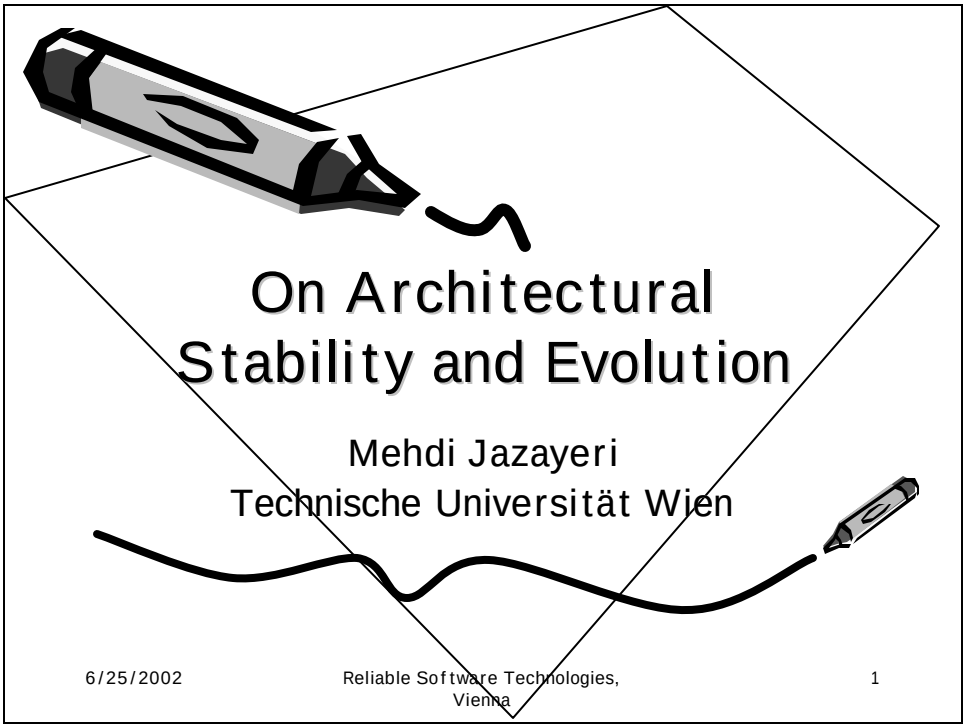


On Architectural Stability and Evolution

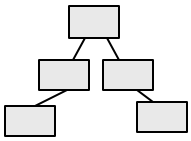
Mehdi Jazayeri
Technische Universität Wien

Outline

- The problem: how to evolve software
- Current ideas: product family, cbse, ..
- Old ideas: information hiding, …
- Case studies of software evolution
- Role of tools: how to remember
- Conclusions and questions

Software Evolution

- Importance
- How to help it in *architecture*
- How to check we did it right?



Architecture of …?

Influential "recent" papers on software architecture

- Perry and Wolf [92]: framework for study
- Shaw and Garlan [96]: classification
- Kruchten [96]: 4+1 Model

All deal with structure of software…

Seminal papers by Parnas [70s and 80s!]

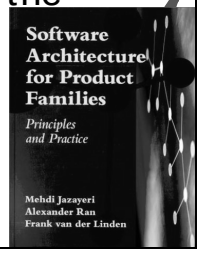
- Information hiding
- Uses relation
- Design for change
- Ease of extension and contraction
- Program families

All address change and evolution…

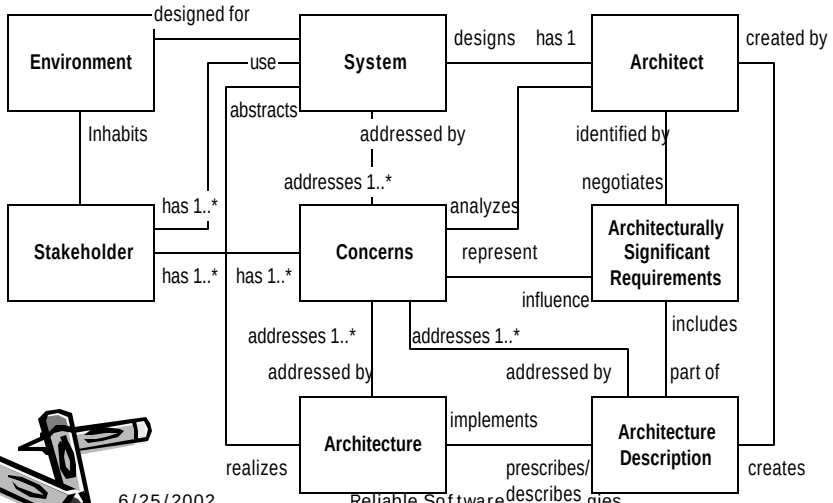
What Is Software Architecture? (3)

Software architecture is a set of *concepts* and design decisions about *structure* and texture of software that must be made prior to *concurrent engineering* to enable effective satisfaction of *architecturally significant*, explicit functional and quality requirements, and implicit *requirements* of the product family, the problem, and the solution domains.

