Lecture 2

Software Re-engineering

Some material is based on the CSER projects at U of T
Covers almost all concepts of the course
Detail explanations to come …

Last lecture …

General Information

- Instructor: Yijun Yu yijun@cs.toronto.edu
- Office: BA7200 (Bahen Center, 7th floor), 946-8530
- Office hours: Wed 5pm – 6pm, Fri 2pm-3pm
- TA: Alexia Giannoula alexia@comm.utoronto.ca
  Clark Merchant Clark.Merchant@utoronto.ca
  Mazen Almaoui mazen@dsp.utoronto.ca
- Class homepage: http://www.cs.toronto.edu/~yijun/ece450h

Marking Scheme adjusted

- No midterm
- Final Exam 50% (Exam week)
- Course Project 50%
  - Assignment 1 (15%): Feb 11
  - Assignment 2 (15%): Feb 25
  - Assignment 3 (20%): April 8

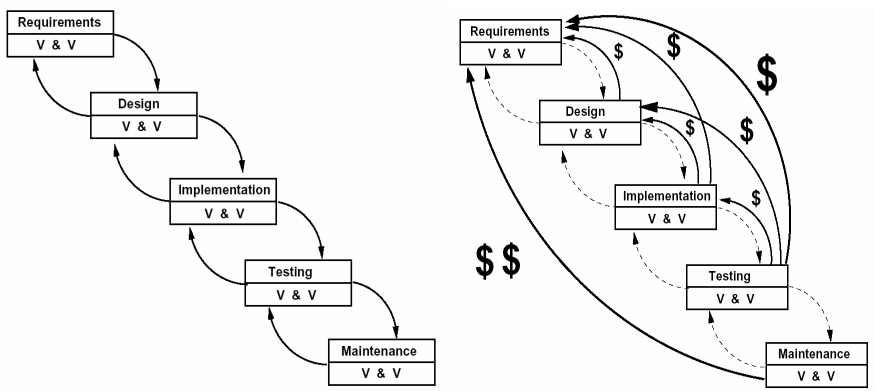
Our Course Project

- This is a “brand-new” software reengineering project, emphasizing on reusing, restructuring, refactoring large-scale software systems, and team work !
  - A1: Understanding the architecture of a legacy system (OpenOME, OmniEditor) (15%)
  - A2: Design OmniGraphEditor web service (15%)
  - A3: Reengineering OpenOME to use OmniGraphEditor web service of other teams (20%)
- Tutorials will cover detailed approaches and tools to help you with the project

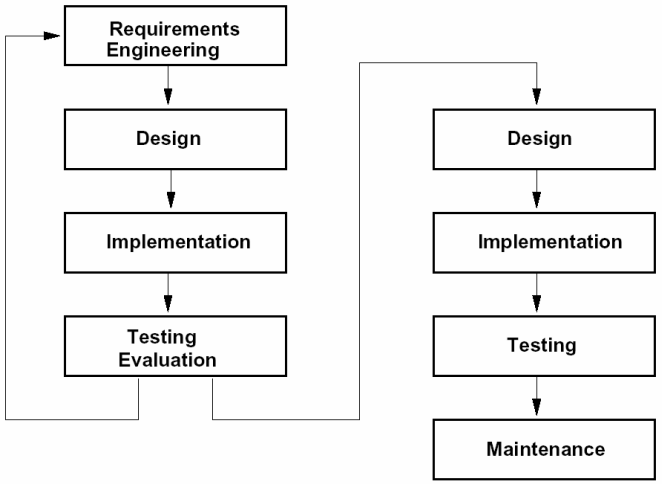
Today …

1. Review SE process
2. Discuss Reengineering Concepts
3. Go over some case studies, a road map to our lectures and tutorials:
   VIM: componentization, reveal architectures
   osCommerce: aspect elicitation, reveal requirements
   SquirrelMail: goal elicitation from refactored code
4. Your exercise is to use the learnt knowledge to study two other legacy software systems:
   OpenOME and *OmniEditor*
5. Summary

