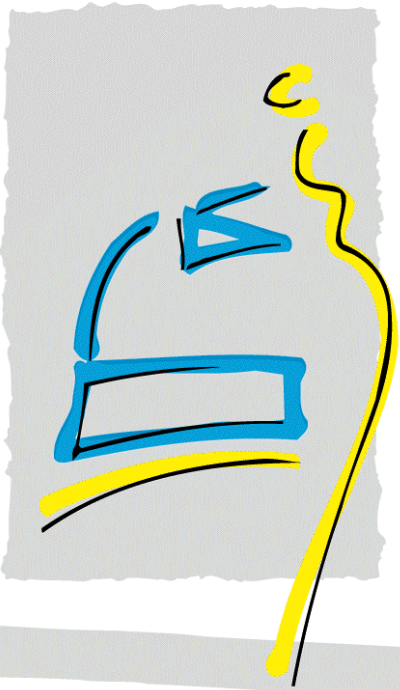
Programmstrukturen

Objektorientierte Systeme unter der Lupe

Markus Bauer

Oliver Ciupke



Übersicht

- **Einführung**
- **Methode und Werkzeug zur automatische Suche nach Strukturproblemen**
  - **Erzeugen einer Design Datenbank auf Basis des Quellcodes des Systems**
  - **Abstraktion**
  - **Suche nach Strukturproblemen mit Hilfe von Heuristiken**
- **Typische Heuristiken**
- **Fallstudie**
- **Zusammenfassung**

Einführung

- **In großen Systemen ist gute Systemstruktur unentbehrlich**
  Grund: Systemstruktur beeinflusst Entwicklungs- und Wartungsaufwand maßgeblich
- **Evolution und lange Entwicklungszeiten führen oft zu verwässerten Strukturen → Zerwartung**
- **Abhilfe: Methoden und Werkzeuge zur automatischen Untersuchung von Systemen auf Strukturprobleme auf Basis des Quellcodes**

Vorgehensweise

- **Analyse des Quellcodes und Aufbau einer Design-Datenbank**
- **Visualisierungen der Design-Datenbank**

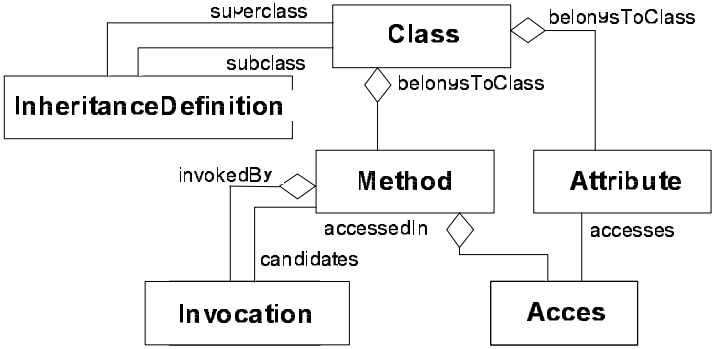


Figure 2: The Core Model

- **Anfragen an Design-Datenbank zur Suche nach Strukturproblemen**
- **Manipulation der Design-Datenbank zur Abstraktion von Implementierungsdetails und zur Informationsreduktion**

Werkzeugunterstützung

```
% Base classes should not have knowledge about
% their descendants
knowsOfDerived(Class, DerivedClass) :-
    ...
    ...
```

Werkzeugsammlung GOOSE

Problem structure

Query

Source code

Parsing

Design information

Transformation

Query language

Analysis

Foo

...

