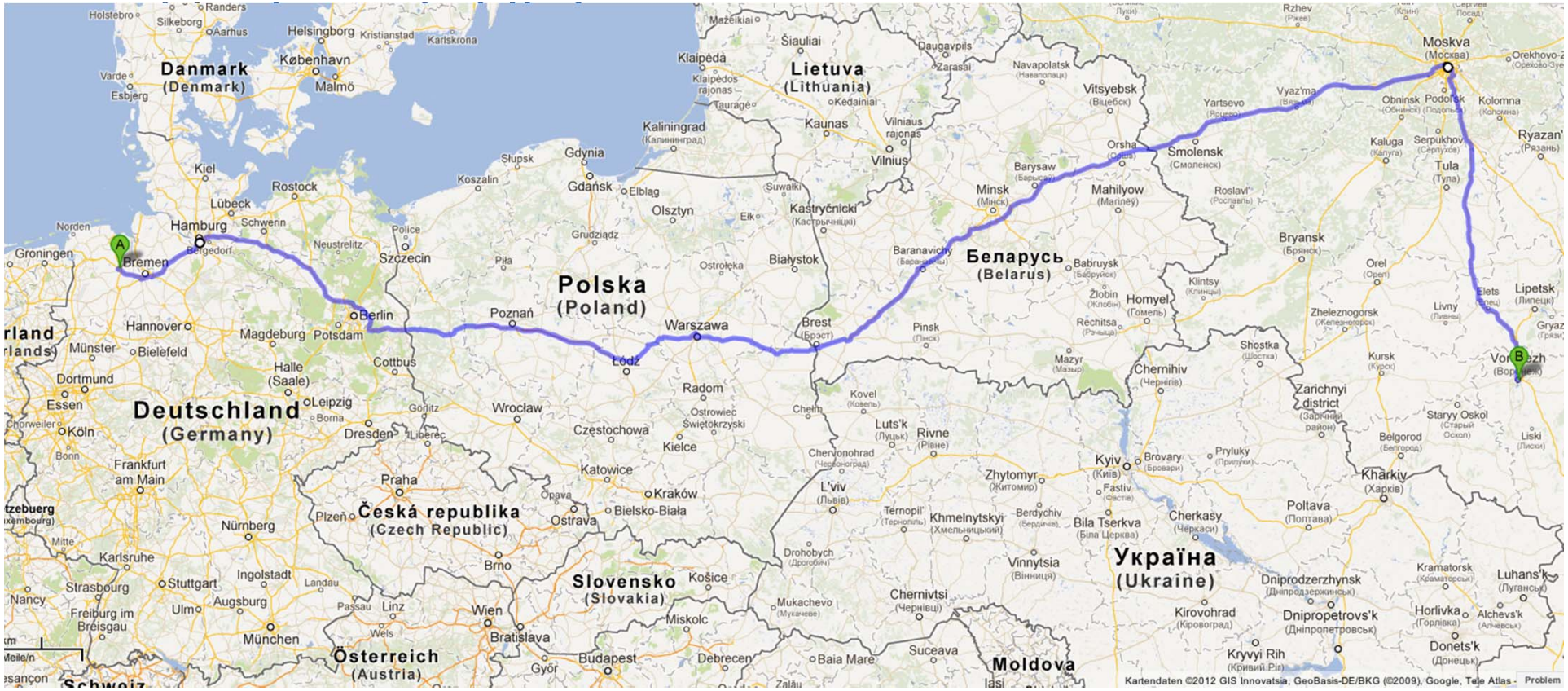
CARL VON OSSIETZKY universität OLDENBURG

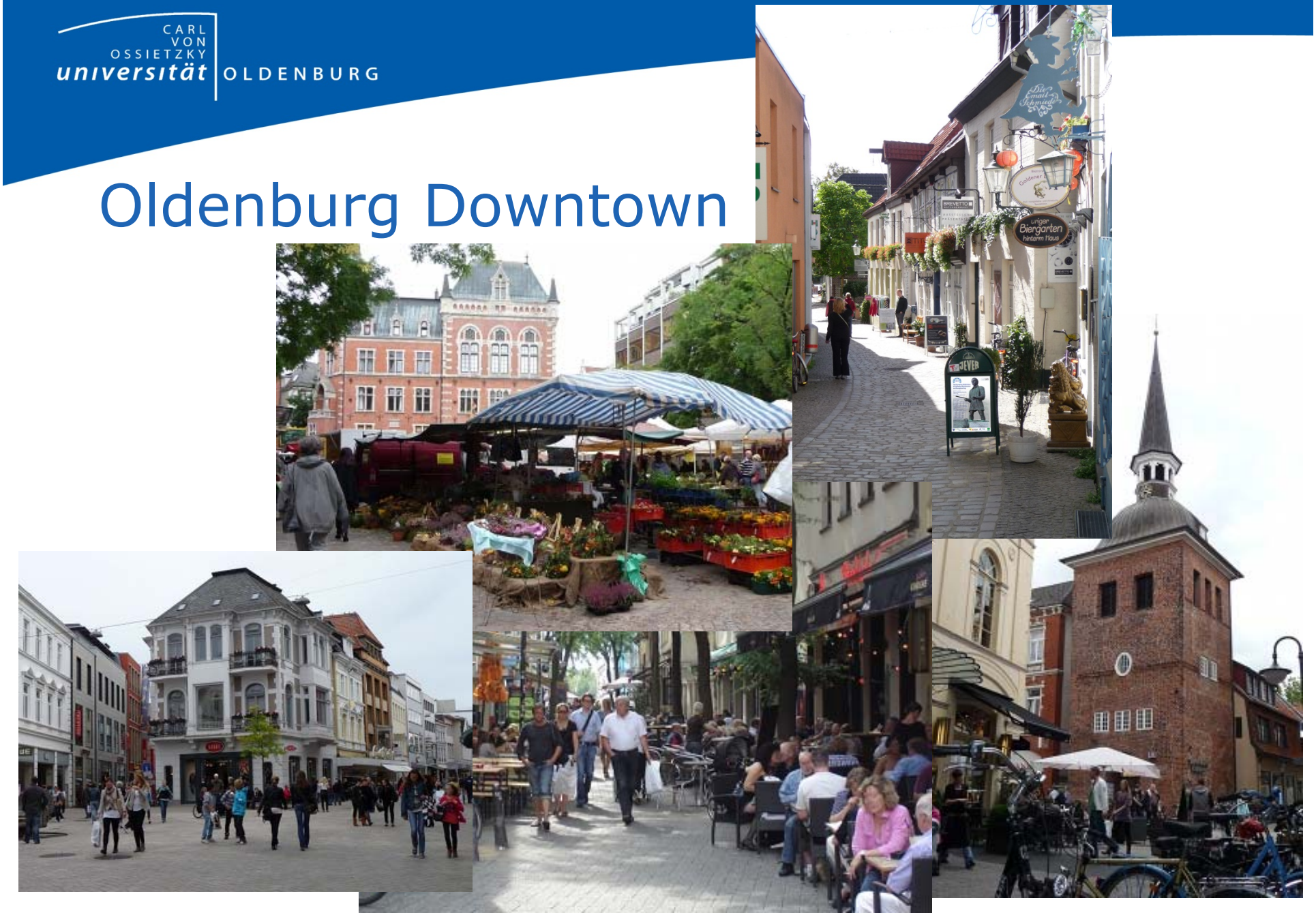
Studying Computer Science in Oldenburg (Oldb.), Germany

Воронеж - Oldenburg



… some Impressions from Oldenburg

Oldenburg Downtown



CARL VON OSSIETZKY universität OLDENBURG

Oldenburg Waters

Oldenburg Gardens

Oldenburg Bikes

Carl von Ossietzky University

Statistics

Students: 11325

- Female: 6354
- Male: 4971

Professors: 182

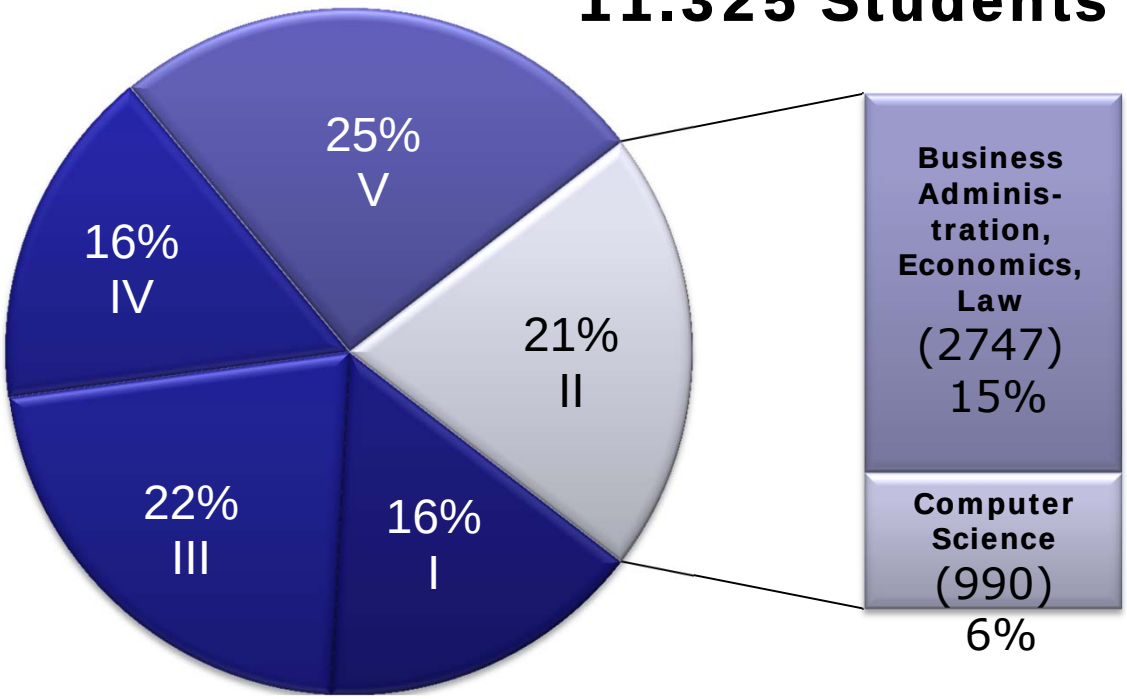
- Female: 57
- Male: 125

Researcher: 999

- Female: 436
- Male: 563

Faculties

11.325 Students



I. School of Education
II. School of Computing Science, Business Administration, Economics and Law
III. School of Linguistics and Cultural Studies
IV. School of Humanities and Social Sciences
V. Faculty of Mathematics and Science

CARL VON OSSIETZKY universität OLDENBURG

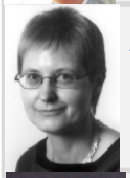
Department for Computer Science

CARL VON OSSIETZKY universität OLDENBURG

Statistics

Students	990
o Bachelor	> 620
o Master	> 180
o Diploma	> 80
o PhD	> 50
Professors	21
Researcher	> 50