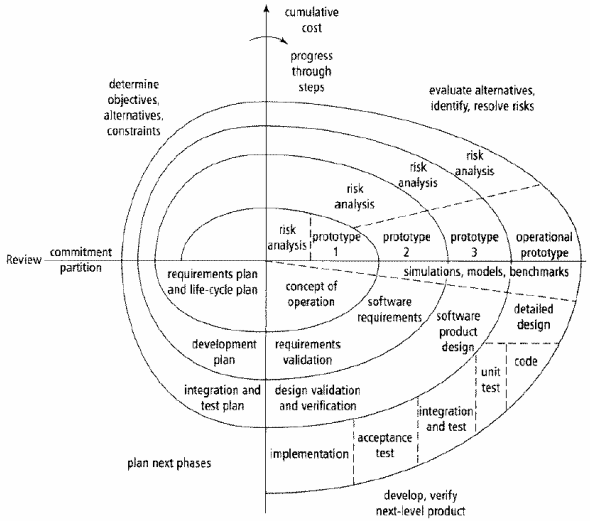
The Waterfall process model



Rapid Prototyping process



Spiral (incremental) process



2. Reengineering concepts

- Why Software Reengineering?
  - Legacy software are increasing (Software vs. Hardware)
  - New technology appearing (Moore's law)
  - Successful ratio of projects increasing (IBM internal history)
  - Companies are more competing (now we have the "open-source" movement and free-software foundation)
  - Quality attributes are demanding (That's the selling point)
  - People are changing (developers joining and leaving, customers are changing)
  - Software maintenance are pressing (Largest cost in software development lifecycle >60%)

2. Reengineering concepts
What is software reengineering?

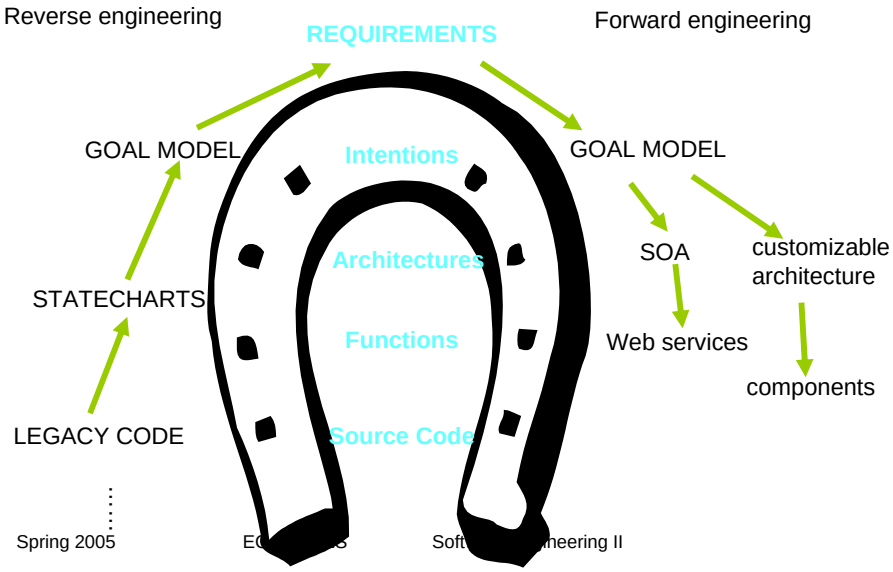
To a large extent, it involves maintenance activities:

- Understanding (predictive)
- Repairing (corrective)
- Improving (perfective)
- Evolving (adaptive)

- Related topics
  - Quality-driven software engineering (-ilities, quality attributes)
  - Requirements engineering (goals, non-functional requirements)
  - Software architectures (architectural views: components, statecharts, features, …)
  - Model-driven development (MOF, UML, EMF)
  - Design patterns (structural, behavioural)
  - Software refactoring (the code smells)
  - Performance tuning (trade-offs, multi-criteria optimizations)
  - Paradigms: Object-oriented, Goal-oriented, Agent-oriented, Aspect-oriented…