[ARES project, 2000]



Context of Architecting

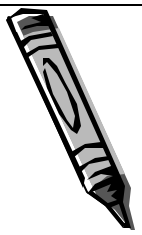


Importance of software architect

"Conceptual integrity is central to product quality. Having a system architect is the most important single step toward conceptual integrity."

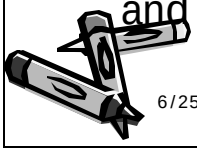
Fred Brooks, The Mythical Man-Month, 1995

Importance of software architecture

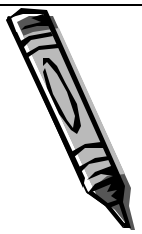


"If a project has not achieved a system architecture, including its rationale, the project should not proceed to full-scale system development. Specifying the architecture as a deliverable enables its use throughout the development and maintenance process."

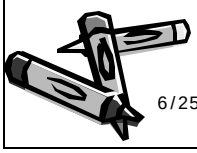
Barry Boehm, 1995



Architecture evaluation



- Identify the goals
- Predictive evaluation
  - Inspections and reviews
  - Process (e.g. SAAM, ATAM)
- Retrospective analysis
  - Process
  - Tools (e.g. visualization)

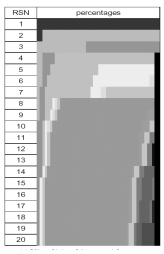


Predictive vs Retrospective Analysis

- Evaluate a recipe or the resulting dish
  - Taste Topfentorte or read its recipe ☺
- Architecture
  - as-planned versus as-implemented
  - analytical versus empirical

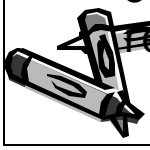
Retrospective analysis: How?

- Maintain a history of releases
- Visualize the history to see evolution patterns



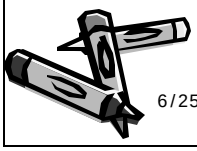
History from the project database

- Many companies keep track of projects
- Data is maintained in database to support management decisions
- Examples: duration of project, effort estimates, defect data, …
- Create a database for product releases



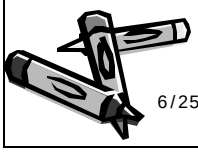
Visualization

- Large amounts of data is useful but unwieldy
- Patterns can be seen visually
- Many scientific visualization techniques are available



Case Study: Telecommunication System

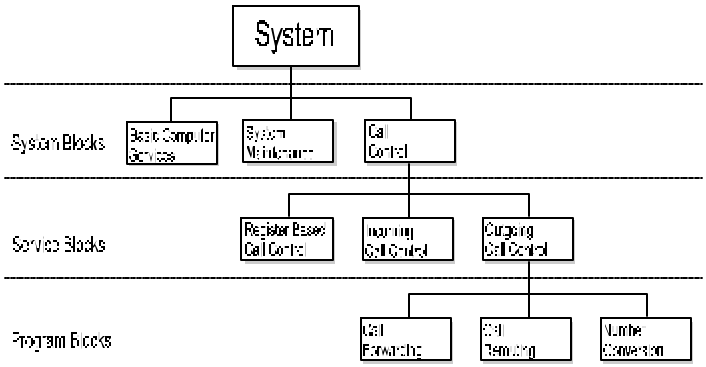
- TSS system in use for many years
- Contains 10M lines of code
- Database *about* the software modules
- Data about 20 successive releases



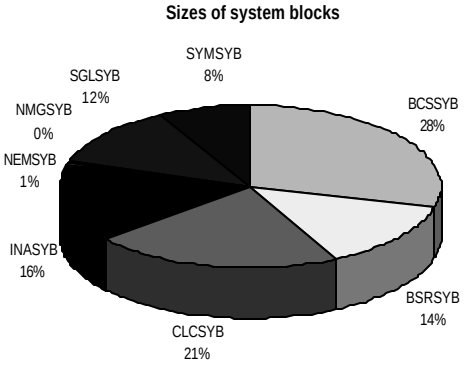
Architectural stability properties

- Average and distribution of module sizes?
- How are modules related?
- Are module sizes growing or shrinking?
- Are particular modules "hot spots" for change?
- How do these properties change in different releases?