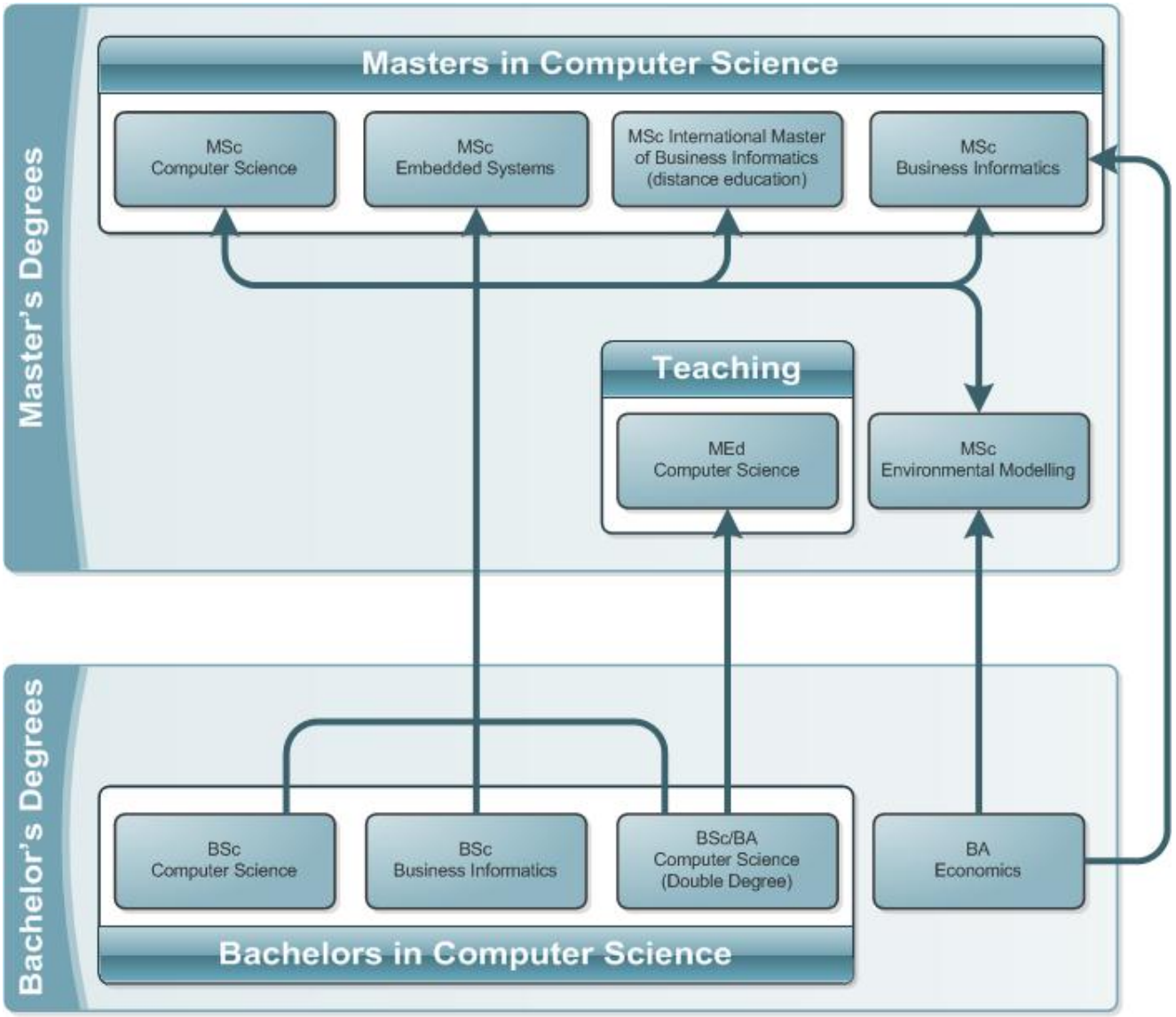
Professors and Groups

Theoretical CS	Practical CS	Applied CS	Technical CS
 **Eike Best** Parallel Systems	 **H.J. Appelrath** Information-systems	 **Ira Diethelm** Didactics in Computer Science	 **Werner Damm** Safety Critical Embedded Systems
 **Annegret Habel** Formal Languages	 **Susanne Boll** Multimedia Systems	 **Axel Hahn** Business Informatics	 **Sergej Fatikow** Micro-Robotics & Control Engineering
 **E.R. Olderog** Correct Systems	 **W. Kowalk** Computer Networks	 **J. Marx Gomez** Business Informatics (VLBA)	 **Martin Fränzle** Hybrid Systems
	 **Oliver Theel** Distributed Systems	 **M.Sonnenschein** Environmental Informatics	 **Andreas Hein** Integrated Systems and Microsensors
	 **Andreas Winter** Software Engineering	 **Oliver Kramer** Computational Inteligence	 **Wolfgang Nebel** Embedded HW/SWSystems
	 **Daniela Nicklas** Database & Inter-net Technologies	 **S. Lehnhoff** Energy Informatics	 **Achim Rettberg** Integrated Embedded Systems

Teaching Programs

- Bachelor
- Master

- PhD
  - research assistants (CvO, OFFIS)
  - PhD Grants
    - e.g. Erasmus Mundus, TARGET
  - Post-Graduate Program
    - DFG SCARE



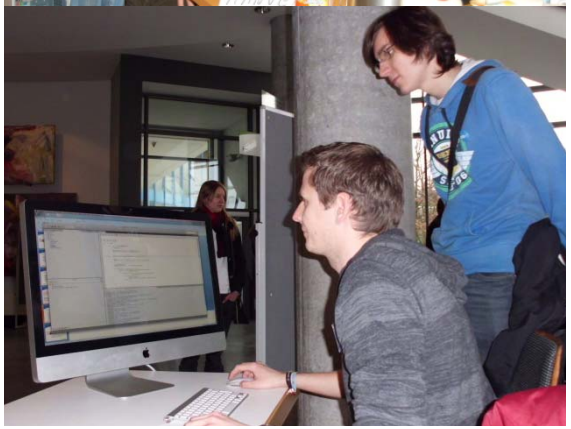
Areas of Specializations

Bachelor

- Environmental Informatics
- Modeling and Analysis of Complex Systems
- Embedded Systems and Micro-Robotics
- System Software
- Information Systems and Software Engineering
- Computer Science in Education

Master

- Environmental Informatics
- Modeling and Analysis of Complex Systems
- Computer Science in Education
- IT in Power Industry
- IT in Health
- Complex Information and Software Systems
- Reliable Systems



Curriculum: Bachelor Informatics

1.	Algorithms and Programming	Programming in Java	Technical Computer Science 1	Discrete Structures	Linear Algebra
2.	Algorithms and Data Structures	Soft Skills	Technical Computer Science 2	Theoretical Computer Science 1	Analysis
3.	Information-systems 1	Software Engineering 1	*Choice*	Theoretical Computer Science 2	*Specialization in Mathematics*
4.	Operating Systems	Project and Proseminar	Technical CS Practical Training	Computer Networks	*Choice*
5.	Computer Science and Society		*Choice*	*Choice*	*Choice*
6.	Bachelor Thesis		Seminar	*Choice*	*Choice*

Curriculum: Master Informatics - Specialization KISS

1.	Early Phases	Information-Systems	Choice	Choice	Applications
2.	Project (24 KP)		Seminar	Choice	non CS
3.			Applications	Choice	non CS
4.	Master Thesis (30 KP)				

Applications

- Mobile Systems, HCI, Health Care Information-Systems, Special Topics Software Engineering, Special Topics Information Systems, Hybrid Systems, Knowledge Management, Secure Communication, Intelligent Systems, ...

non CS

- law, data protection law, commercial law, legal informatics, ...

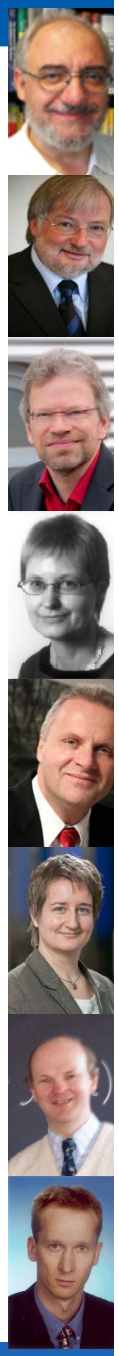
SCARE (System Correctness under Adverse Conditions)

Objective

- SCARE addresses system correctness of hard- and software systems to guarantee robustness of the system behavior under adverse conditions. Correctness focusses on satisfying a given specification of desired cooperation properties between environment and systems
- System correctness under adverse conditions refers to
  - limited knowledge
  - unpredictable behavior
  - changing system environment and system structure

Research Topics

- Modeling Techniques
- Verification and Analysis Techniques
- Constructive Techniques (combination of formal methods with engineering approaches)



Research Training Group SCARE

Application

- 15 Research Positions for PhD Candidates
  - Five positions: 1 October 2012
  - Five positions: 1 October 2013
  - Five positions: 1 October 2014
- Funding:
  - Three years by DFG, TV-L E13
- Prerequisite
  - excellent Master's degree (or equivalent) in Computer Science
- apply to (deadline 1st group: June 30, 2012)
  - Prof. Dr. Ernst-Rüdiger Olderog, Department of Computing Science, FK II, University of Oldenburg, 26111 Oldenburg, Germany (scare@uni-oldenburg.de)

more information
http://scare.uni-oldenburg.de/

Research

Research Foci

applied computer science in strong cooperation with OFFIS

- Safety Critical Embedded Systems
- Energy Efficiency in Information Technologies

Research Perspective

exploring future research foci

- ExploIT Dynamics

more information
http://fzsks.uni-oldenburg.de/

Safety Critical Embedded Systems

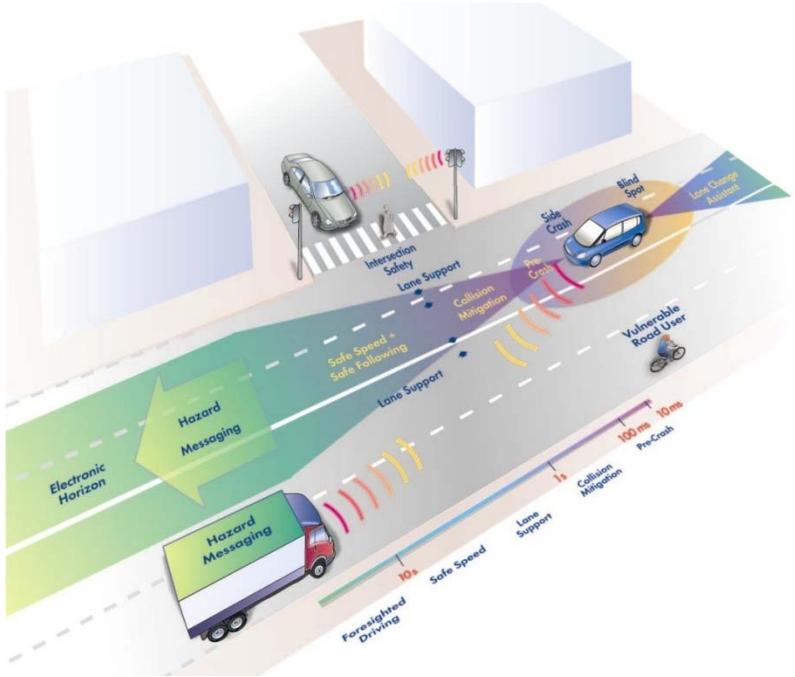
Objective

- aims at improving productivity and quality in developing digital, embedded hard- and software-systems
  - car driving safety technologies
  - computerized control systems in avionics

Center of Excellence

Deutsche Forschungsgemeinschaft DFG

- AVACS (Automatic Verification and Analysis of Complex Systems)
  - automatic verification of hard- and software systems used in safety critical embedding e.g.
    - Avionics
    - Transportation by car
    - Transportation by train



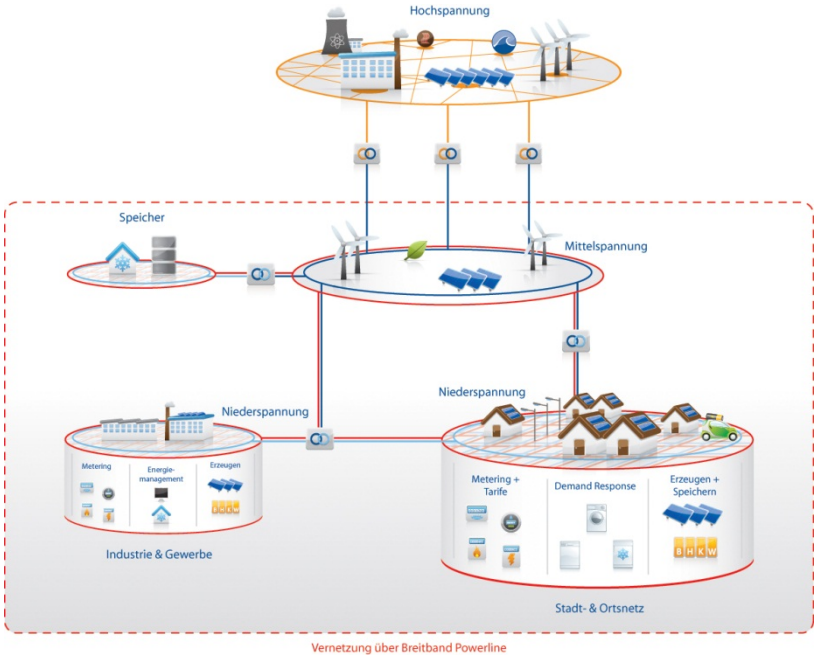
Energy Efficiency in Information Technologies

Objective

- Acheiving climate protection goals requires a rethinking of producing and using energy. Energy Efficiency in Information Technologies aims at improving energy efficient behavior by applying Information technologies

Topics

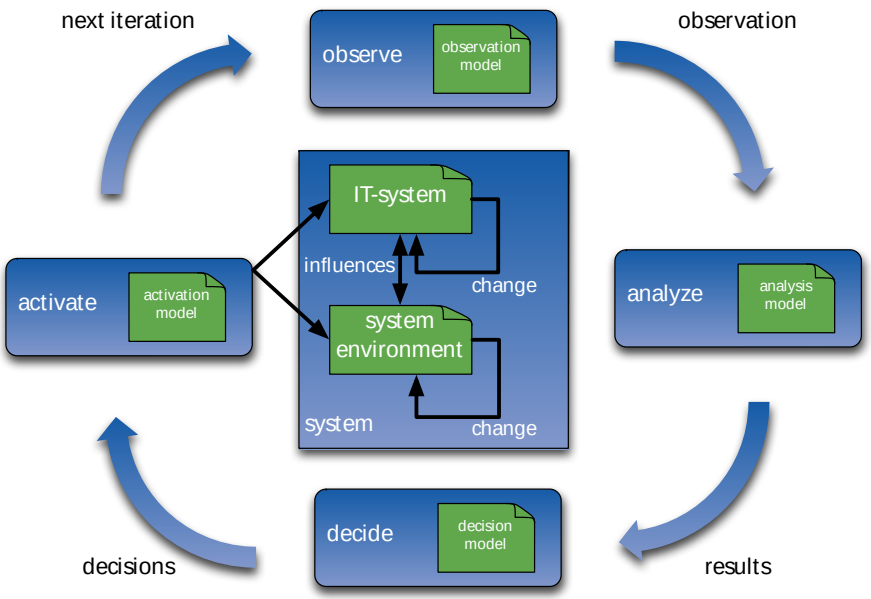
- Smart Grids
  - ensuring future power supply environmentally friendly and economically
- IT2Green
  - ensuring energy efficiency of production and transportations processes
- GreenIT
  - ensuring energy efficiency of information technologies (e.g. in data centers; software applications)



ExploIT Dynamics

Objective

- improve and maintain the quality of systems in a changing world
- develop and apply (*domain independent*) techniques to control and make use of the dynamic behavior of systems



Topics

- Software Evolution
  - provide model-based techniques to statically and dynamically analyze software systems and perform automatic reconfigurations (RQ, BI)
- Quality Monitoring
  - provide techniques to constantly monitor and maintain systems quality (e.g. during software migration and systems operation)
- Highly Reliable Data and Services
  - provide replication and prediction techniques to improve efficient and reliable data access or usage of services