2. Reengineering concepts
The Horseshoe model

The Reengineering Horseshoe



Reading assignments on software architectures

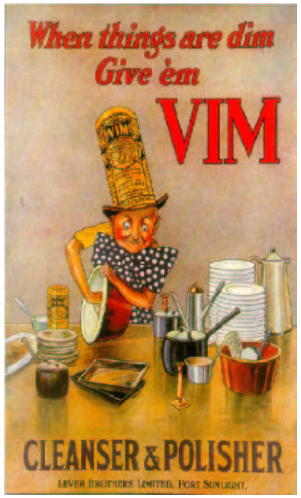
- Previous lecture note for ECE450H1S: "What is software architecture?" "How to represent it?"
  - D. Penny. "Introduction to software architecture": http://www.cs.toronto.edu/~chechik/courses00/ece450/lectures/penny.2up.pdf
  - M. Chechnik. "ADL and Darwin". http://www.cs.toronto.edu/~chechik/courses00/ece450/lectures/Marsha-Darwin.pdf

Further readings

- Martin Fowler. "The Refactoring homepage": http://www.refactoring.com/
- CMU SEI: "Software architecture". http://www.sei.cmu.edu/ata/ata_init.html
- KMLab. "On goal oriented software engineering". http://www.cs.utoronto.ca/km/goal_oriented

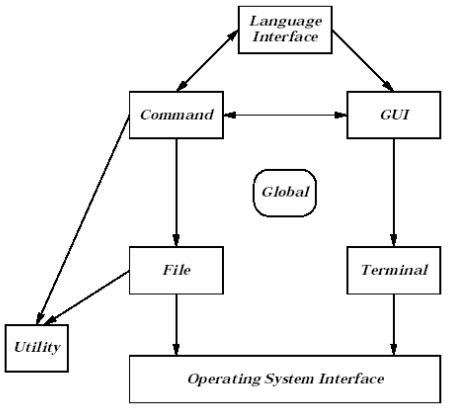
Case Study I. VIM

- VIM stands for Vi-IMproved http://www.vim.org
- Are you a VIMer?
- Current version 6.3
- Bram Moolenaar
- Developed in C
- 172 KLOC



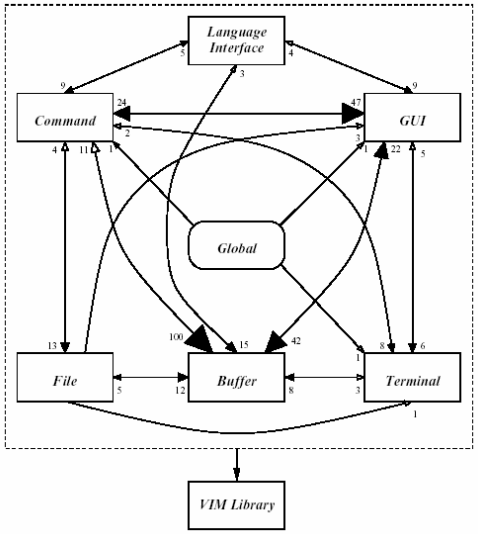
Understanding the architecture of VIM

- Lee's initial VIM architecture



John Tran et al. “Architectural Repair of Open Source Software”. IWPC 2000.

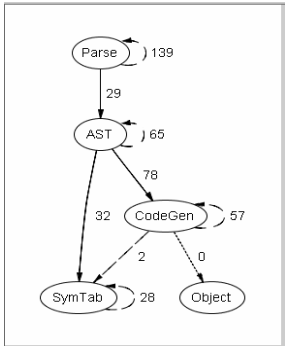
- Vim 5.3



G. Murphy, et al. “Software Reflexion Models: Bridging the gap between design and implementation”, IEEE Trans. On Software Engineering 27(4):364-380, 2001.