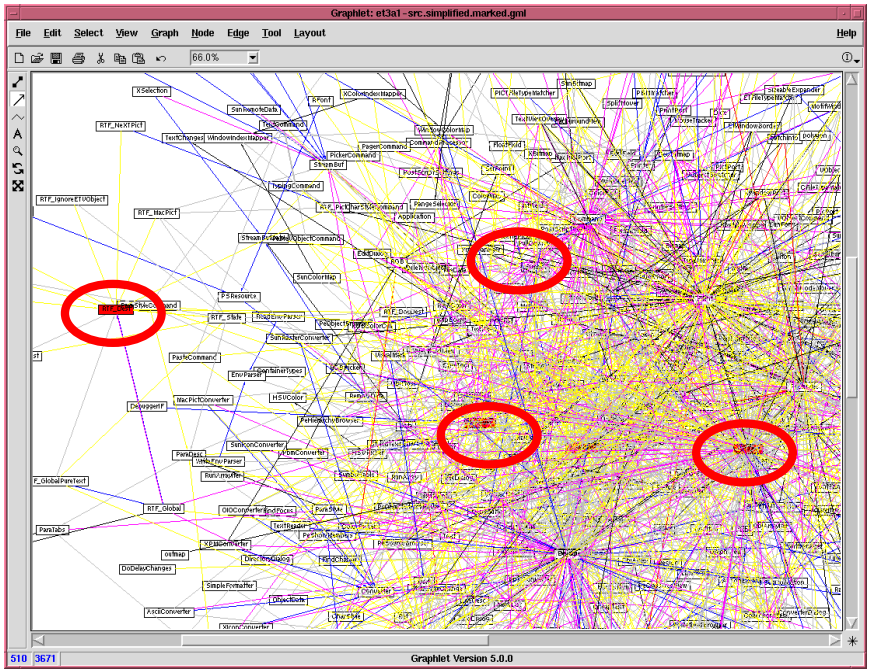
Bar

Abstraction

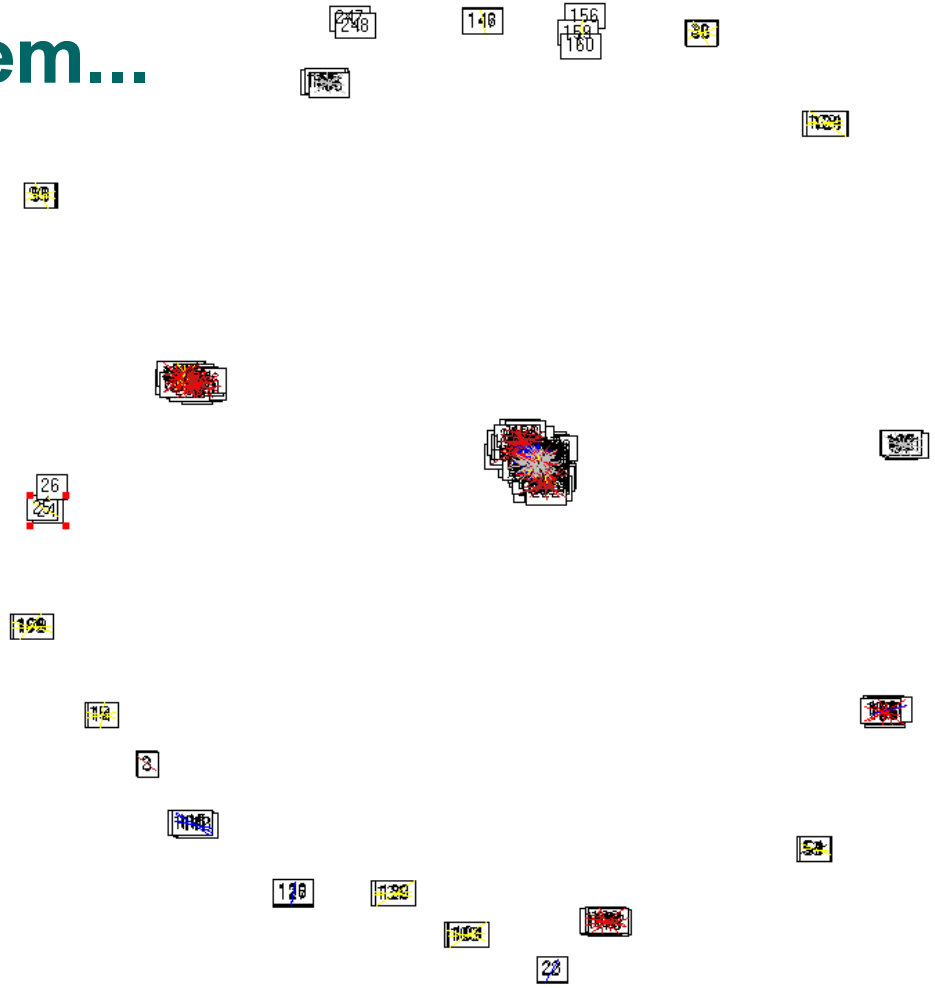
Anfragen und Heuristiken

Derived classes must have knowledge of their base class by definition, but base classes should not know anything about their derived classes.
(Heuristic 5.2 in Riel 1996)

```
% Base classes should not have knowledge about
% their descendants