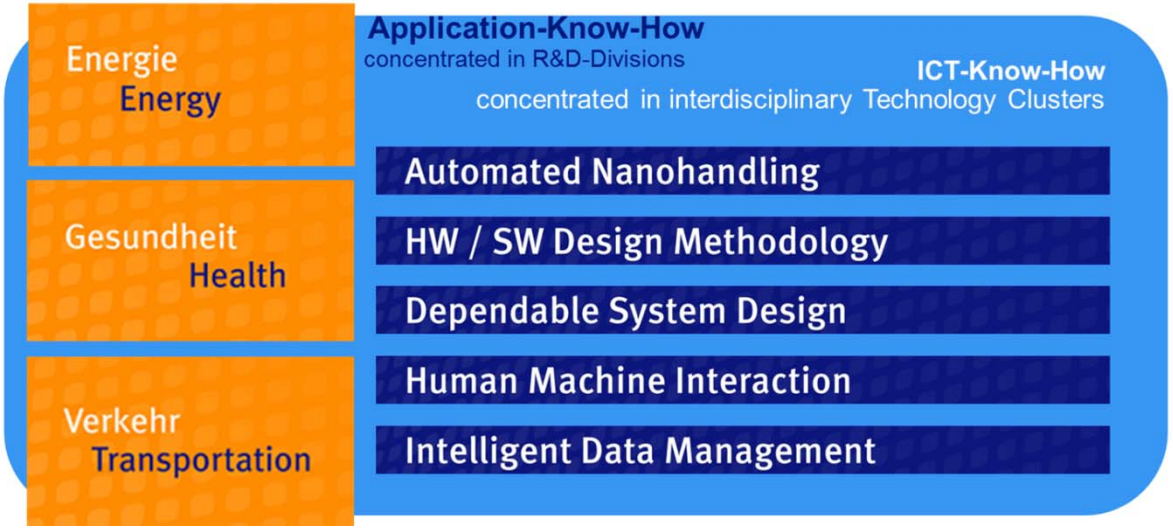
OFFIS

Institute for Information Technology

OFFIS

Oldenburg Research Institute on Applied Computer Science

- founded in 1991
- affiliated to Department of Informatics of CvO
- more than 290 researcher
- research profile



About OFFIS

Mission:

- Support of innovation through technology transfer
- Strengthening of the IT location Oldenburg
- Advancement the Metropolitan Region Northwest

Members:

- State of Lower Saxony and University Oldenburg
- 28 Professors of IT and related studies of University Oldenburg and Jade University

Budget:

- Income in 2011: 13,09 million €
- Basic funding from the state of Lower Saxony approx. 26%
- Third party funding from international, national and regional projects approx. 74%

Performance:

- More than 400 cooperation partners regionally/nationally/internationally
- More than 300 R&D projects only since 2001 carried out
- European-wide interlinking science/economy/politics
- Scores of spin-offs, participation in development of international standards

coming soon

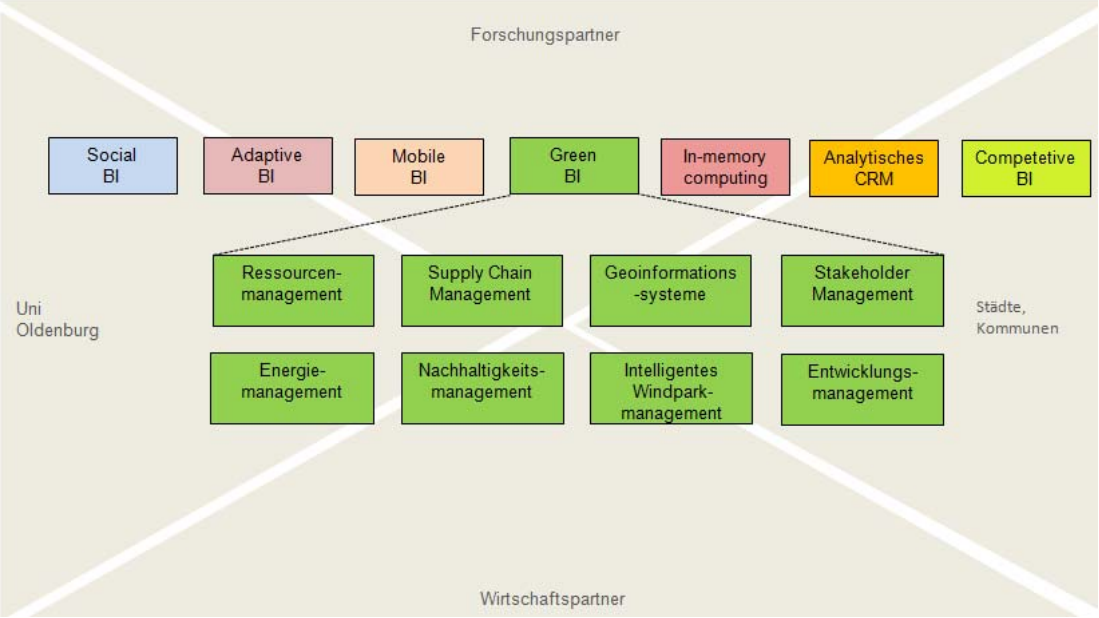
Institute for Business Intelligence

Business Intelligence

- using all business information, available in various formats to support strategic decision finding in business
- requires intensive research on data extracting and merging

uses

- data ware-houses
- static and dynamic analysis techniques



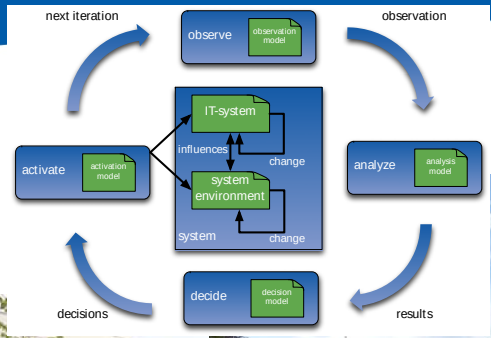
Contact

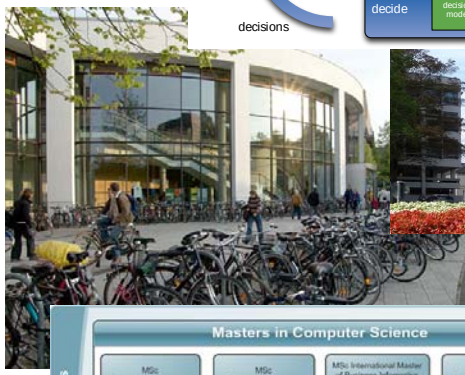
- Jorge Marx Gomez
  jorge.marx.gomez
  @uni-oldenburg.de

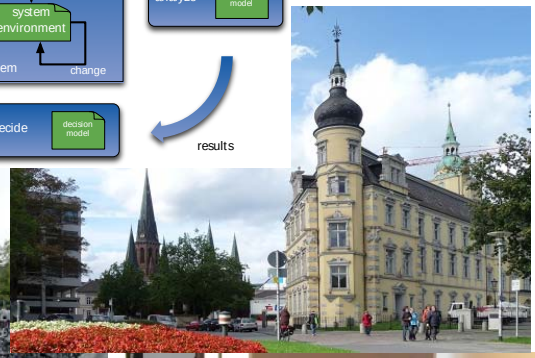
Summary

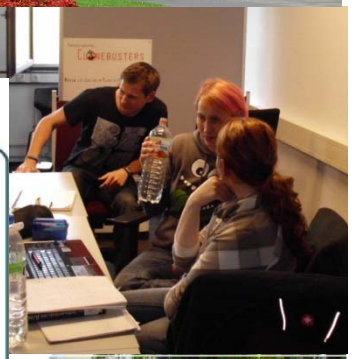
Oldenburg offers a wide variety of chances

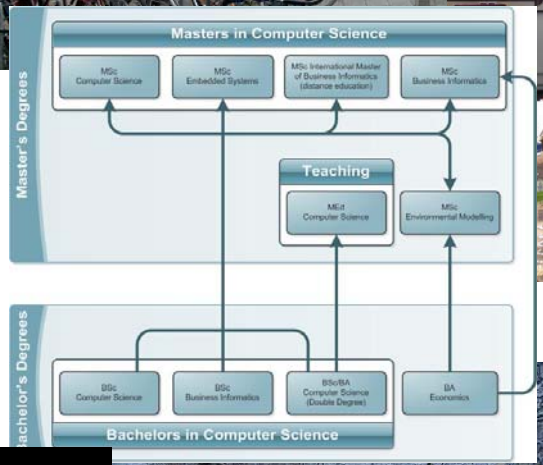
- to deepen your studies in computer science
- to do research on computer science foundations and their applications
- to stay in a worth living environment

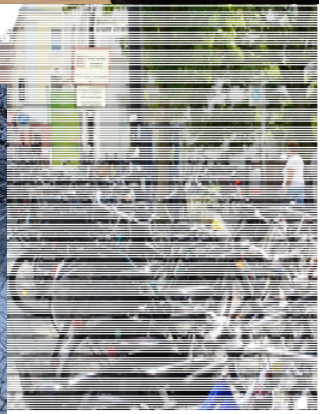


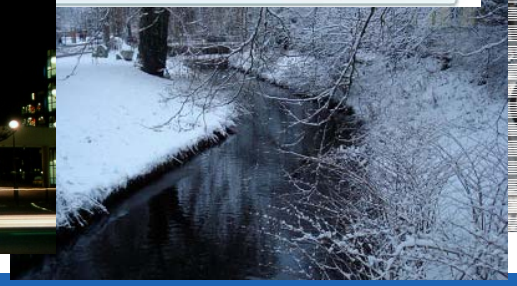












CARL VON OSSIETZKY universität OLDENBURG

Using Graph-Technology to Improve Software-Evolution

Andreas Winter

CARL VON OSSIETZKY universität OLDENBURG

Oldenburg
Software Engineering Group

Software Engineering Group

Head

- Andreas Winter

Secretary

- Marion Bramkamp

PhD Students

- Jan Jelschen
- Maxat Kulmanov
- Dilshodbek Kuryazov
- Yvette Teiken (OFFIS)

Student Assistants

- Marion Gottschalk
- Mirco Josefiok

Software Engineering Group

Topics in Research and Teaching

- Software-Engineering
- Modeling and Metamodeling
- Graph-Technology
  - Graph based modeling and implementation
- Process-Models in Software Development
- Software Evolution

Mission

- Development and Application of Graph-Technology to improve Software Evolution

Outline

http://www.thereelbits.com/2011/02/22/the-way-back/

Foundations

- Software-Engineering
- Software-Evolution

Graph-Technology

- Graph-based Modeling
- Graph-Querying

Current Activities

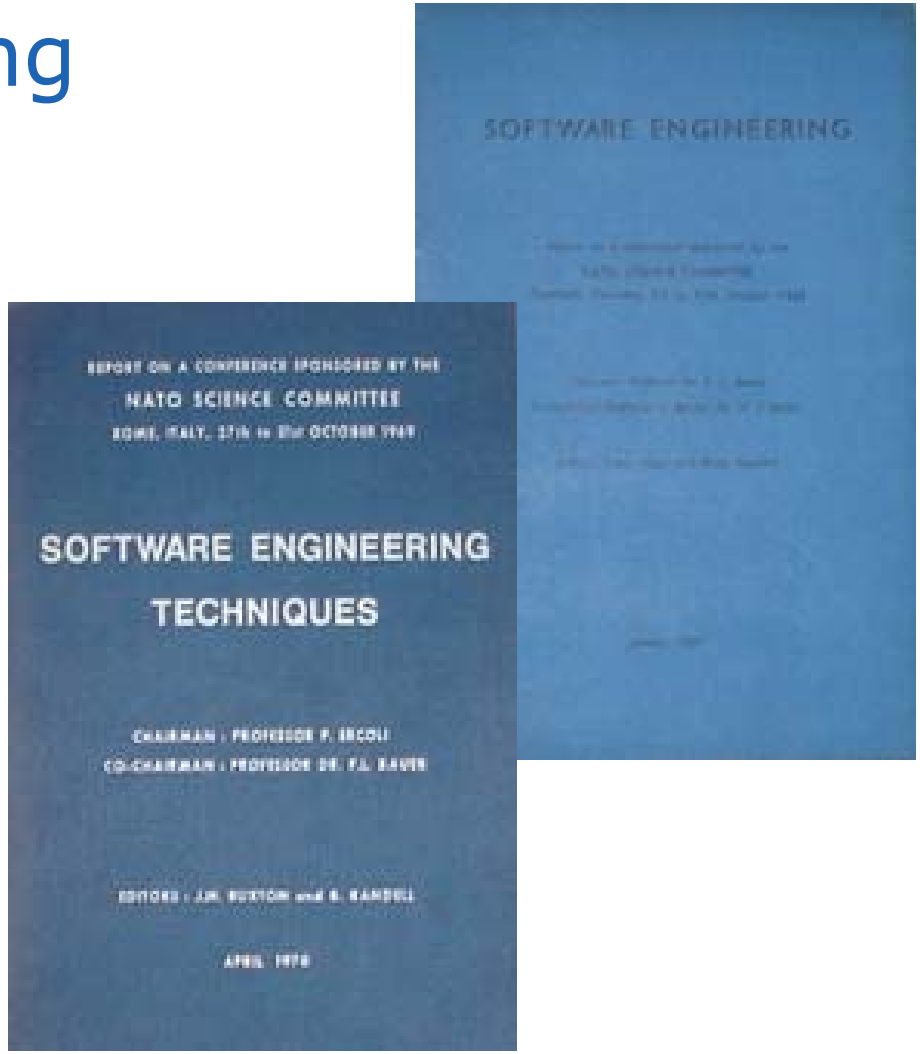
- SOAMIG: Migrating Legacy Software to Service-Oriented Architectures
- SES: Software-Evolution Services
- EEA: Removing Energy Code Smells with Reengineering Services
- Modeling Deltas: Version Control for Software-Models

Software Engineering and Software Evolution

Software Engineering

Software Crisis

- Software development in the sixties
  - Increase of software complexity
  - missing suitable programming languages
  - missing suitable methods and techniques for engineering software systems
  - no mail, internet, Java, .net, eclipse, Google, sourceforge, twitter, facebook, ...



