Teil IV: Objektorientierter Entwurf (OOD)

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

Version 15-1.0, 11.07.15

Erinnerung: UML-Aufgaben im Praktomaten

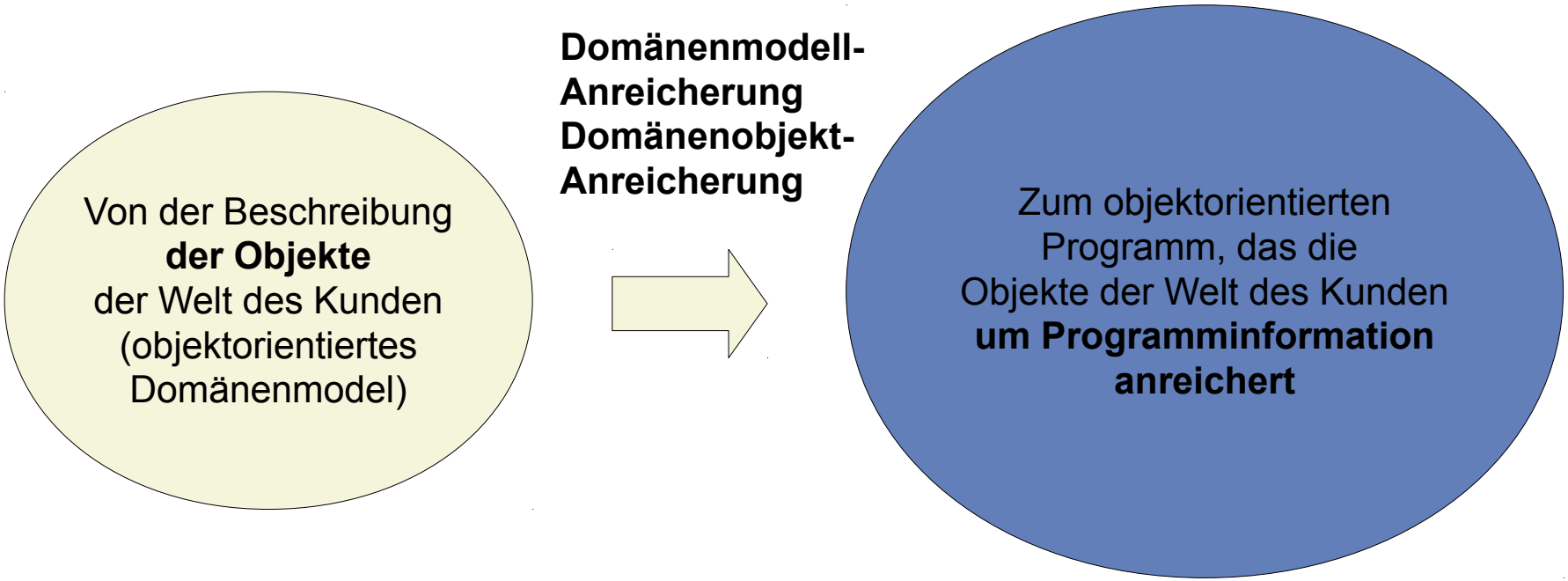
- einführende Aufgaben zur Java-Programmierung
- Aufgaben zum Übungsmaterial
- zusätzliche, komplexere Aufgaben
- Klausurrelevante Aufgaben (Implementierungsteil)

Teil IV - Objektorientierter Entwurf (Object-Oriented Design, OOD)

1) 40: Überblick
2) 41: Einführung in die objektorientierte Softwarearchitektur
   1) Architekturprinzipien, Architekturstile, Perspektivenmodelle
   2) Modularität und Geheimnisprinzip
   3) BCD-Architekturstil (3-tier architectures)
3) 42: Architektur interaktiver Systeme
4) 43: Punktweise Verfeinerung von Lebenszyklen
   - Verfeinerung von verschiedenen Steuerungsmaschinen
5) 44: Verfeinerung mit querschneidender Objektanreicherung

Die zentralen Fragen des objektorientierten Ansatzes

Wie kommen wir vom Problem des Kunden zum Programm (oder Produkt)?