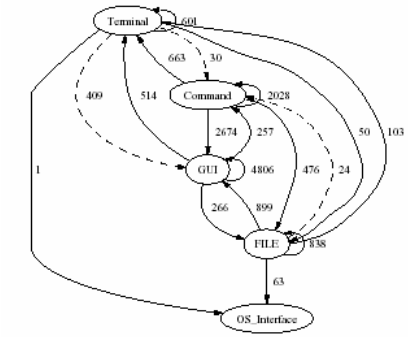
- Reflexion model (jRMTool)

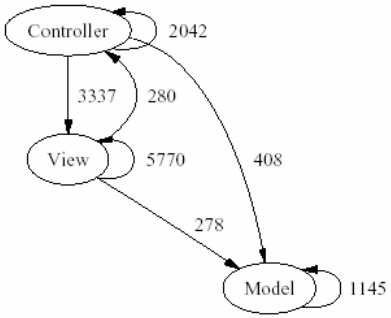
- High-level model (HLM) multi-graph
- Source model (SM) multi-graph (source code or trace)
- Mapping from SM to HLM is defined by regular expressions
- Identify three kinds of edges:
  - Convergence
  - Divergence
  - Absence



H. Dayani-Fard, Y. Yu, J. Mylopoulos, P. Andritsos. “*Improving the build architecture of legacy C/C++ software systems*”, Fundamental Approaches to Software Engineering, April 2005. to appear

- http://www.cs.toronto.edu/~yijun/literature/paper/dayani-fard05fase.pdf
- VIM 6.2