[http://homepages.cs.ncl.ac.uk/brian.randell/NATO/]

Software Engineering

[F. L. Bauer]

- *"[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."*
  (Software Engineering, Garmisch, October 7-11, 1968)

[IEEE Std. 601.12-1990, 1993]

- Software Engineering:
  (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  (2) The study of approaches as in (1).

Software Engineering

http://www.mobileslead-just-needs-high-efficiency/

Engineering

- follows established principles
- applies methods and techniques purposefully
- looks for technically and costly efficient solutions
- rejects blindly and imprudently ad hoc problem solving

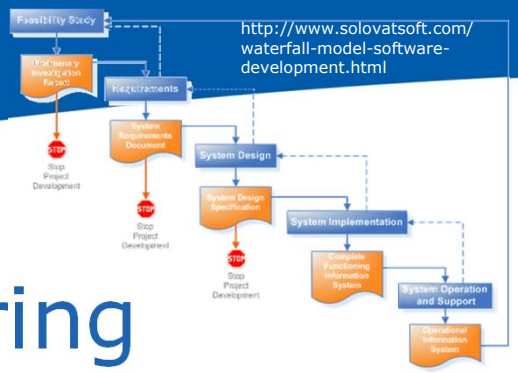
Software Engineering

- elicits and clearly defined system requirements
- constructs (models) alternative solutions (software architecture)
- evaluates solutions
- realizes solutions (i.e. programming)
- reviews solutions according their requirements

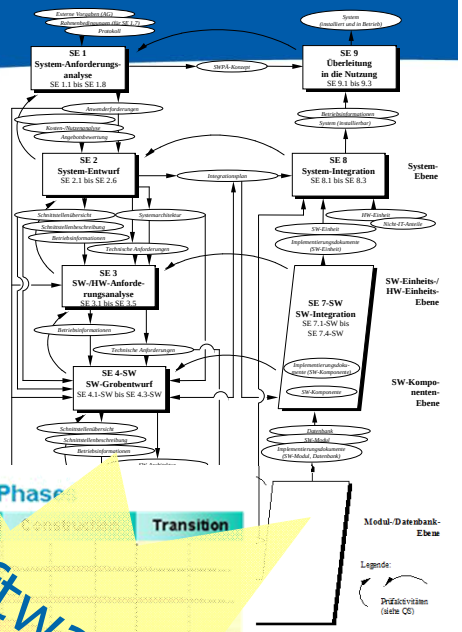
Conclusion

- **`software engineering != programming`**

Software Engineering



http://www.solovatsoft.com/waterfall-model-software-development.html



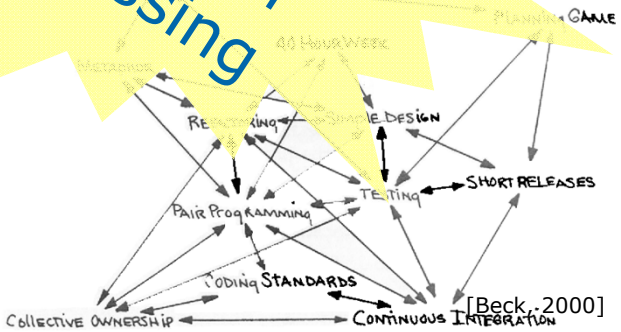
[SWB IT, 1997]

Software development activities

- plan and organize projects
- elicit requirements
- define software architecture
- construct software systems
- test software systems
- run software systems

[Kruchten, 2001]

Software Evolution is missing

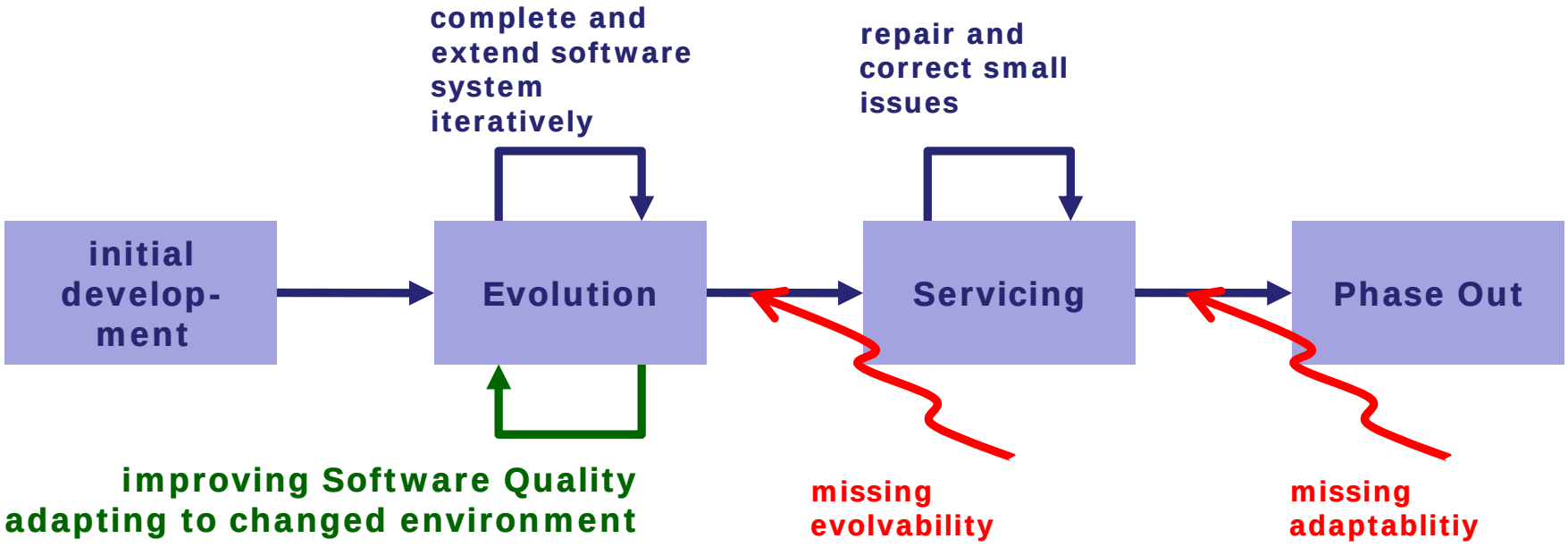


[Beck, 2000]

Software Evolution Life Cycle

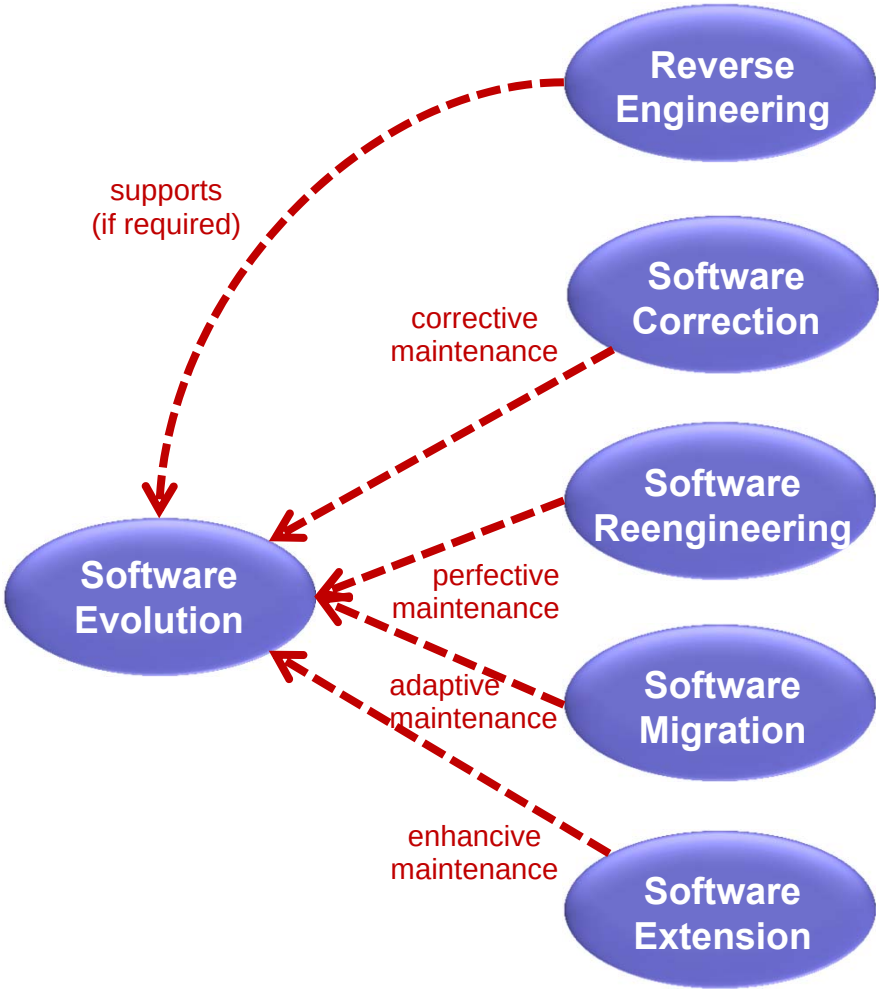
Software Evolution

- covers all activities to keep an existing software system running in its changing environment



[Rajlich/Bennett, 2000]

Activities in Software Evolution



Extracting a more abstract system description

software ⟶ documentation

eliminating software errors

software ⟶ more (?) correct software

improving software quality (not changing functionality)

software ⟶ better (?) software

transferring software to new environment (not changing functionality)

software ⟶ software in environment

extending software

software ⟶ software with new or changed functionality

Graph Technology

Graph Technology

wanted:

- powerful means to represent and analyze
  - program code
  - software models

TGraphs

- directed graphs with
- typed nodes and edges
- attributed notes and edges
- ordered node, edge, and incidence sets

A $TGraph\ G = (Vseq, Eseq, \Lambda seq, type, value)$ consists of

- an injective sequence $Vseq$ of a set $V \subseteq VERTEX$ of *vertices*
- an injective sequence $Eseq$ of a set $E \subseteq EDGE$ of *edges*
- a total *incidence function* $\Lambda seq : V \rightarrow \mathrm{seq}(E \times \{in, out\})$ such that for each $e \in E$
  there is exactly one vertex v with $(e, out) \in \Lambda seq(v)$
  and exactly one vertex w with $(e, in) \in \Lambda seq(w)$
- a total *type function* $type : V \cup E \rightarrow TYPE$
- a total *value function* $value : V \cup E \rightarrow (ATTR \nrightarrow VALUE)$

TGraph-Classes

- grUML-Schemas providing conceptual modeling of Graph structures

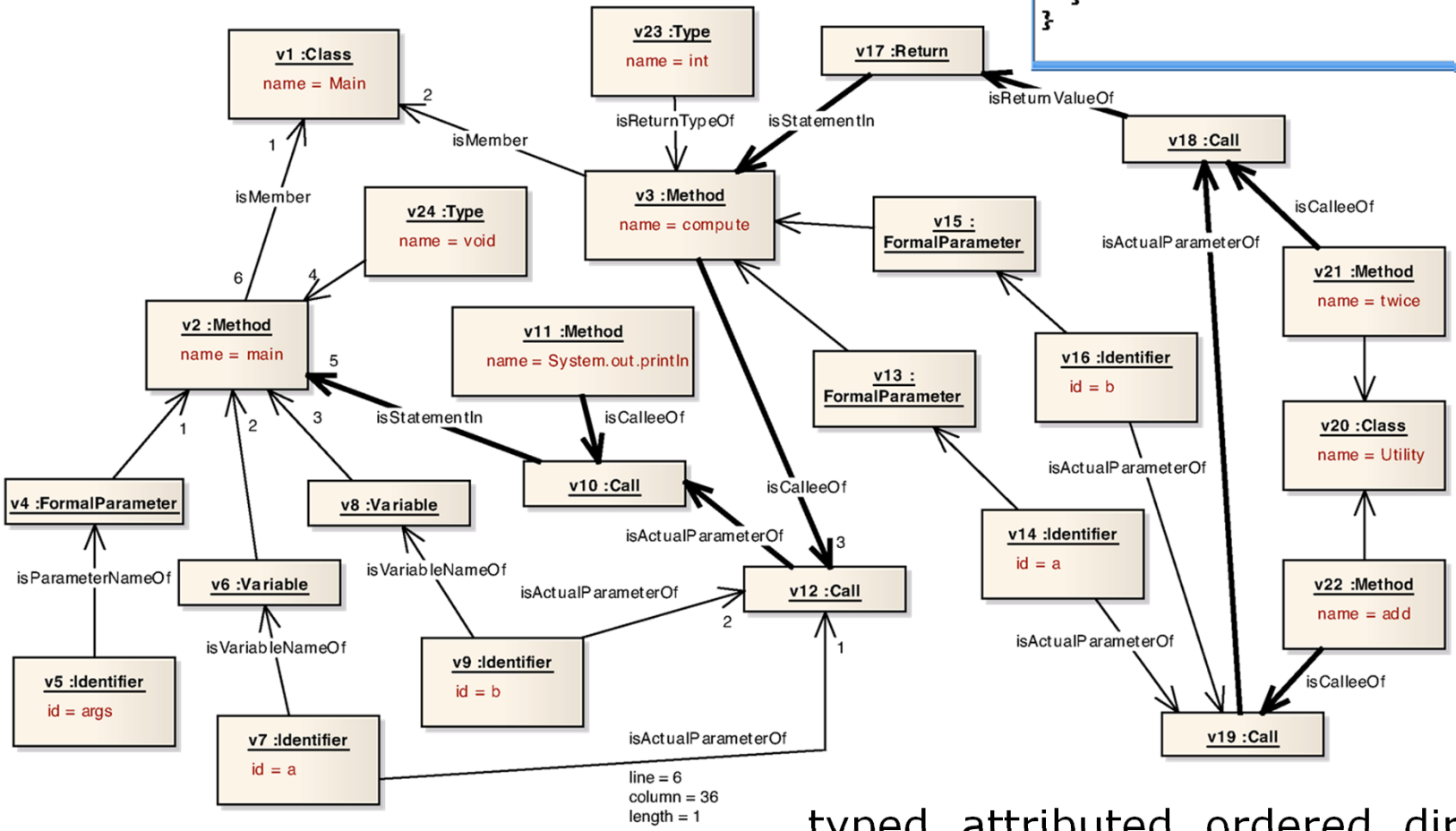
[*Ebert, Jürgen; Riediger, Volker; Winter, Andreas:* Graph Technology in Reverse Engineering, The TGraph Approach, In :10th Workshop Software Reengineering (WSR 2008), LNI 126, pp. 67-81,2008]