Anreicherung/Verfettung: Anreicherung durch technische Programminformation
„object fattening“: Anreicherung von Objekten des Domänenmodells

Verfeinerung von UML von der Analyse zum Entwurf zur Implementierung

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung (Programm)
vollständig	nein	nein	beinahe	ja
Sprache	aUML	dUML	jUML	Java
Charakter	Fachlichkeit; Domäne	Fachlichkeit und System	Fachlichkeit, System und Technologie	Fachlichkeit, System, Technologie und Resourcen
Technologie	Annahme perfekter Technologie; funktionale Essenz	perfekt; funktionale Essenz und interne Aktivititäten	technologieabhängig (plattformabhängig)	technolgieabhängig (plattformabhängigg), resourcenabhängig
Struktur	noch wenig	Architektur des Systems	Verhalten des Systems	Lauffähiges Programm

Verfeinerung der Sprachkonzepte

	Analyse-Modell	Entwurfs-Modell (Architektur)	Implementierungsmodell (Feinentwurf)	Implementierung
Klassen	Begriffe und Domänenkonzepte; Facetten, Rollen	Systemkonzepte; Facetten, Rollen; Komponenten	Systemkonzepte nur noch in einfachen Klassen; Rollen aufgelöst durch Entwurfsmuster	Systemkonzepte
Objekte	Domänenobjekte	Systemobjekte auf Architekturebene	Alle Systemobjekte	Alle Systemobjekte
Vererbung	Begriffshierarchien Mehrfachvererbung, Produktverbände	ad-hoc Vererbung	konform, 1-D-Vererbungshierarchie	konform
Assoziationen	oft ungerichtet	mit Kardinalitäten	abgeflacht	

Verfeinerungsschritte beim Übergang vom Analyse- zum Entwurfsmodell

- **Vervollständigung** fehlender Angaben
  - Typisierung
  - Ausarbeitung von Objekt-Lebenszyklen (punktweise Verfeinerung)
  - Ausarbeitung von Kollaborationen (querschneidende Verfeinerung)
- **Detaillierung**
  - Querschneidende Objektanreicherung durch Kollaborationen
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen
    - Einziehen von Qualifikationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Einziehen von Schnittstellen zur Sicherstellung von homogenem Verhalten
  - Einziehen von Materialbehälterklassen (Verwaltungsklassen)

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (II)

- **Strukturierung** und Restrukturierung
  - Refactoring (Restrukturierung unter Beibehaltung der Semantik)
  - Erhöhung von Wiederverwendung:
    - Einziehen von Komponenten mit angebotenen und benötigten Schnittstellen
    - Einziehen von Paketen zur Strukturierung
    - Einschränkung durch Sichtbarkeiten zur loseren Kopplung (Datenkapselung und Austauschbarkeit)
  - Identifikation von abgeleiteten Modellelementen zur Elimination von Redundanz (Verschlankung)
- **Flachklopfen** (Abflachen, Realisierung)
  - Integration von Unterobjekten (Rollen, Facetten, Teile) mit Hilfe von Entwurfsmustern: Compositum, Brücke, u.v.m.
  - Realisierung von Objektnetzen: Abflachen der Assoziationen
- **Erhöhung Zuverlässigkeit**
  - Verfeinerung der Vererbungsrelation zu konformer Vererbung zur Elimination von Fehlern
  - Schreiben von Verträgen mit Vor- und Nachbedingungen

Verfeinerungsschritte beim Übergang zum Entwurfsmodell (III)

- Siehe Kapitel 41
- Entwickeln des **Architekturaspektes**
  - Entwurfsmuster anwenden
  - Schichtung
  - Architekturprinzipien wie Geheimnisprinzip oder lose Kopplung
- Erhöhe **Wiederverwendbarkeit**
  - Verwende **Variabilitätsmuster**
  - Verwende **Erweiterungsmuster**
  - Integriere externe Komponenten durch **Klebemuster**
  - Ausnutzen des **Quasar**-Prinzips zur Identifikation von wiederverwendbaren Einheiten

Schritte beim Übergang zum Implementierungsmodell (Feinentwurf)