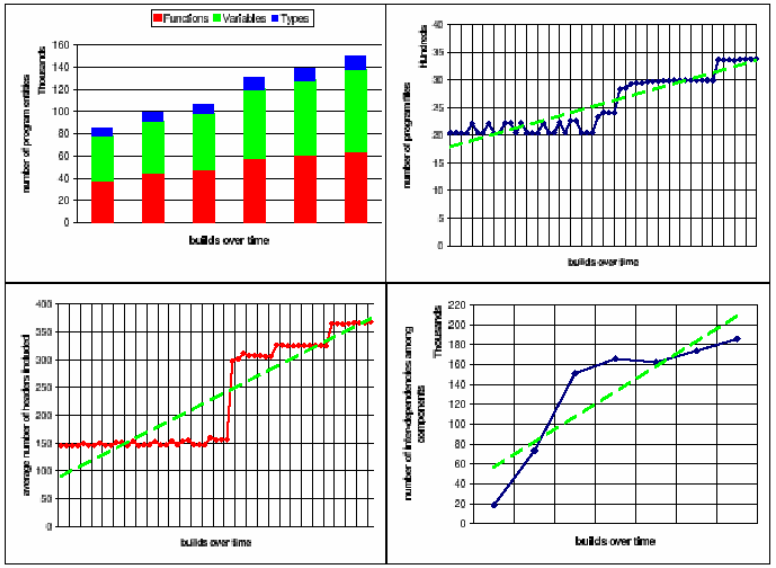




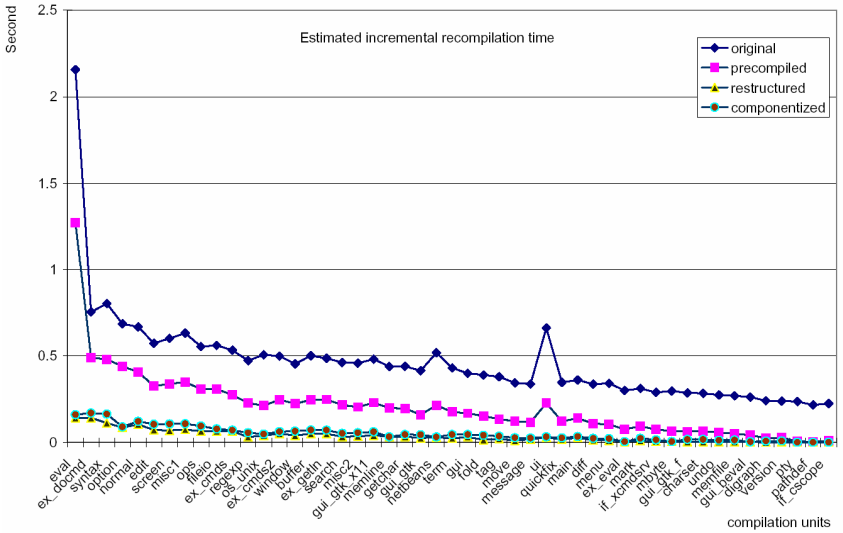
Restructuring headers

- Components provides and uses interfaces
- In C/C++, such as VIM, interfaces are written in Headers
- “Abstraction and information hiding” is a good principle in SE, thus we should do the componentization …
- “Large-cohesion and Low coupling” is the modularity principle of SE
- The inclusion of the headers may violate this principle
  - Too much entities included leads to redundancies, and also
  - False dependencies
- It is an advanced topic to show how to restructure the program to remove all false dependencies
- And also componentize the program to minimize the number of interfaces.
- Implementation in the adapted version of GCC 3.4.0
- Applications to IBM database product and potentially a Wind River product

Motivation: Decaying metrics of an industrial product

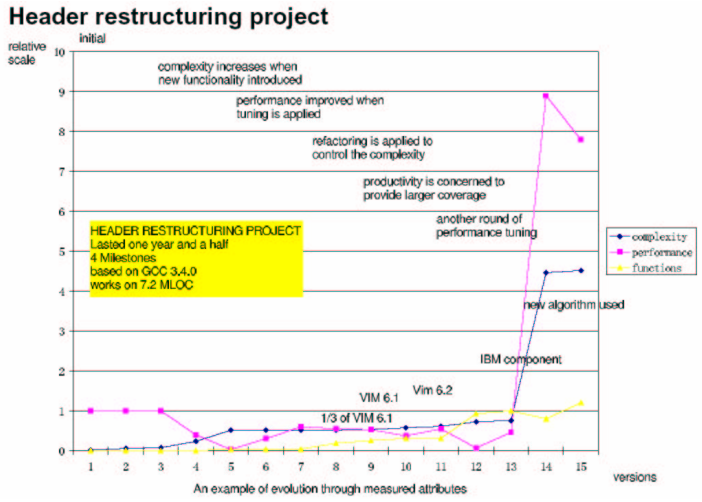


Build performance results

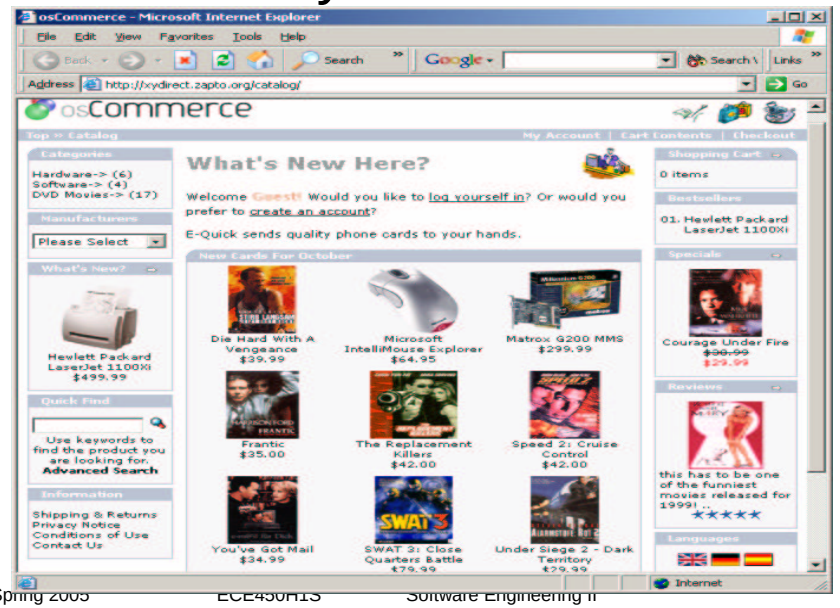


Quality-driven software refactoring

- Refactoring is a technique to reveal hidden structure of the system. It helps maintainability by reducing complexity, but may hurt performance…