knowsOfDerived(Class, DerivedClass) :-

    % Both Class and DerivedClass must be classes
    class(Class), class(DerivedClass),

    % DerivedClass is a direct or transitive
    % descendant of Class
    trans(inheritsFrom, DerivedClass, Class),

    % The base class knows its heir
    knows(Class, DerivedClass).
```

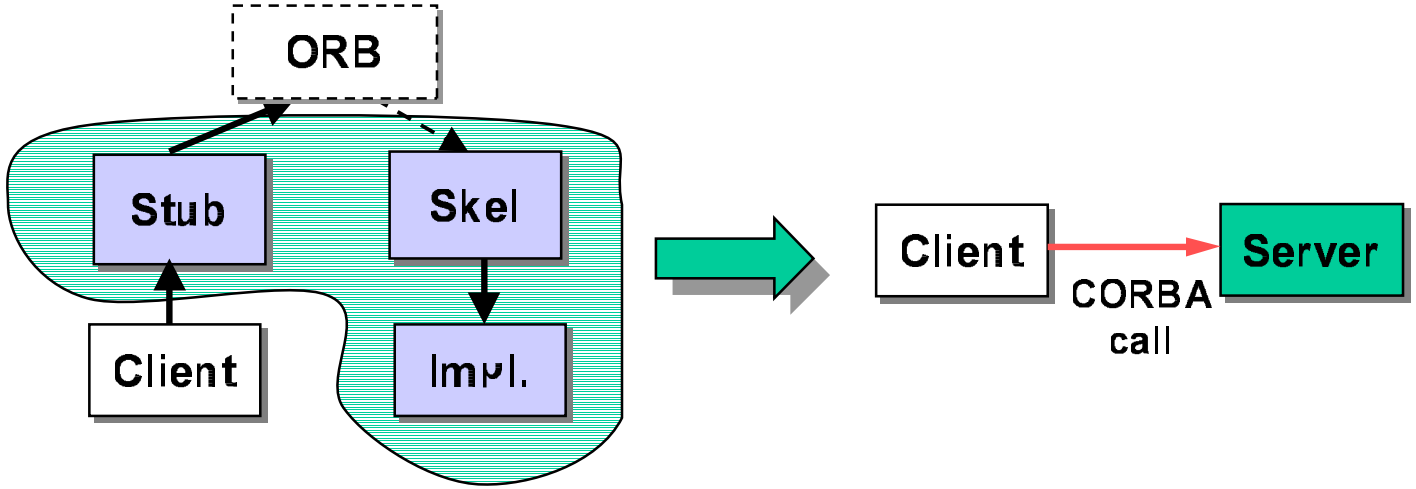


Ein CORBA-System...