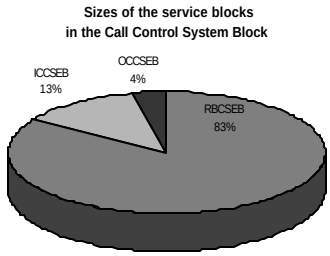
The TSS System Architecture



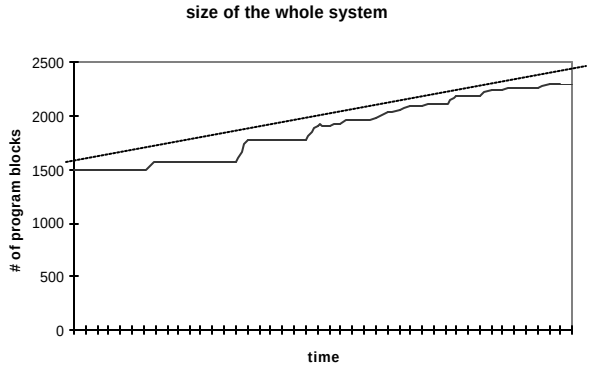
Sizes of system blocks



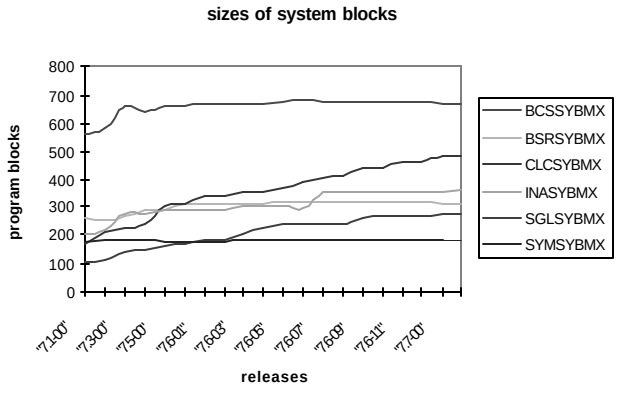
Sizes of service blocks in the Call Control System Block