- Vervollständigung fehlender Angaben
  - Implementierung von Methoden
- Strukturierung und Restrukturierung
  - Auflösen von Mehrfachvererbung in Aggregation
  - Einziehen von abgeleiteten Relationen, um Zugriffe, Navigationen und Abfragen zu beschleunigen (Verfettung)
- Detaillierung (Implementierungsmuster anwenden)
  - Assoziationen:
    - Einziehen von Navigationsrichtungen, um Platz zu sparen; Ersetzen durch Suchalgorithmen
    - Abbildung von qualifizierten Assoziationen, um Suchen zu beschleunigen
    - Annotation von geordneten und sortierten Assoziationen
  - Verfeinerung der Vererbungsrelation zu *konformer Vererbung* zur Elimination von Fehlern
  - Transformation in die Implementierungssprache
- Verhalten angeben
  - Lebenszyklen modellieren
  - Implementierung von Schnittstellen auswählen

Schritte beim Übergang zum Code

- **Codegenerierung**
  - Nutzung von Codegenerator-Werkzeugen, um Lebenszyklen in Programm zu übersetzen
  - Nutzung von Webe-Werkzeugen, um querschneidende Kollaborationen in Klassen einzuweben
  - Alternativ: Umsetzung der Modelle in Code von Hand
- Hand-Implementierung fehlender Komponenten

40.2. Herstellung Großer Softwaresysteme

software: computer programs, procedures, rules, and possibly associated documentation and data pertaining to the operation of a computer system.

(IEEE Standard Glossary of Software Engineering)

Was heißt hier "groß"?

- Klassifikation nach W. Hesse:

Klasse	Anzahl Code-Zeilen	Personenjahre zur Entwicklung
sehr klein	bis 1.000	bis 0,2
klein	1.000 - 10.000	0,2 - 2
mittel	10.000 - 100.000	2 - 20
groß	100.000 - 1 Mio.	20 - 200
sehr groß	über 1 Mio.	über 200