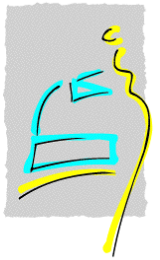


Abstraktion von Implementierungsdetails

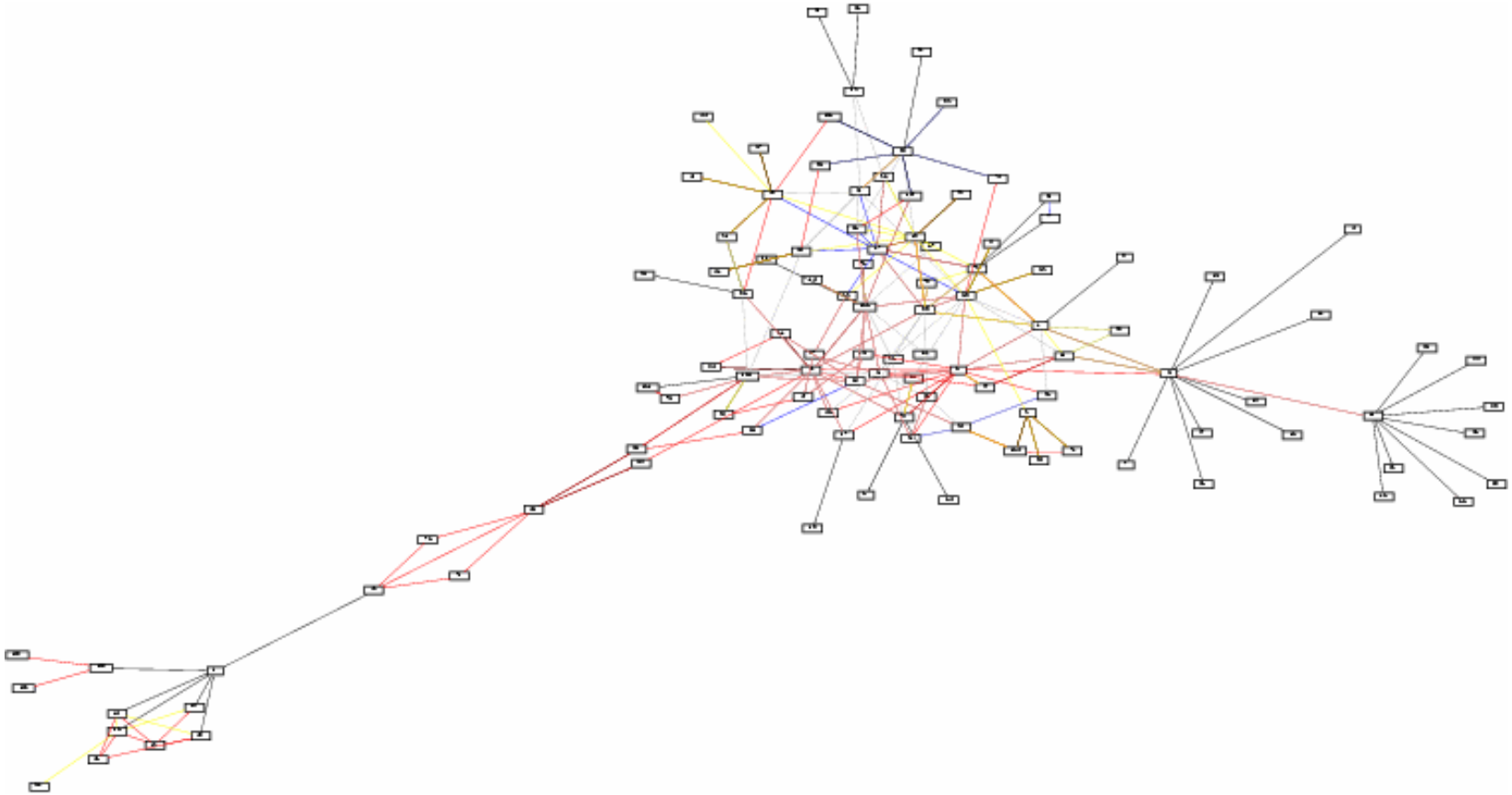
Am Beispiel CORBA:



- In Komponentenarchitekturen werden viele Abhängigkeiten durch Infrastruktur realisiert
- Gewinnung von Strukturinformationen durch Abstraktion



...nach Abstraktion



Typische Heuristiken

- Klassifizierung
  - Objekt orientierte Heuristiken
  - Heuristiken für Subsysteme
  - Technik und projektspezifische Heuristiken
- Probleme
  - Vielzahl und Widersprüchlichkeit existierender Heuristiken
  - Interessante Phänomene oft nicht abgedeckt
- Erfahrung:
  - Projekterfahrung bringt neue Heuristiken hervor
  - Kritische Stellen werden oft durch mehrere Heuristiken erfasst