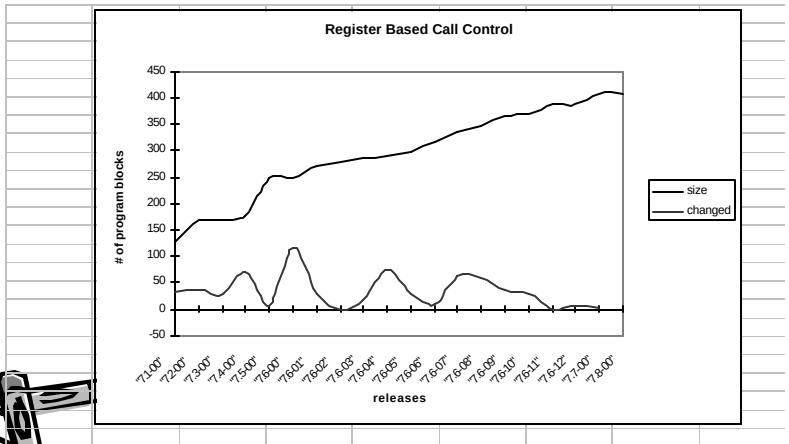
Size of the whole system



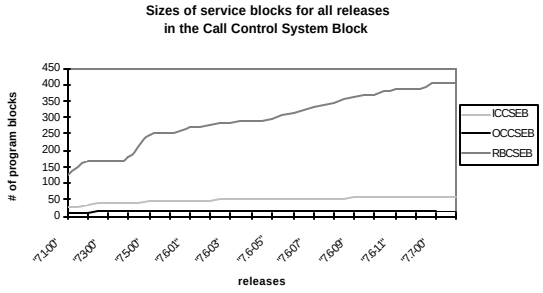
Sizes of System Blocks



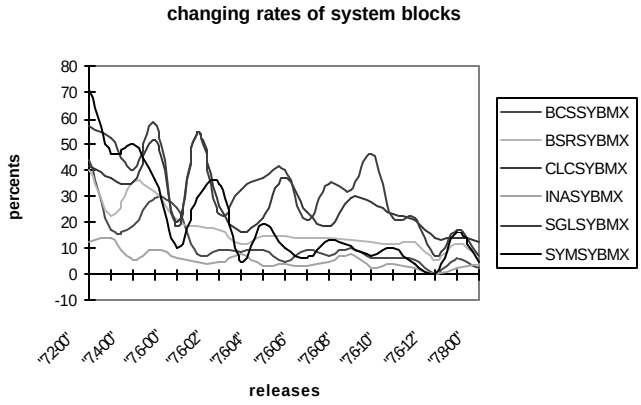
Size of the Call Control Block



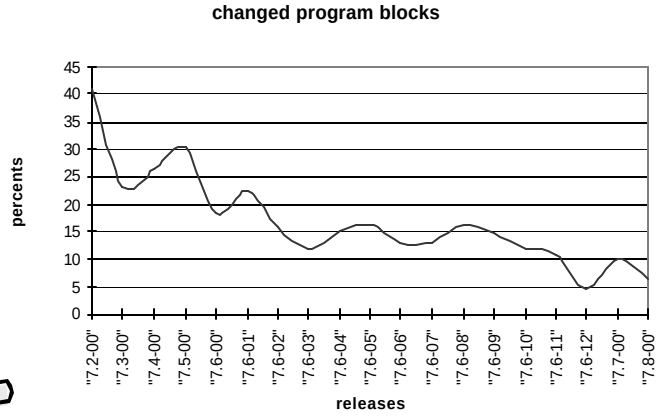
Sizes of service blocks in the Call Control System Block

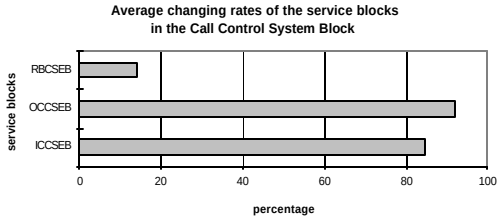


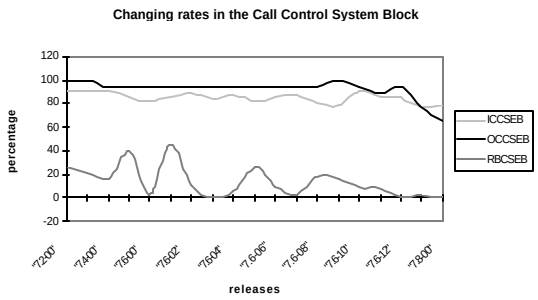
Changing Rates of System Blocks

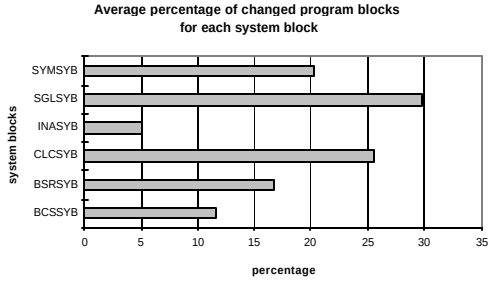


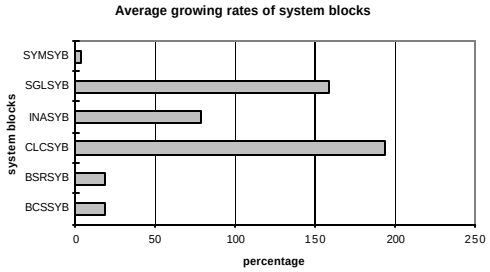
Changed Program Blocks

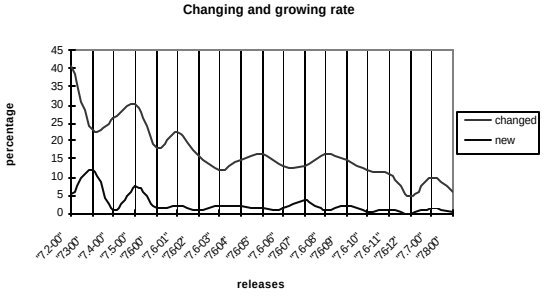




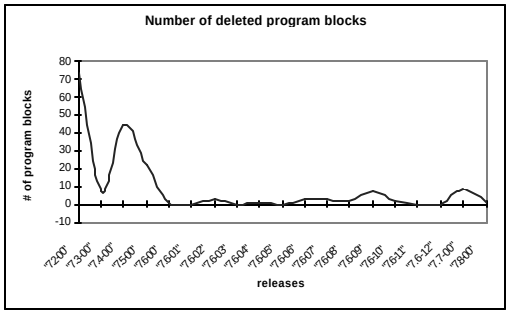

Changing rates in the Call Control System Block
120
100
80
60
40
20
0
-20
percentage
ICCSEB
OCCSEB
RBCSEB
'7200'
'7400'
'7600'
'7602'
'7604'
'7606'
releases