TGraphs

Main.java

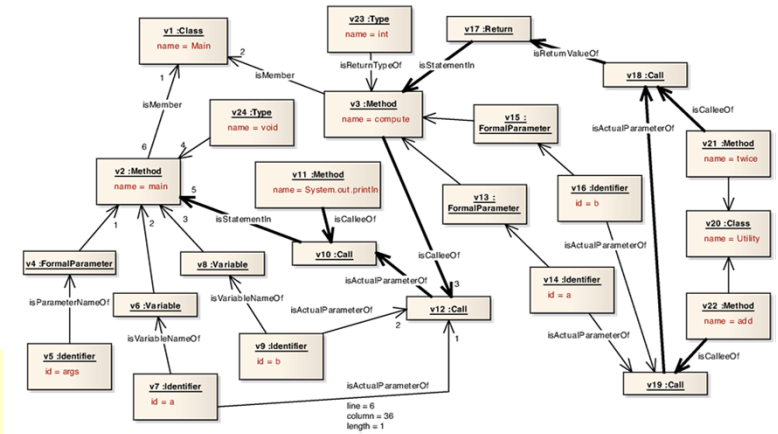
File Edit Search Preferences Shell Macro Windows Help

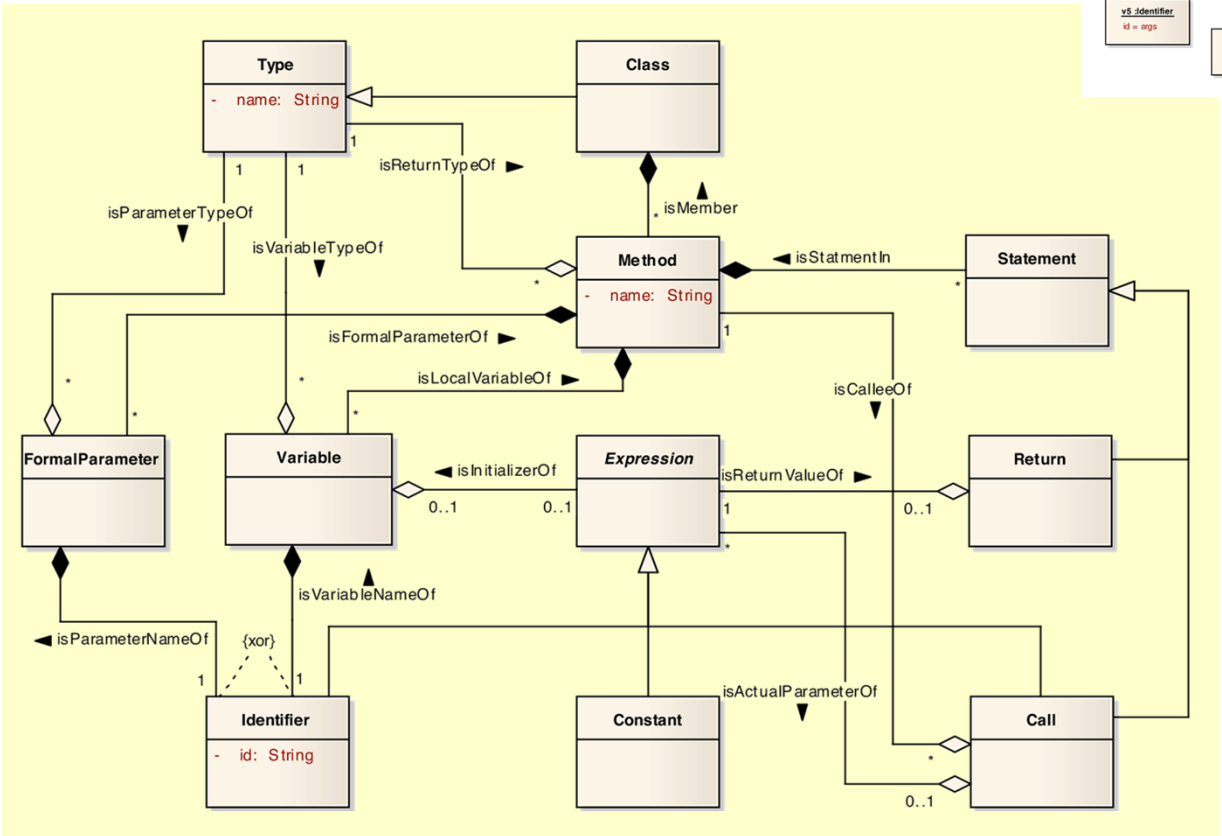
```
public class Main {
  public static void main(String[] args) {
    int a = 26;
    int b = -5;

    System.out.println(compute(a, b));
  }

  private static int compute(int a, int b) {
    return Utility.twice(Utility.add(a, b));
  }
}
```



typed, attributed, ordered, directed graph

TGraph Schema





grUML Schema

Classes:
- node types

Associations:
- edge types

attributes:
- node and
- edge attributes

Multiplicites:
- degree restrictions

Graph-Analysis

wanted

- efficient analysis of graph structures

GReQL graph queries

- metamodel based qurey language for Tgraphs, with
  - regular path expressions
  - transitive closure
  - extendible library for graph predicates and functions

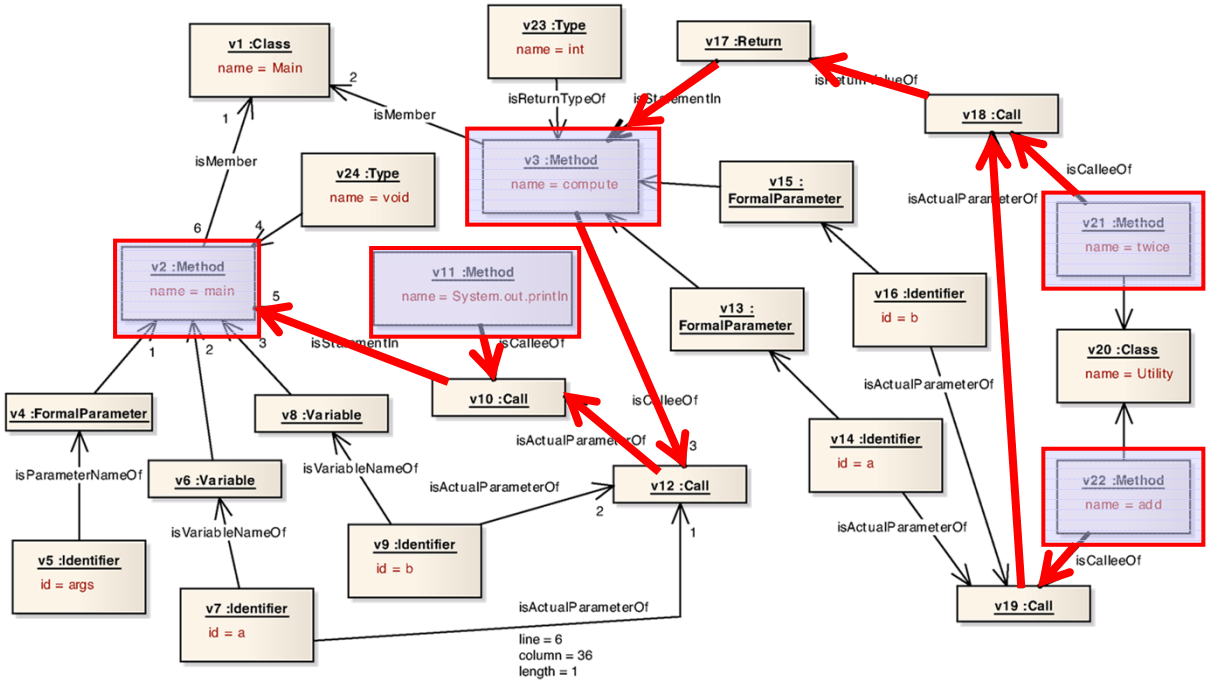
```
from
    declaration
with
    predicate
report
    result description
end
```

Query Example

show all caller/callee-pairs

Caller	Callee
main	System.out.println
main	compute
main	twice
main	add
compute	twice
compute	add

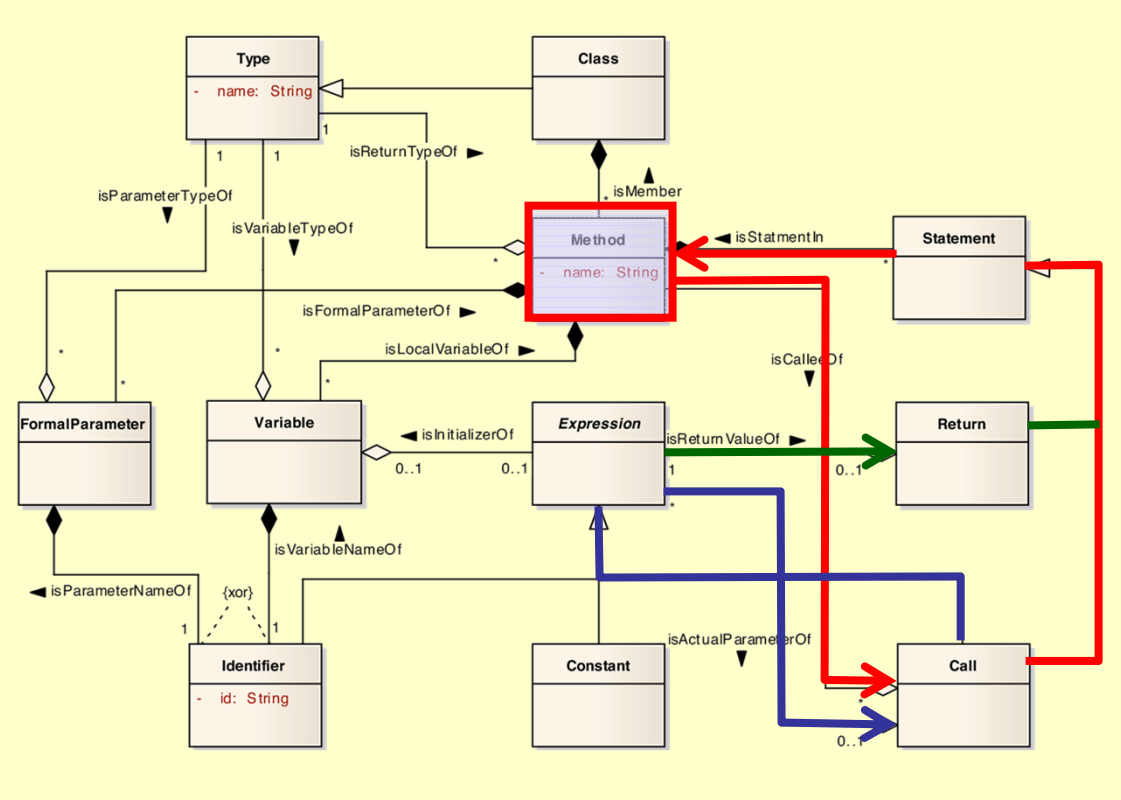
```
from
   caller,callee: V{Method}
with caller (
   --> {isStatementIn}
   [ <-- {isReturnValueOf} ]
  <-- {isActualParameterOf}*
  <-- {isCalleeOf}
  )* callee
report
   caller.name as "Caller"
   callee.name as "Callee"
end
```



Anfragebeispiel

show all caller/callee-pairs

```
from
   caller,callee: V{Method}
with caller (
   --> {isStatementIn}
   [ <-- {isReturnValueOf} ]
   <-- {isActualParameterOf}*
   <-- {isCalleeOf}
   )* callee
report
   caller.name as "Caller"
   callee.name as "Callee"
end
```



Tool Support (GUPRO)

QGupro

File Edit Project Query Window Help Settings

GuproWorkspace

Workspace
- Cosmos
  - results
  - variables
  - c
    - cosmos-client
      - client.c
      - comm.c
      - ident.c
      - debug.c
      - server.c

Queries
- functions_ca...
- functions_ca...
- functions_ca...
- functions_wi...
- identifier.grq
- identifier_co...
- identifier_wit...
- indirect_func...
- indirect_func...
- indirect_func...