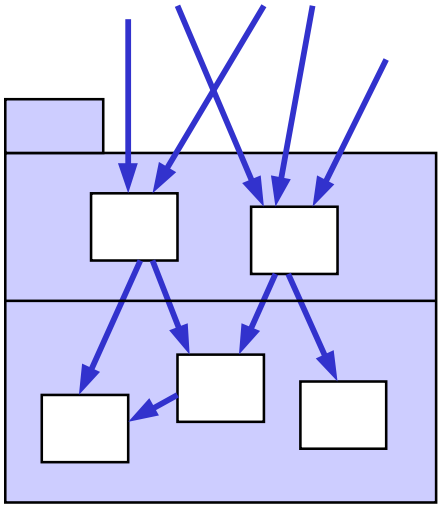
Objektorientierte Heuristiken

- **Klassen sollen nicht von Unterklassen abhängen**
- **Vererbung soll nur eingesetzt werden, wenn polymorphe Struktur vorliegt**
- **Vermeide überflüssige Abstraktionen**
- **Keine Flaschenhalsklassen**
- **Keine Gottklassen**

Bemerkung:
Für viele Heuristiken nur abschätzende Berechnung möglich!

Heuristiken für Subsysteme

- **Schlanke, explizite Schnittstellen**
- **Keine fragilen Klassen in Schnittstellen**
- **Klassen in Schnittstellen sollen möglichst wenig von Klassen in anderen Subsystemen abhängen**
- **Entkopplung von Subsystemen**
- **Keine zyklische Vererbung zwischen Subsystemen**



Technologie- und projektspezifische Heuristiken

- **EJB-Systeme:**
  - Klare Trennung von Entitäten und Aktivitäten
  - Entitäten dürfen nicht von Aktivitäten abhängen
  - Zugriff auf Entitäten nur über Zwischenschicht (Editierer)
- **Trennung von Framework- und Anwendungscode**
  - Abhängigkeiten nur von Anwendung zu Framework erlaubt!
  - Einhalten einer geschichteten Architektur

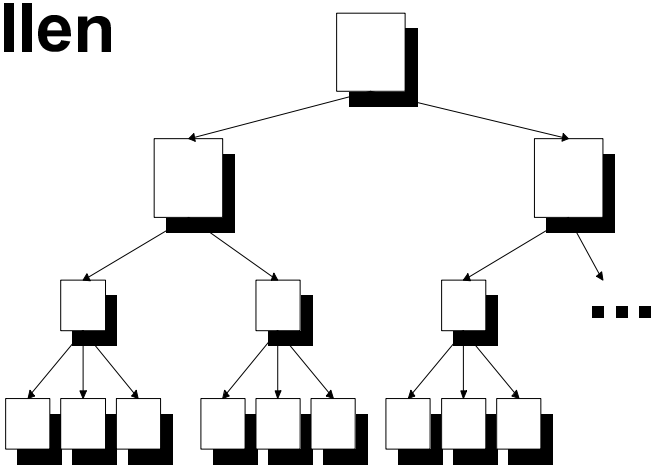
Fallstudie

- **Modernes EJB-System zum Vertragsmanagement von Versicherungen**
- **Identifikation von Strukturproblemen, um Ansatzpunkte für Restrukturierungen zu finden**
- **Ergebnis: Report mit potenziellen Strukturproblemen**

70 Subsysteme

600 Geschäftsobjekte

6000 Klassen und EJBs
– 1 Mio LOC



Fallstudie – Ergebnisse

- **Methode und Werkzeuge für größere OO-Systeme der industriellen Praxis geeignet**
- **Abstraktion unverzichtbar**
  - Informationsreduktion
  - Integration infrastrukturspezifischer Abhängigkeiten
- **Ergebnisse untermauern Entwickler-Intuition**
- **Gutes Design:**
  Kompromiss Einfachheit - Flexibilität
- **Heuristiken müssen noch erweitert werden**

Zusammenfassung

- Problemidentifikation: regelmäßige, werkzeuggestützte Untersuchung der Strukturen eines Systems
- Einzig zuverlässige Quelle: Quellcode
  Aber: Abstraktion zur Aufbereitung der Information erforderlich
- Für moderne Systeme:
  Allgemeine Heuristiken (OO, Subsysteme) + technologie und projektspezifische
- Ableiten von Restrukturierungsmaßnahmen
- Abfragetechnik auch für Verbesserung des Systemverständnisses nützlich