Retrospective Software Evolution Analysis

Harald Gall and Mehdi Jazayeri
Technical University of Vienna
Distributed Systems Group

A-1040 Wien, Argentinierstraße 8/184-1
{gall,jazayeri}@infosys.tuwien.ac.at

Foundations of Software Evolution Network, January 18, 2002

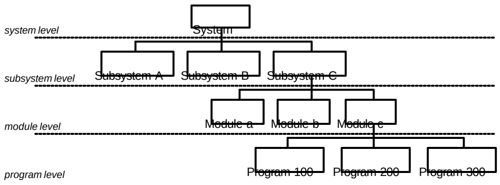
Outline

- Software Evolution Analysis (SEA)
  - Goals and approach
- SEA processes:
  - Change Sequence Analysis (CSA)
  - Change Report Analysis (CRA)
- Software evolution and features
- Conclusions and outlook

Software Evolution Analysis

- Problem
  - analyze the evolution of (very) large systems (e.g. 10 MLOC, 4 programming languages) across many releases
  - reveal hidden dependencies among modules
  - identify parts to be restructured/reengineered
- Approach:
  - observe software evolution via release history
  - detect logical coupling via
    - change sequence analysis, and
    - change report analysis
  - visualize software release histories using color and third dimension
- Case Study: Telecommunication Switching System

TSS case study structure



The Release Database

- For each release stored:
  - Entries for elements at system, subsystem, module, and program level together with relations among them
  - Systems and programs are characterized by version numbers
  - Program version numbers are independent of the system's version number
  - Changes result in incremented version number(s)
- Each system release consists of
  - 8 subsystems, 47 to 49 modules, and 1500 to 2300 programs.

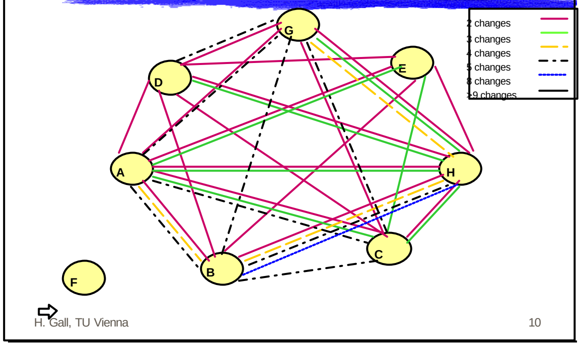
SEA: Detection of logical Coupling

- Change Sequence of a program <1 2 3 5 7 11>
  - program changed in releases 1, 2, 3, 5, 7, and 11
  - 5 changes
- Subsequences as contiguous parts
  - <1 2 3>, <3 5 7>, etc.
- Changes are represented by a (sub-) sequence
- Identify potential 'logical couplings" among programs

SEA: Change Sequence Analysis

- Approach:
  - compare change sequences of different modules
  - identify patterns of change
  - identify common “change sequences” (patterns)
- Result: potential logical couplings

Coupling among subsystems



SEA: Change Report Analysis

- Approach:
  - verify logical coupling
  - examine change reports of modules with the same change sequence
  - same reason for change defines logical coupling
- Result: logical couplings among modules / subsystems

Example of a change report

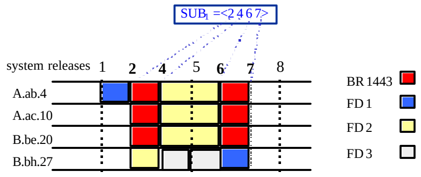
Ver 2.4 — 96/03/12 10:10:07
TSS---PROGRAM CHANGE DESCRIPTION

ELEMENT NAME: Program 111 **2.3 --> 2.4**
CHANGED BY: John DOE
CHANGES as follows:

CHANGE NR: 1
CHANGE TYPE: B // **bug fix**
REFERENCE: BR 1443 // reference to a **bug report number**
ERROR CLASS: A // **error class**, i.e. operation in working state
DESCRIPTION: hanging of the circuits in environment xy.

CHANGE NR: 2
...