Query Editor

```
FROM    caller, callee: V{Identifier}
WITH
        caller
        (  -->{isDirectDeclaratorIn}+
           -->{isFunctionDeclaratorIn}
           <--{isCompoundStatementIn}
           ((
             <--{isStmtIn}*
           ) | (
            <--{isDeclarationIn}+
            <--{isInitDeclaratorIn}
            <--{isInitializationIn}
           ))
           <--{isExprIn}*
           <--{isFunctionNameIn}
        )+
        callee
REPORT  caller.name AS Caller,
        callee.name AS Callee
END
```

Query Result

Graph id = b08340cdbd0cabe86370f7

Computation time = 80 ms

Result size = 409

Caller	Callee
message	__builtin_va_start
error_msg	__builtin_va_start
info_msg	__builtin_va_start

debug.c

```
void message (int printident, char *fmt, ...)
{
        va_list ap;

        if (printident) {
                print_identification(
        }

        va_start(ap, fmt);
        vfprintf(INFOFP, fmt, ap);
```

Software Evolution Projects

SOAMIG: Migrating Legacy Software to Service-Oriented Architectures

SES: Software-Evolution Services

EEA: Removing Energy Code Smells with Reengineering Services

Modeling Deltas: Version Control for Software-Models

Motivation

Software Migration

- provides transferring software systems to a new environment without changing its functionality
- enables reusing legacy assets in new environments
- preserves value of existing software systems during software evolution

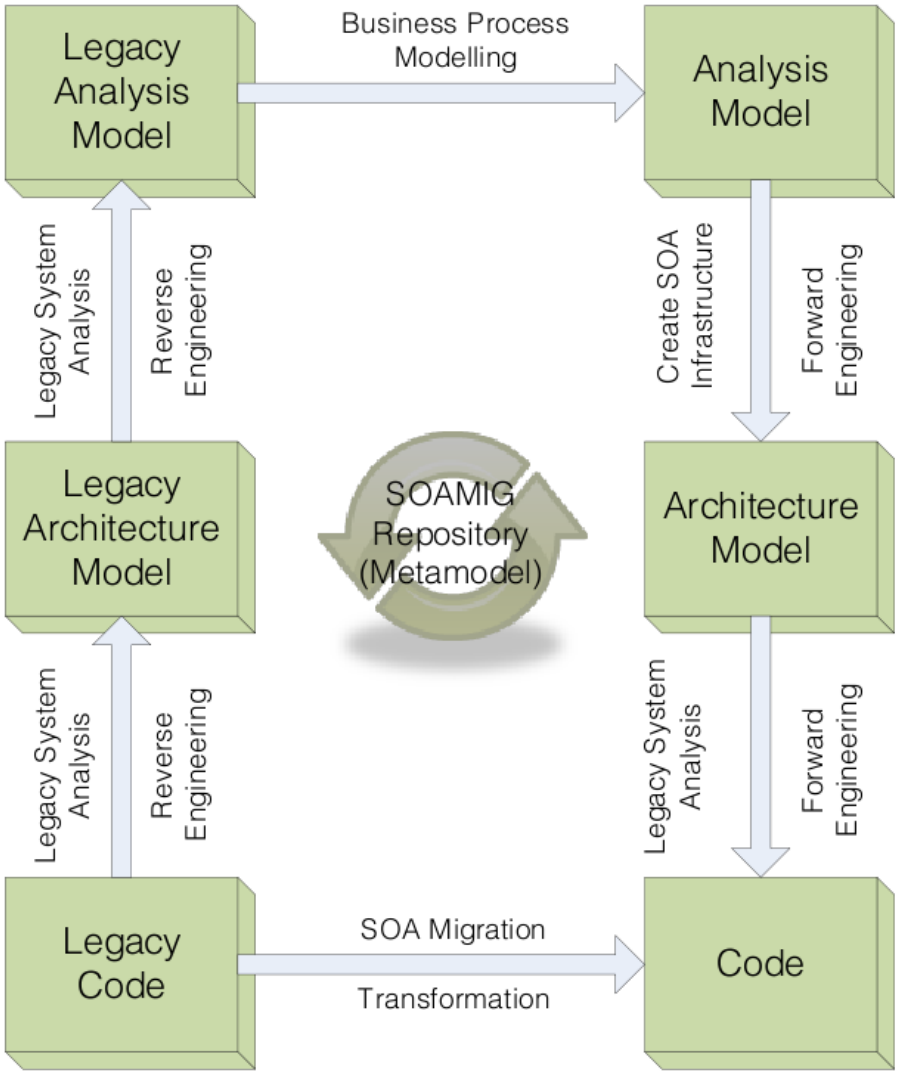
SOAMIG

- focuses on migrating software assets
  - by transformation
  - to service-oriented architectures (SOA)
- addresses architecture and language migration

soamig

Project idea

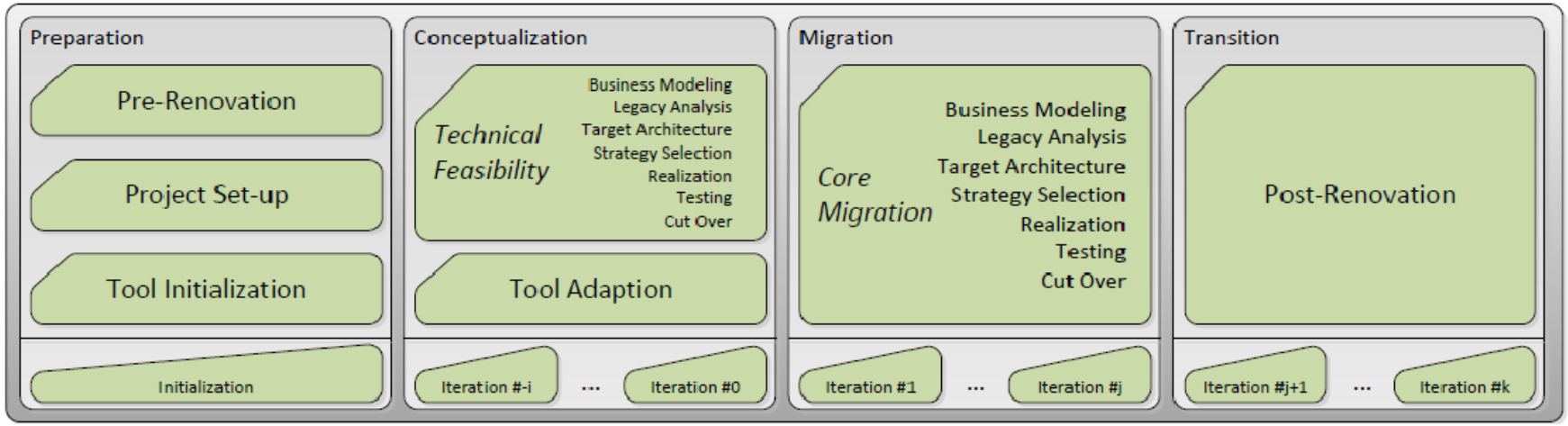
- use of model-driven techniques
  - integrated meta-model representing legacy and target system on
    - business process level
    - architecture level
    - code level
  - meta-model based reverse-engineering and software-analysis by
    - querying
    - transforming



SOAMIG Process Model

SOAMIG Contribution

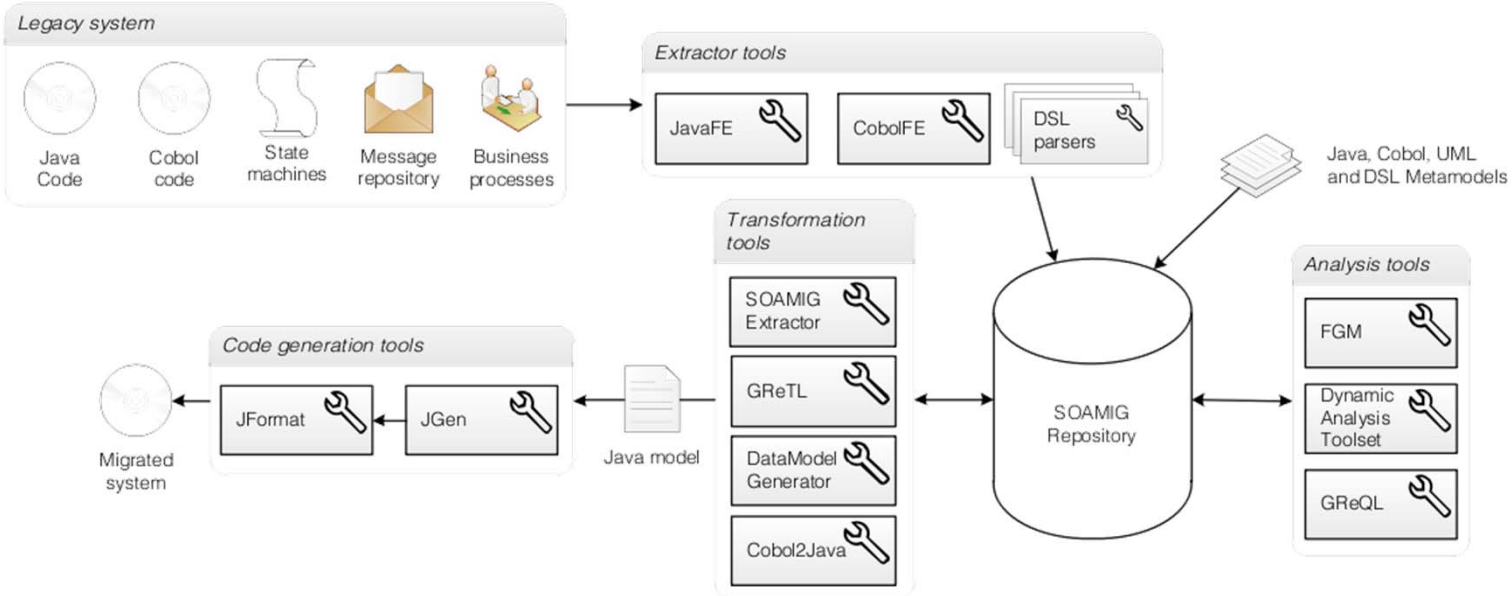
- adaptable, iterative process model
- phases for
  - Preparation (incl. setting up a migration factory)
  - conceptualization (incl. feasibility analysis and tool adaption)
  - (core) migration (incl. migration disciplines)
  - post-renovation (incl. reengineering activities)



SOAMIG Tool Chain

SOAMIG Contribution

- migration tool support integrated by a graph-based repository
  - Parser/Unparser (JavaFE, CobolFE, DSL-Parser, JGen, JFormat)
  - Repository based Analysis (FGM, JGraLab, Dynamic Analysis)
  - Repository based Transformation (Cobol->Java-Translator, Datamodel Generator, SOAMIG-Extractor)



Evaluation and Application

Case Study 1: RAIL

Migration Type

- architecture migration to SOA

Legacy System

- selling Deutsche Bahn products, developed by Amadeus
- monolithic 229 228 LOC Java rich client

Migration Objectives

- reducing deployment costs
- increasing reusability of functional components

Migration in SOAMIG

- focuses on technical feasibility

Case Study 2: LCOBOL

Migration Type

- code migration

Legacy System

- transaction system, maintained by pro et con
- monolithic rich client
- 81 600 LOC MF-COBOL/SQL

Migration Objectives

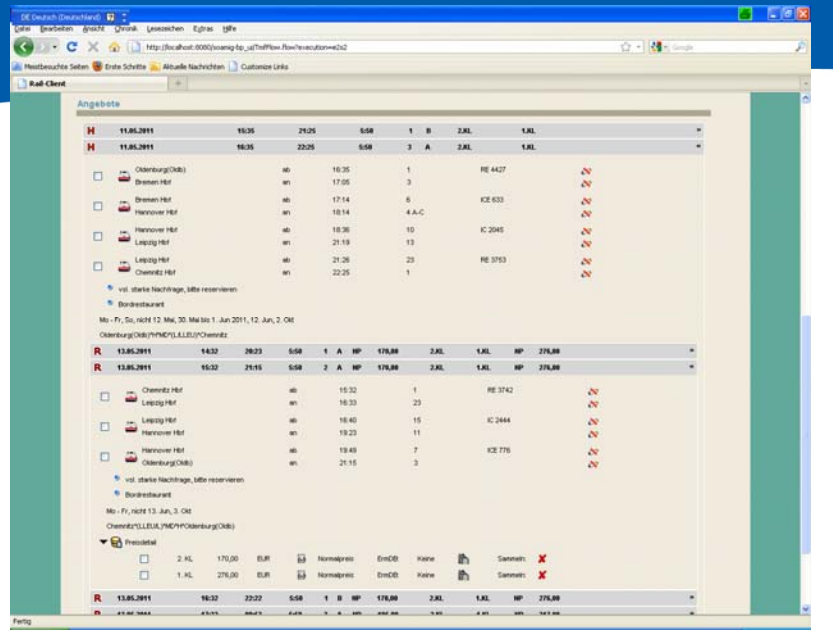
- increasing reusability and subsequent use in web service based environment

Migration in SOAMIG

- focuses on full language migration

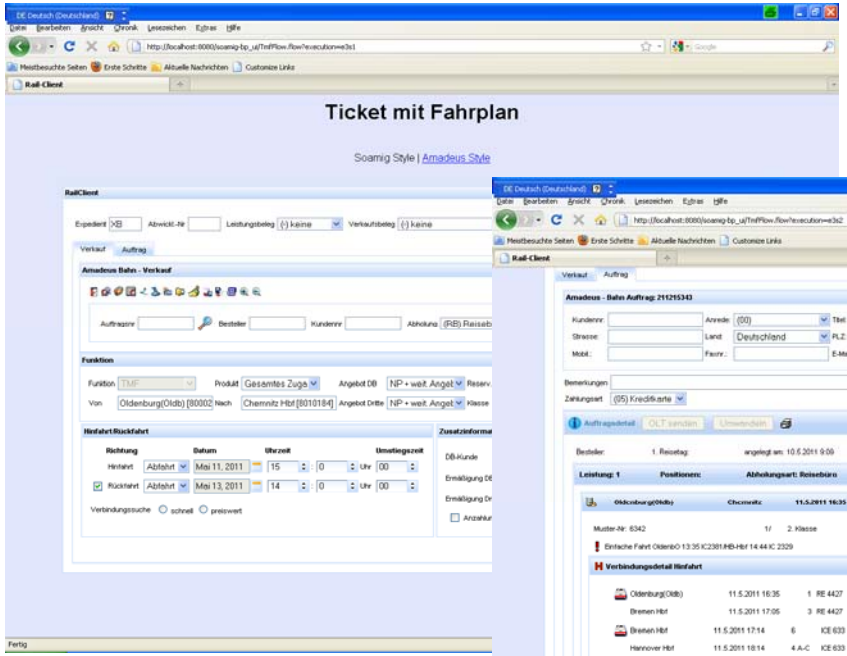
CARL VON OSSIETZKY universität OLDENBURG