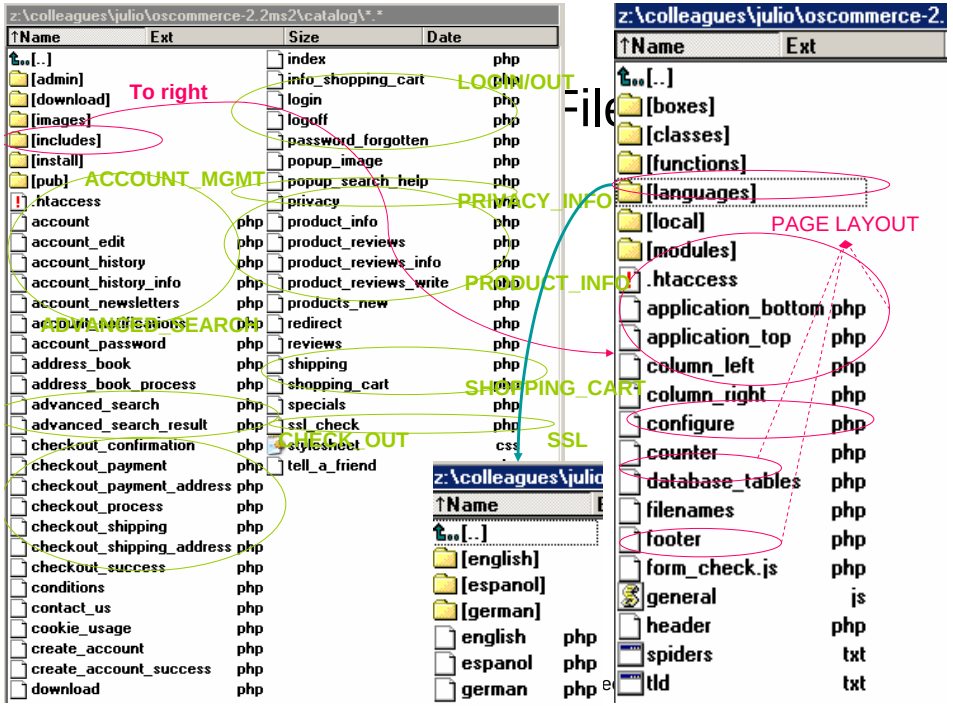
Case Study II. osCommerce



Motivation

- PHP, 65 KLOC
- It is an parallel implementation of the Media Shop, an information system example in Goal-oriented Requirements Engineering
- It has been studied by clone detection
- We want to show the connection of goal models with aspect elicitation
  Y. Yu, J.C. Leite, J. Mylopulos. "**From Goals to Aspects: Discovering Aspects from Requirements Goal Models**", RE 2004. 38-47.

