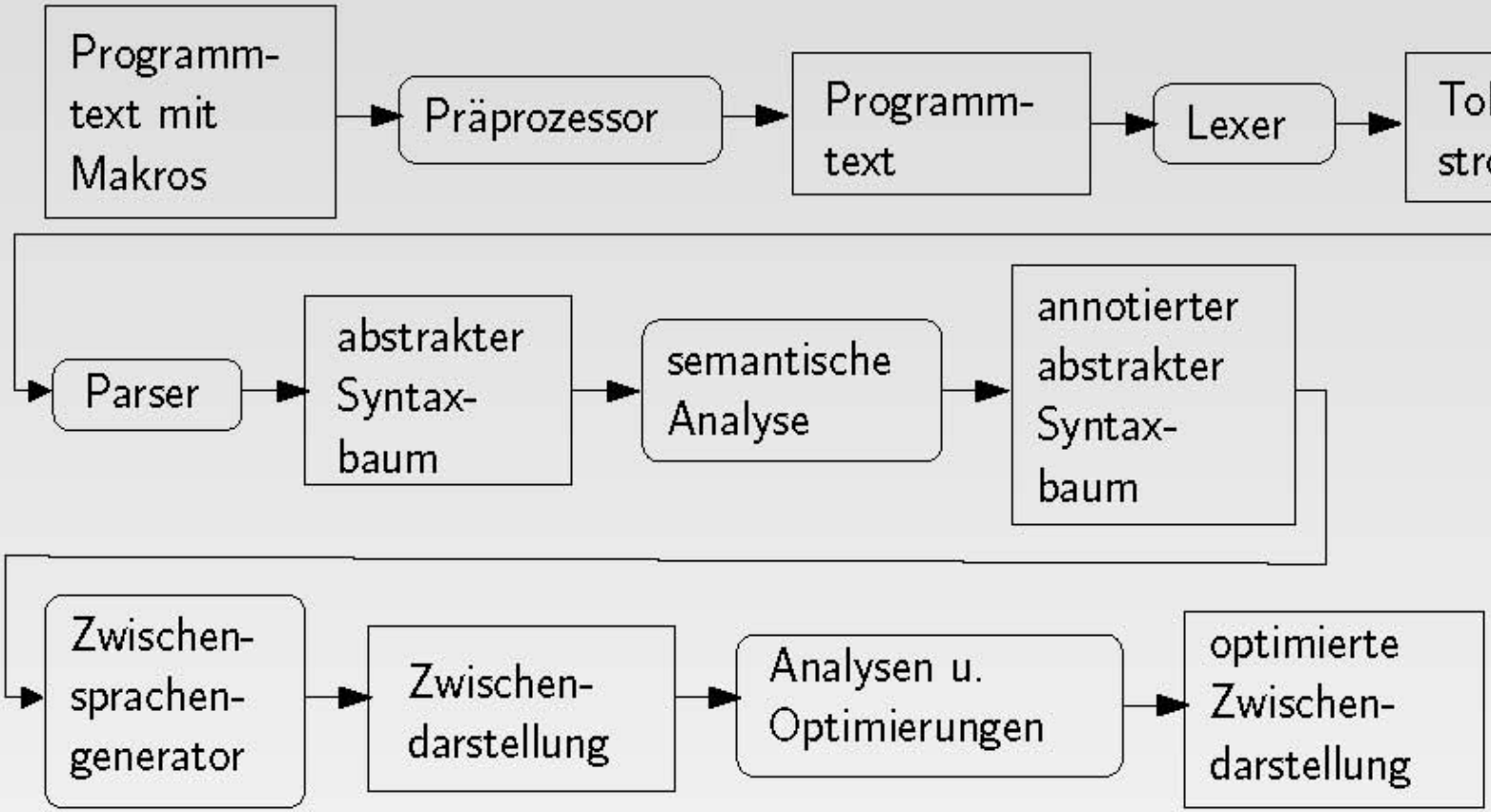
Compiler-Struktur



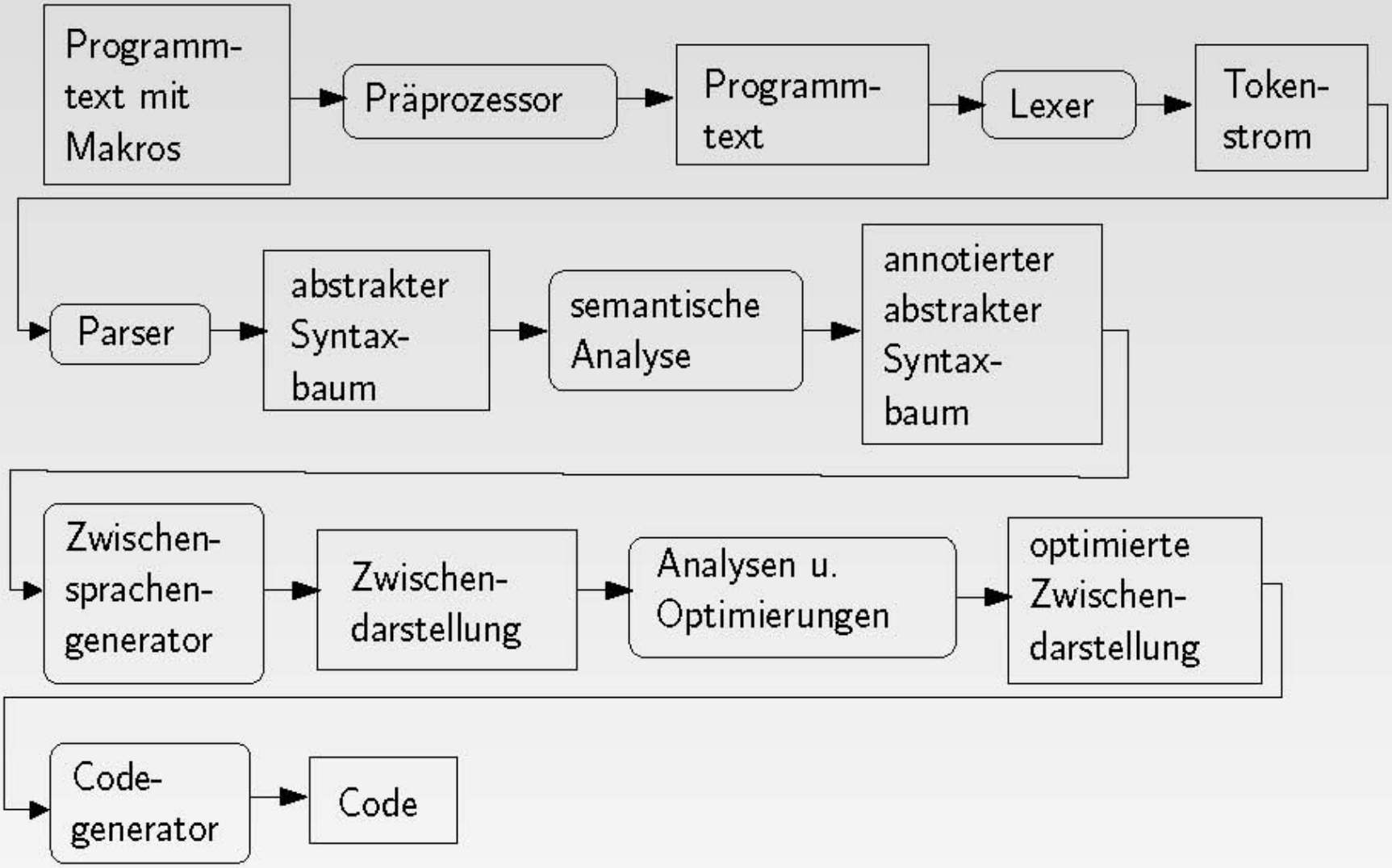
Compiler-Struktur



Compiler-Struktur



Compiler-Struktur



Compiler-Struktur