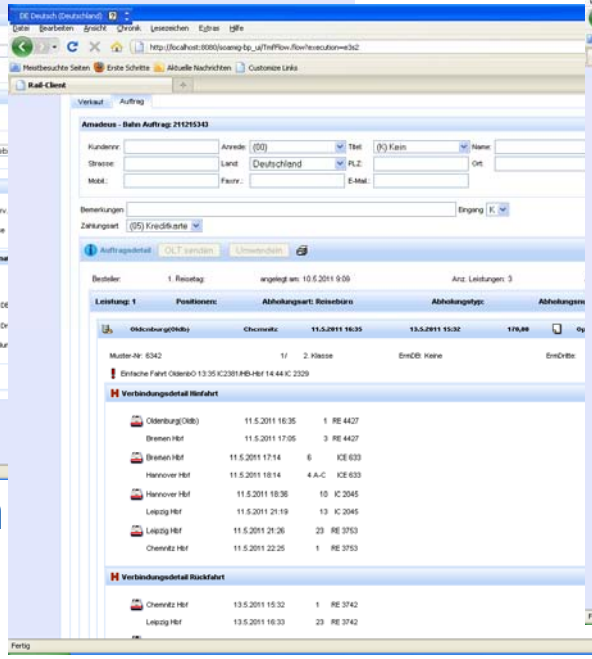
legacy system

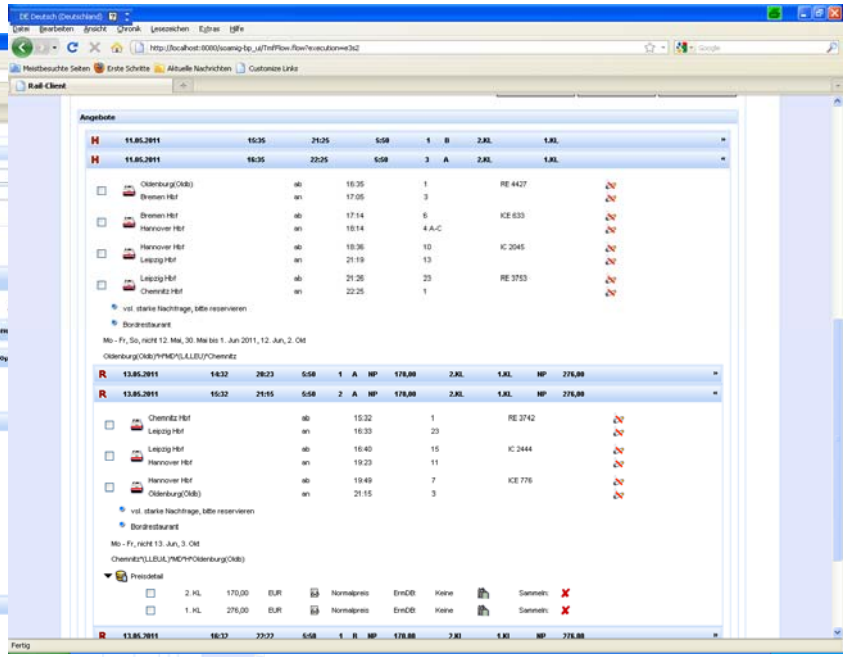


Rail: Target GUI

Presentation Layer (by re-implementation)







after migration

Removing Energy Code Smells with Reengineering Services

Motivation

Consumption:

- over 10 % of Germany's overall electrical energy consumption due to ICT by 2007

Environment:

- CO2 emissions higher than entire German aviation sector

Mobility:

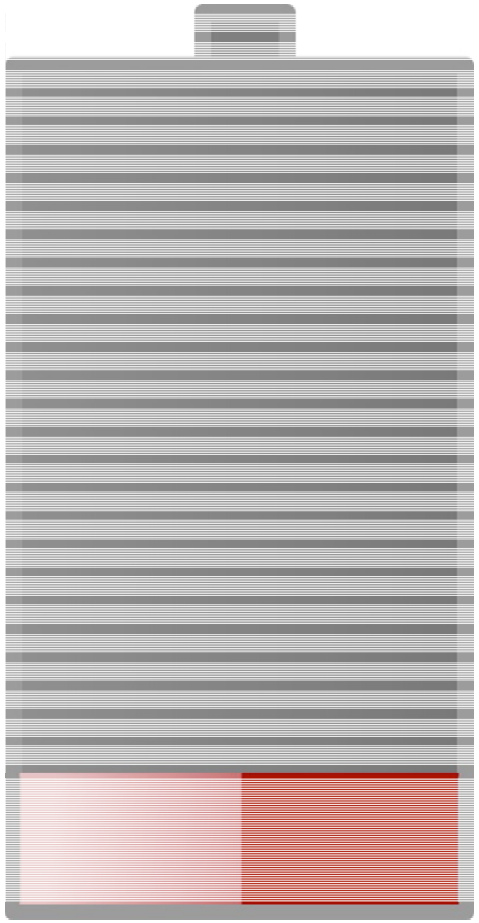
- ubiquitous and powerful mobile devices, batteries cannot keep up

Research:

- focused on hardware and low-level software optimizations

Our Focus:

- application level with feedback to OS, mobile computing



Refactoring = Identification + Restructuring

Energy Code Smells

- Energy code smells are energy-inefficient patterns in code
- Platform dependent as well as platform independent energy code smells exit

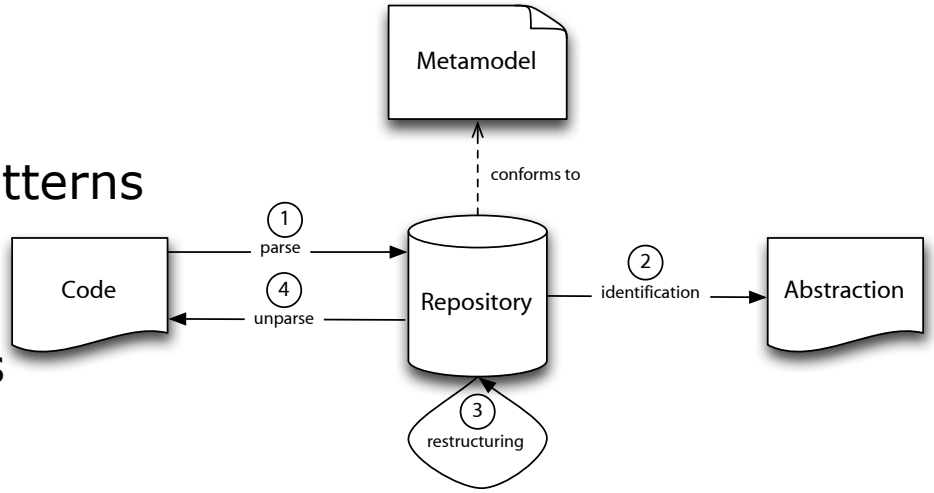
Process

① Preparation: parse source to TGraphs

② Identification: identify code patterns by graph queries (GReQL)

③ Restructuring: improve code through graph transformations

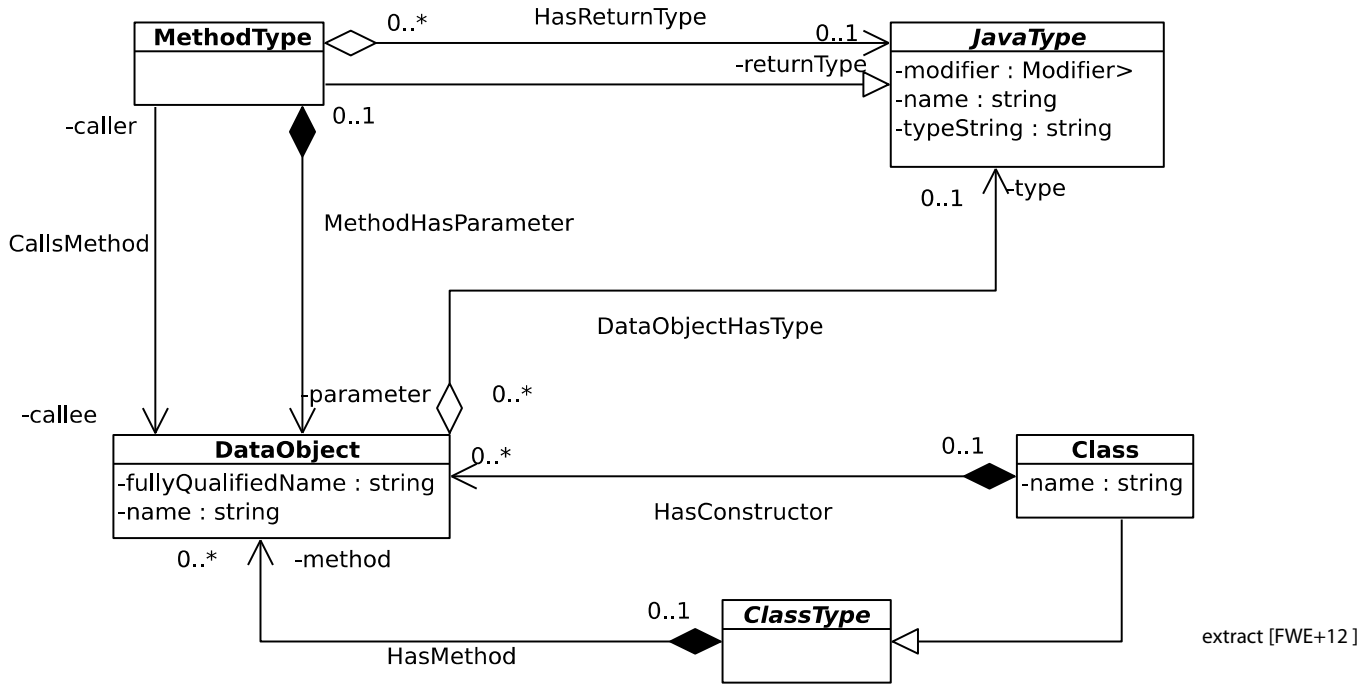
④ Post processing: unparse TGraphs to source



Meta Model for Graph Representation

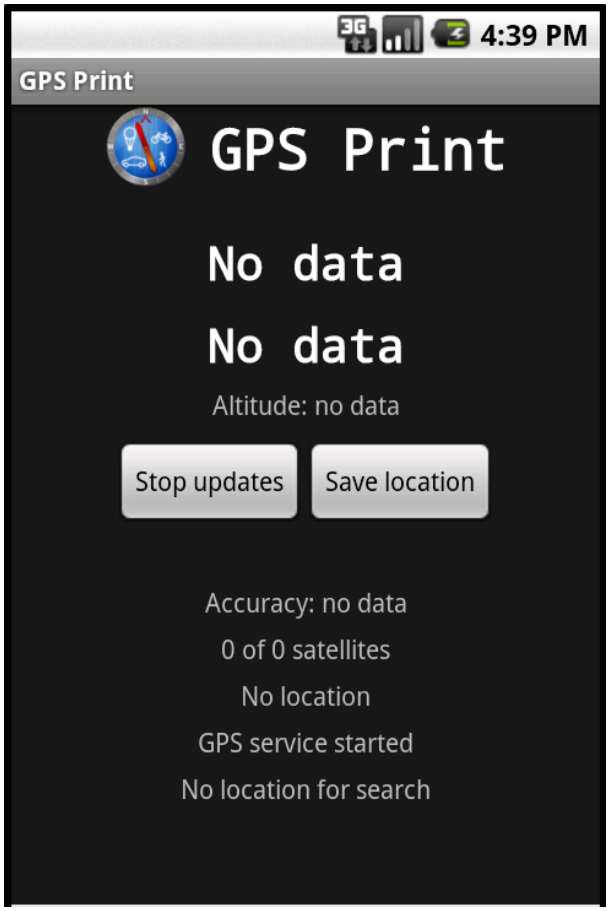
Representation of Java Code

- According to SOAMIG Java Meta Model
- Originally contains 86 node types and 67 edge types