Aspect-Orientation changes the way of thinking

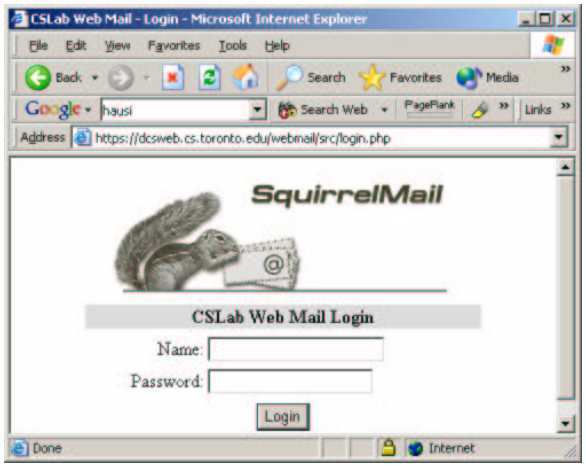


Case Study III. Squirrel Mail

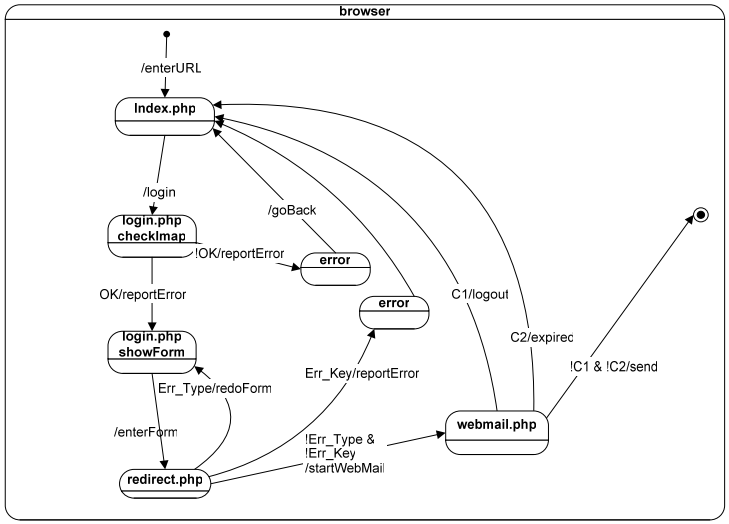
- It is a web-based email system used by the CS department
- We will explore the steps on how to refactor it to reveal the intention of developers: Code -> Statechart -> Goals
- The research is on-going on building the tool support. It will be associated with a tutorial on Eclipse tools

The appearance of the system

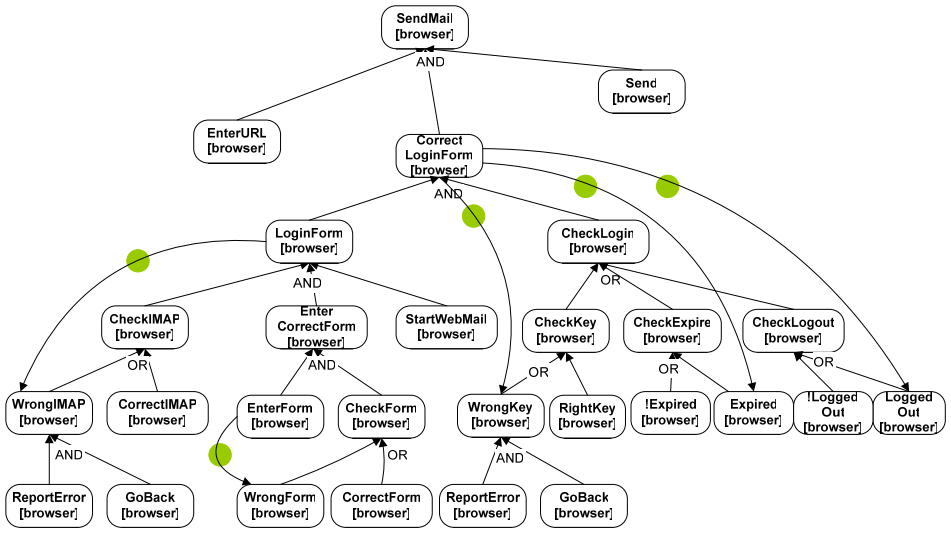
SquirrelMail 1.5.0
Open source
70 KLOC
PHP + HTML



A result of the refactoring



A result goal model



Summary

- Reengineering is a hot topic in the software engineering research
- Case studies show some ways to understand a legacy software
- We will use several tutorials to explore further on individual case studies, explaining advanced topics on:
  - The concepts of software architecture (components, service-oriented architecture, build architecture), aspect-oriented paradigm, software refactoring
  - The software engineering tools for these tasks, including code fact extraction, reflexion model, Eclipse, aspectJ etc.
  - How to apply them to our course project

What's next …

- A Tutorial on Web Services
- Next lecture will give you some examples of requirement specifications and project documents
- Do we cover the material you want to learn? If no, please send me email and see whether the course can motivate your study …