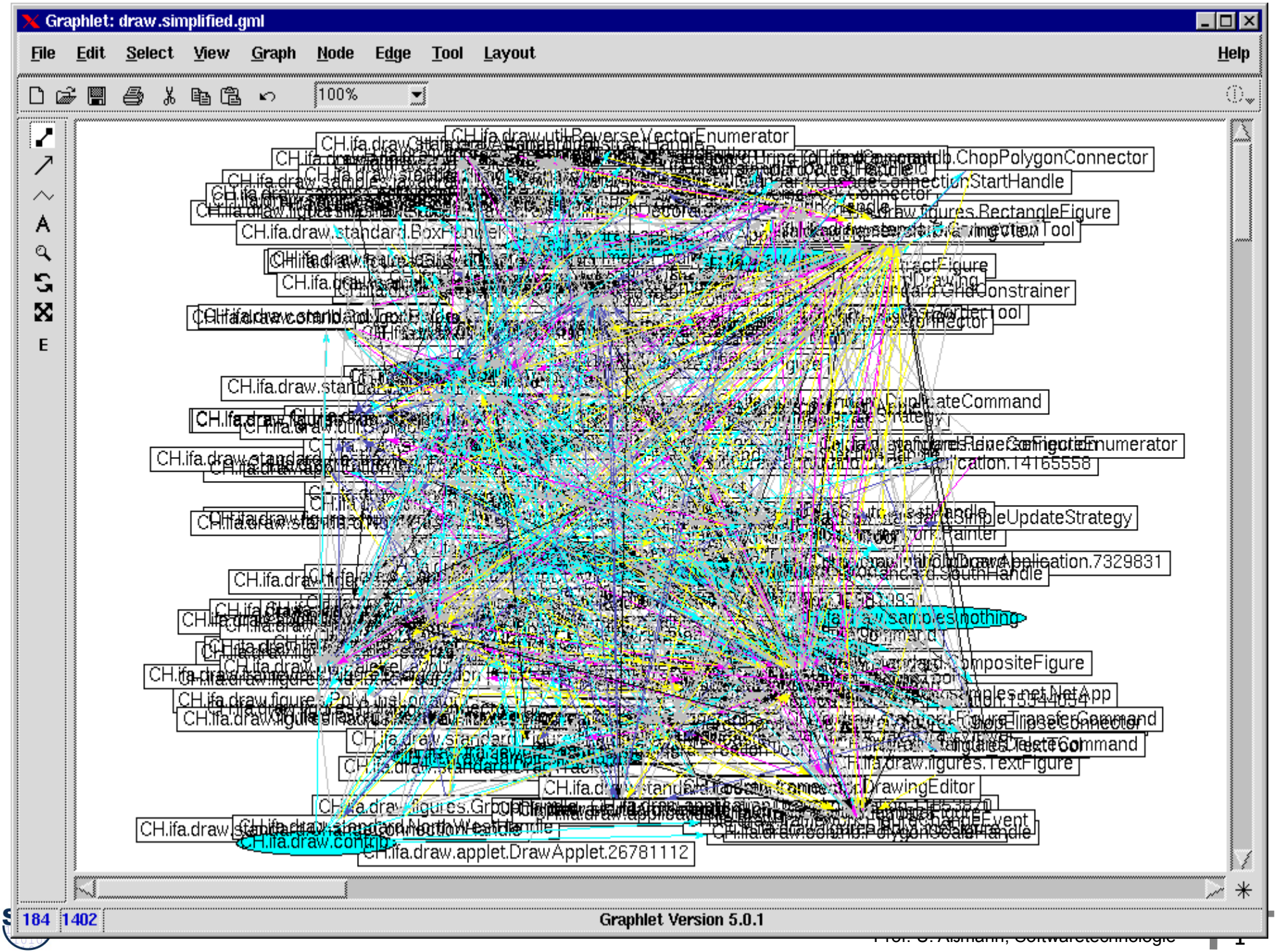
Riesige Systeme

- Telefonvermittlungssoftware EWSD Siemens (Version 8.1):
  - 12,5 Mio. Code-Zeilen
  - ca. 6000 Personenjahre
- ERP-Software SAP R/3 (Version 4.0)
  - ca. 50 Mio. Code-Zeilen
- Umfang der verwendeten Software (Anfang 2000):
  - Credit Suisse 25 Mio. Code-Zeilen
  - Citicorp Bank: 400 Mio. Code-Zeilen
  - AT&T: 500 Mio. Code-Zeilen
  - General Motors: 2 Mrd. Code-Zeilen
- HEUTE:
  - Mercedes Benz S-Klasse 9 Mio LOC
  - Credit Suisse > 100 Mio LOC