Example: GPSPrint Android Application

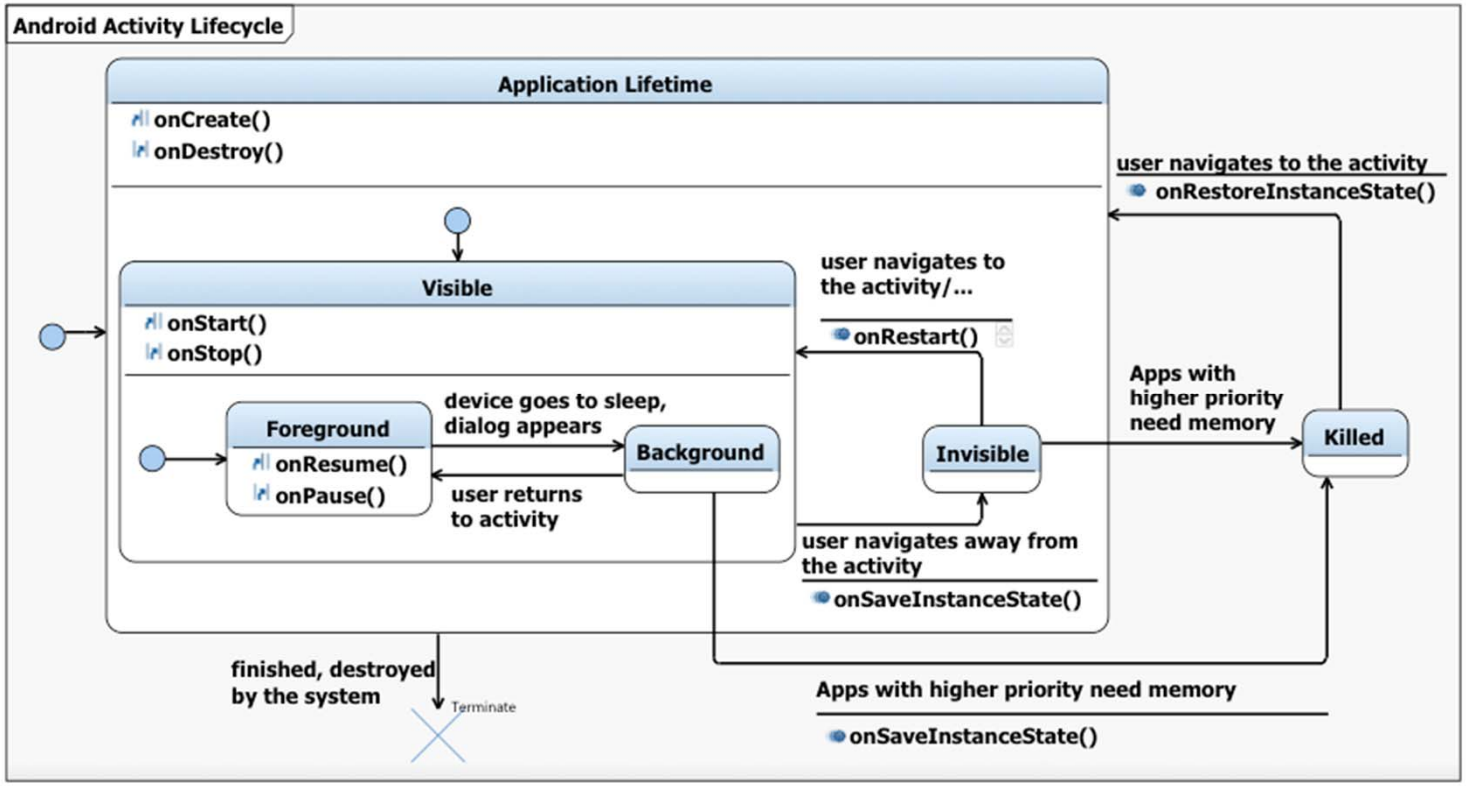
GPSPrint is a simple Android App showing actual GPS information like signal strength, count of satellites etc.



Example: Resource Utilization in Android

code smell

- claiming and releasing resources in onCreate() and onDestroy()



cmp. Android Developers: "Activities", 2012. `http://developer.android.com/guide/topics/fundamentals/activities.html`

Example: Resource Utilization in Android

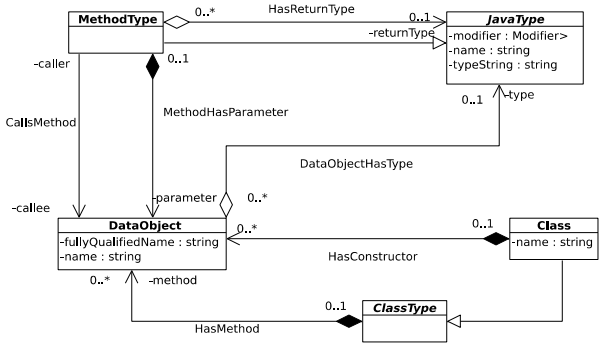
```
1  public class GpsPrint extends Activity
2      implements OnClickListener, Listener,
3      LocationListener {
4  [...]
5    public void onCreate(Bundle
6        savedInstanceState) {
7  [...]
8      LocationManager lm=(LocationManager)
9          this.getSystemService(Context.
10         LOCATION_SERVICE);
11     if(lm.getAllProviders().contains(
12         LocationManager.GPS_PROVIDER)) {
13       if(lm.isProviderEnabled(
14           LocationManager.GPS_PROVIDER)){
15         lm.addGpsStatusListener(this);
16         lm.requestLocationUpdates(LocationManager.
17             GPS_PROVIDER, 1000, 0, this);
18         status_view.setText(
19             "GPS service started");}
20       else {
21         status_view.setText(
22             "Please enable GPS");
23         save_location_button.setEnabled(
24             false); }
25 [...] }
26 [...]
27   public void onPause() {
28 [...]
29     lm.removeUpdates(this);
30 [...] }
31   public void onResume() {
32 [...]
33     lm.requestLocationUpdates(
34         LocationManager.GPS_PROVIDER,
35         1000, 0, this);
36 [...] }
37 }
```

Before Refactoring

```
1  public class GpsPrint extends Activity
2      implements OnClickListener, Listener,
3      LocationListener {
4  [...]
5    public void onCreate(Bundle
6        savedInstanceState) {
7  [...]
8      LocationManager lm=(LocationManager)
9          this.getSystemService(Context.
10         LOCATION_SERVICE);
11     if(lm.getAllProviders().contains(
12         LocationManager.GPS_PROVIDER)) {
13       if(lm.isProviderEnabled(
14           LocationManager.GPS_PROVIDER)){
15         lm.addGpsStatusListener(this);
16         //removed by refactoring
17
18         status_view.setText(
19             "GPS service started");}
20       else {
21         status_view.setText(
22             "Please enable GPS");
23         save_location_button.setEnabled(
24             false); }
25 [...] }
26 [...]
27   public void onPause() {
28 [...]
29     lm.removeUpdates(this);
30 [...] }
31   public void onResume() {
32 [...]
33     lm.requestLocationUpdates(
34         LocationManager.GPS_PROVIDER,
35         1000, 0, this);
36 [...] }
37 }
```

After Refactoring

Detecting Energy Code Smells via GReQL



JGraLabUI – GPSPrint.tg

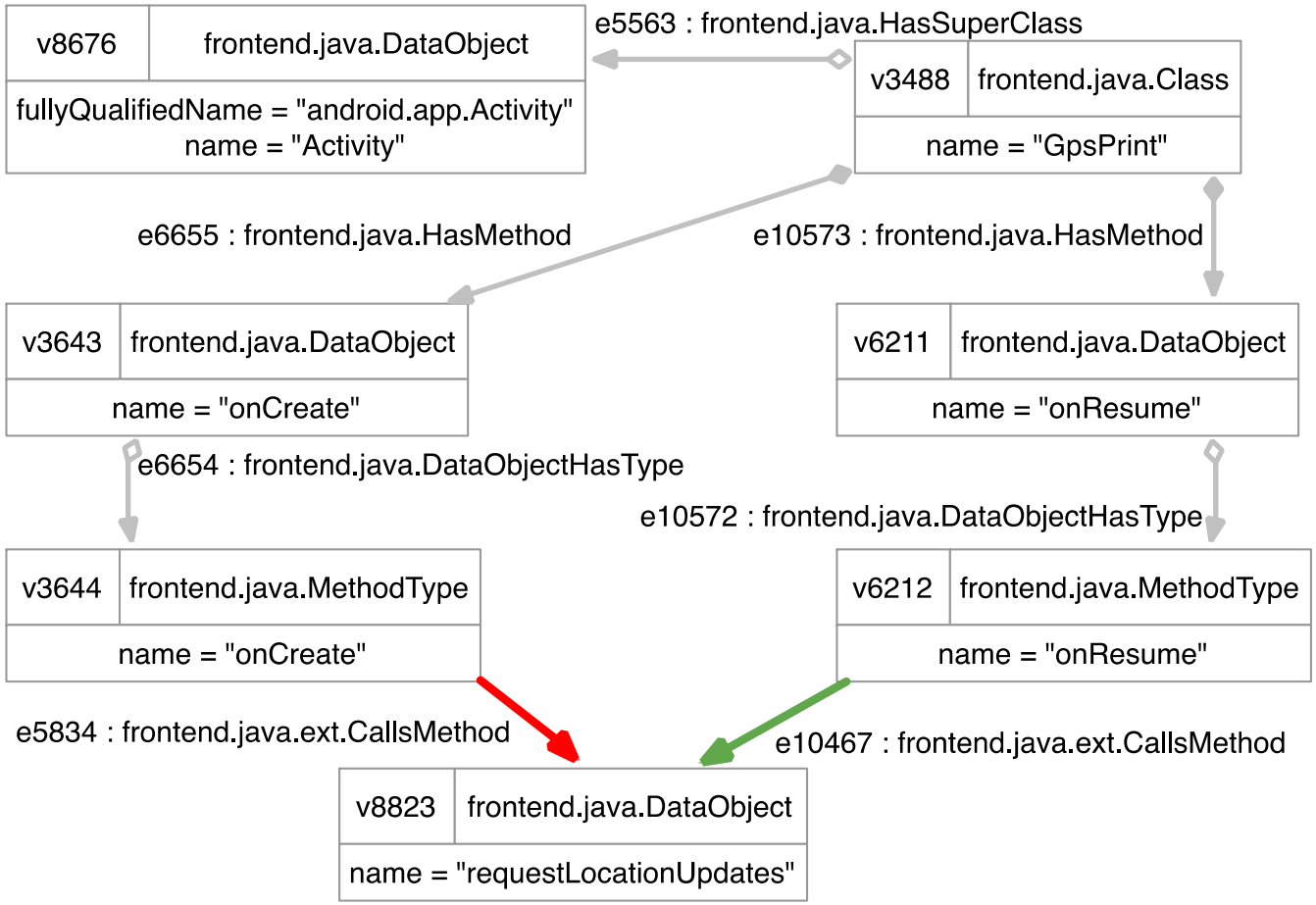
File Edit Query

```
from
        onCreate, caller : V{frontend.java.MethodType},
        actClass : V{frontend.java.Class},
        superClass, callee : V{frontend.java.DataObject}
with
        onCreate.name = "onCreate" and
        superClass.fullyQualifiedName = "android.app.Activity" and
        callee.name = "requestLocationUpdates" and
        callee <--{frontend.java.ext.CallsMethod} caller
        (<--{frontend.java.DataObjectHasType} <--{frontend.java.ext.CallsMethod})*
        onCreate <--{frontend.java.DataObjectHasType}
        <--{frontend.java.HasMethod} actClass
        -->{frontend.java.HasSuperClass} superClass
report
        actClass.name, caller.name
end
```

((GpsPrint, onCreate))

AST JAVA Graph opened.

Code Restructuring



Classes of Energy Code Smells

Loop Bug

- A program behaviour wherein an application is repeating the same activity

Dead Code

- Source code which is never used, but needs to be loaded into memory

In-line method

- Replacing a method call with the actual body of the called method

Moving too much data

- Unnecessary communication between processor and memory

Immortality Bug

- Describes applications respawning after explicitly being killed by the user

Redundant storage of data

- Different methods of an application store the same data in memory

Using expensive resources

- Swap energy-expensive resources against “cheaper” alternatives

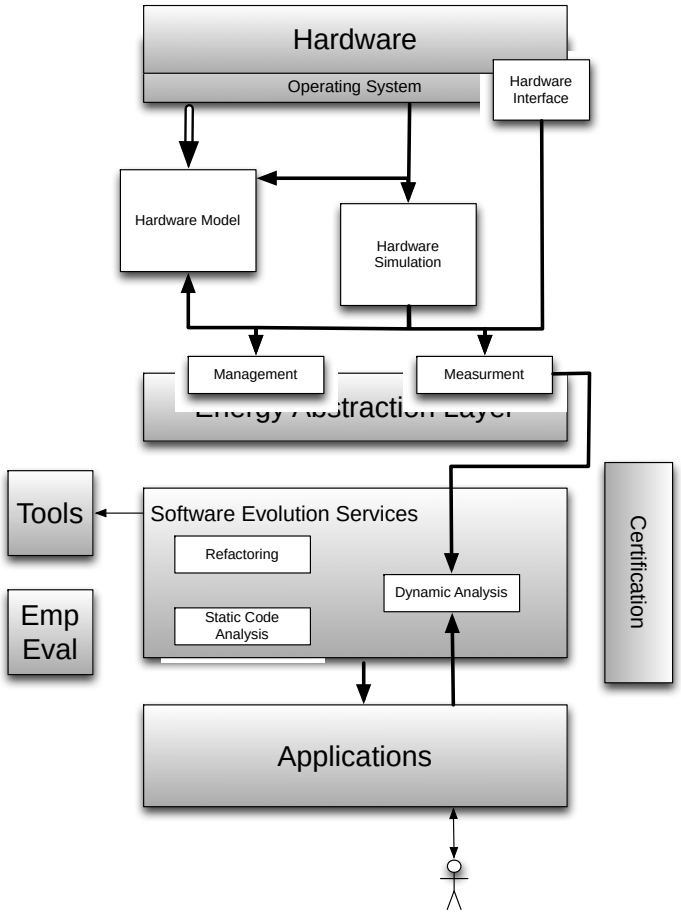
Infrastructure for Measurement and Management

Layered Structure on Mobile Devices

- Hardware which runs the operating system
- application environment (sometimes; e.g. Dalvik VM)
- Application layer interacts with the user

Energy Abstraction Layer

- between hardware and applications
- OS and hardware independent
- Abstract specification for measurement and management of energy



Summary

http://www.kleinezeitung.at/freizeit/aktivwellness/touren/623790/index.do

Foundations

- Software-Engineering
- Software-Evolution

Graph-Technology

- Graph-based Modeling
- Graph-Querying

Current Activities

- SOAMIG: Migrating Legacy Software to Service-Oriented Architectures
- SES: Software-Evolution Services
- EEA: Removing Energy Code Smells with Reengineering Services
- Modeling Deltas: Version Control for Software-Models