Change Reports Analysis



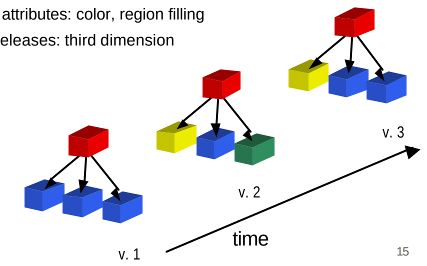
BR ... bug report
FD ... further development

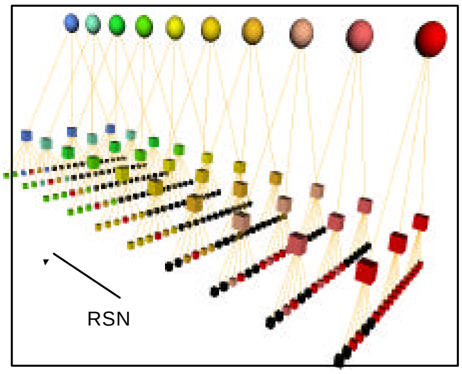
Results

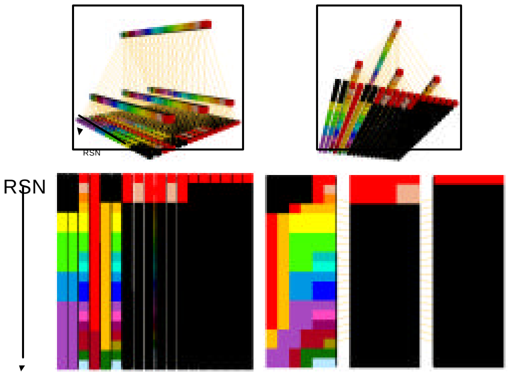
- Identified those modules and programs that should undergo restructuring / reengineering
- TSS: Constantly growing Subsystem C
  - not reflected in a high logical coupling with other subsystems
  - restructuring is “local” within Subsystem C and its modules
  - structural shortcoming is local in terms of subsystems, but high module interrelationships
- We detect stronger logical couplings via longer sequences

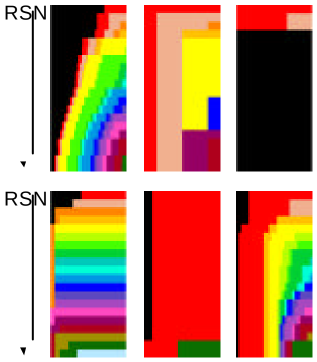
Visualization

- Structure of the system: visualization of tree structure (2-D and 3-D)
- Software attributes: color, region filling
- Multiple releases: third dimension

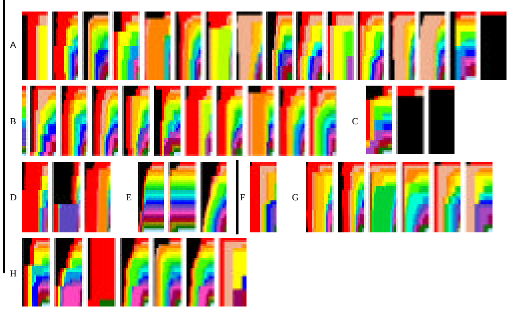








TSS visualized



Conclusions

- Developed techniques to investigate very large software systems on a macro-level to detect
  - structural shortcomings
  - dependencies among modules, change patterns
- Database for product releases
  - data are relatively easy to obtain
  - valuable additional information
    - kind of change (corrective, adaptive, perfective, preventive)
    - correlation of behavior and kinds of changes
- Such retrospective analysis is valuable complement to code-based predictive analysis

CAFÉ: Objectives

- Investigate product family evolution
  - investigate relationship between feature set evolution and product family evolution
  - investigate platform evolution in terms of integration of product-specific features
- Visualize product family evolution
  - investigate the evolution of feature sets in terms of their relationship to components and structures
  - visualize different aspects such as time, structure, feature sets, tasks, styles, patterns etc.