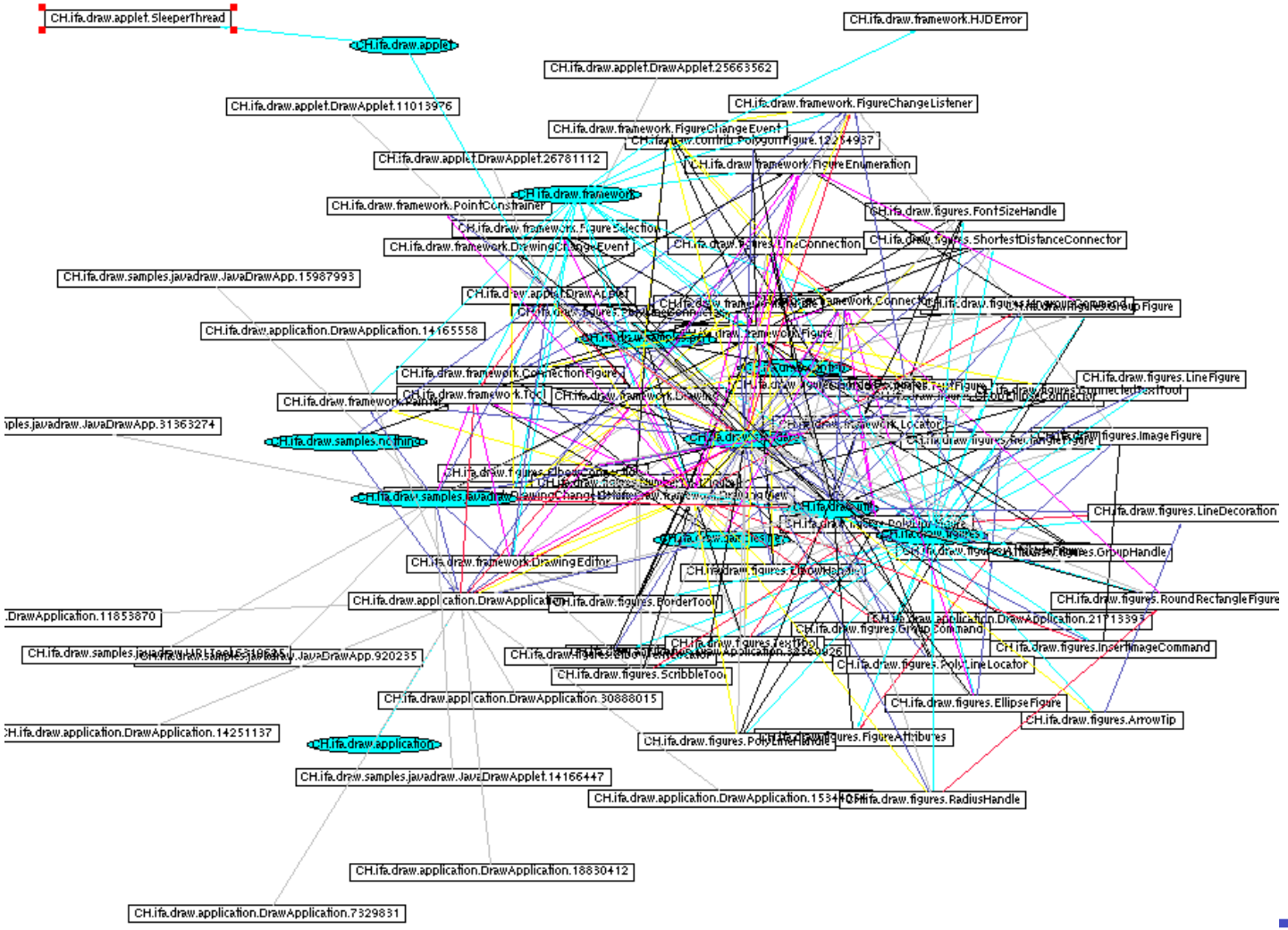
Abkürzungen:
EWSD = Elektronisches Wählsystem Digital (Siemens-Produkt)
ERP = Enterprise Resource Planning
SAP: Deutscher Software-Konzern

Strukturprobleme

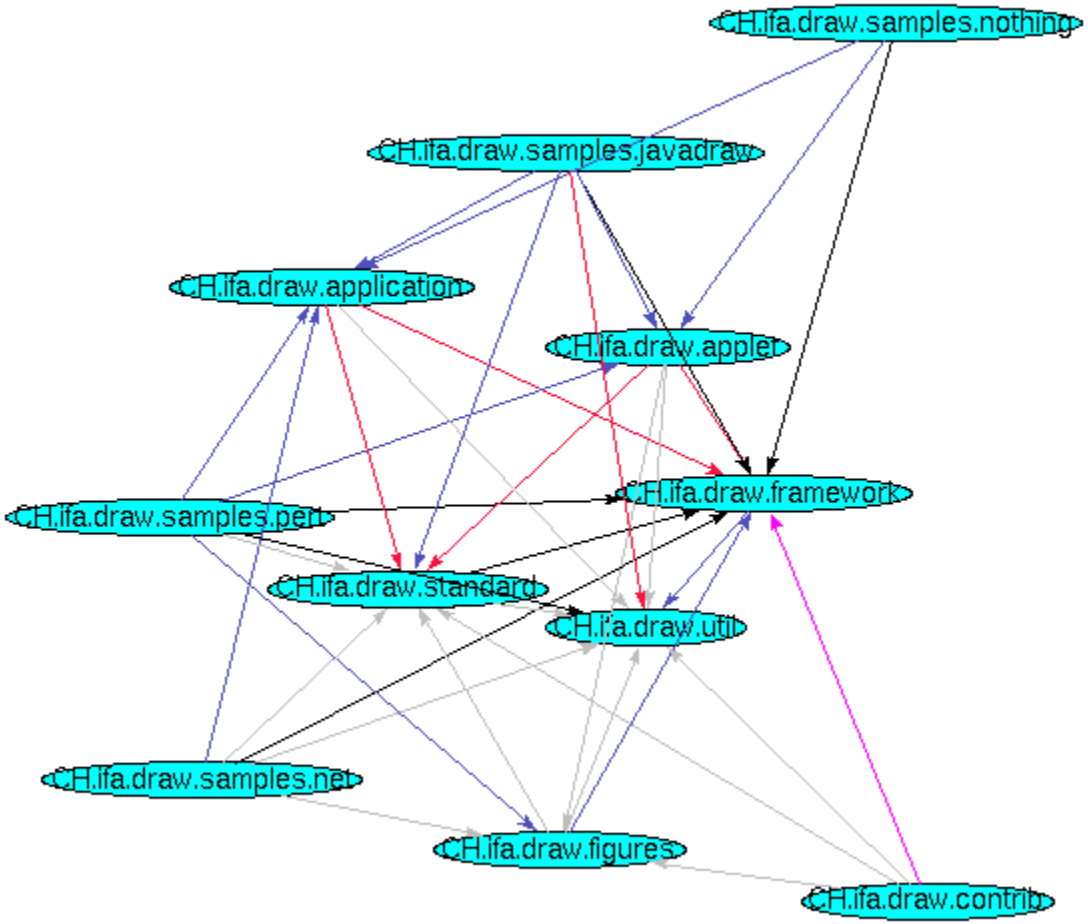
- Folgende Bilder stammen vom Goose Reengineering-Werkzeug aus der Analyse eines Java-Systems [Goose, FZI Karlsruhe, http://www.fzi.de]


Graphlet: draw.simplified.gml
File
Edit
Select
View
Graph
Node
Edge
Tool
Layout
Help
100%
CH.ifa.draw.util.ReverseVectorEnumerator
CH.ifa.draw.figures.RectangleFigure
CH.ifa.draw.standard.BoxHandleKit
CH.ifa.draw.figures.TextFigure
CH.ifa.draw.DrawingEditor
CH.ifa.draw.applet.DrawApplet.26781112
184
1402
Graphlet Version 5.0.1

Teilweise gefaltet



Gefaltet


CH.ifa.draw.samples.nothing
CH.ifa.draw.samples.javadraw
CH.ifa.draw.application
CH.ifa.draw.applet
CH.ifa.draw.framework
CH.ifa.draw.samples.pert
CH.ifa.draw.standard
CH.ifa.draw.util
CH.ifa.draw.samples.net
CH.ifa.draw.figures
CH.ifa.draw.contrib

The Engineer's Ring (Canada)

- Der Ring der kanadischen Ingenieure
  - Pont de Québec
  - Der Schwur des kanadischen Ingenieurs
- **Ingenieure** lösen Probleme von Menschen, basierend auf den Naturgesetzen, aber auch den Gesetzen und Methoden des Entwurfs, der Konstruktion, der Produktion, der Modellierung
- **Ingenieurswissenschaftler** entwickeln neue Gesetze und Methoden
- Software Engineering ist eine Ingenieurs- oder Technik-Wissenschaft (“science of engineering”)

"Gold is for the mistress - silver for the maid!
Copper for the craftsman cunning at his trade."
"Good!" said the Baron, sitting in his hall.
"But Iron, Cold Iron - is master of them all!"

Rudyard Kipling

Softwarearchitektur

- Softwarearchitektur ist der Schlüssel zum Erfolg des Softwareingenieurs und seiner Firma.

Ohne gute Softwarearchitektur keine Zuverlässigkeit, Einbruchssicherheit, Wiederverwendung, Evolution, Variabilität, Erweiterbarkeit

Mit guter Softwarearchitektur Softwareproduktlinien, schnell erstellte neue Produkte, vertikale Portierung auf andere Domänen, einfaches Dienstleistungsgeschäft.

The End