Average percentage of changed program blocks for each system block
SYMSYB
SGLSYB
INASYB
CLCSYB
BSRSYB
BCSSYB
system blocks
0
5
10
15
20
25
30
35
percentage

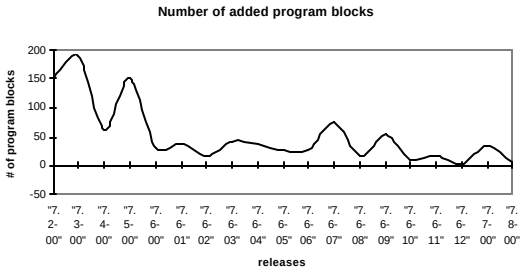

Average growing rates of system blocks
SYMSYB
SGLSYB
INASYB
CLCSYB
BSRSYB
BCSSYB
system blocks
0
50
100
150
200
250
percentage


Changing and growing rate
percentage
45
40
35
30
25
20
15
10
5
0
changed
new
releases

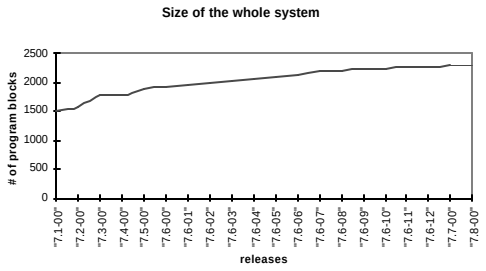
Number of deleted program blocks



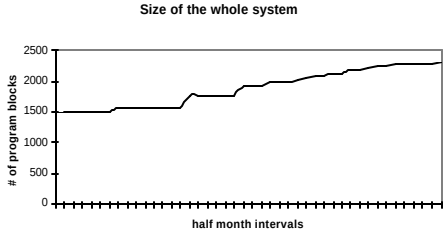
Number of added program blocks



Size of the whole system



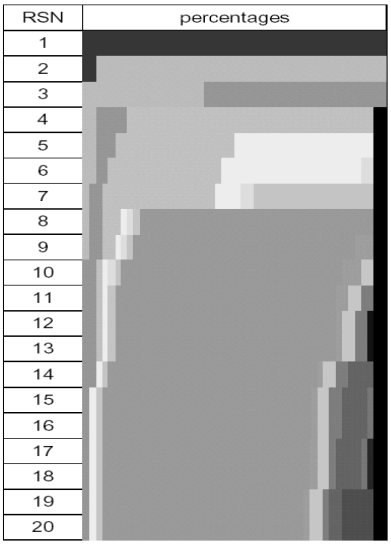
Size of the whole system (in six month increments)



Use of color: summarization

- Can we summarize evolution data in one “picture”
- Is color useful?

Visualizing releases with color and percentage bars



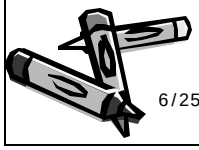
Program Evolution In modules A through H

Interpretations

- Visualization analysis can be applied at different levels
- We can see evolution of individual programs, program blocks, modules, and whole system
- Patterns can be perceived quickly, leading to qualitative trends

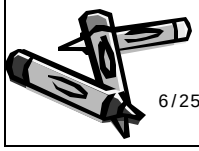
Lanza's observation of class evolution

- Pulsar: grows and shrinks repeatedly
- Supernova: sudden size explosion
- White dwarf: shrinks, and shrinks, …
- Idle: does not change over releases

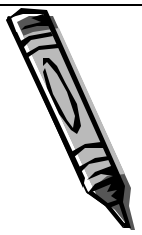


Observations about TSS case study

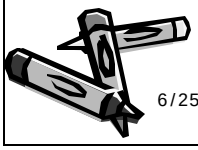
- Software system *as a whole* has stabilized over time
- Some modules still undergoing considerable change
- Releases can be of varying significance
- Some releases look unusually "turbulent"
- Architecture has not been "stable"



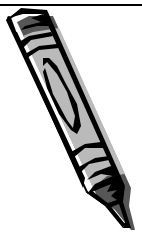
Another case study: Mozilla



- Open-source browser software
- Will we see similar trends?
- Use “Motion Video”
- Disclaimers:
  - experiment just starting
  - very, very, preliminary data

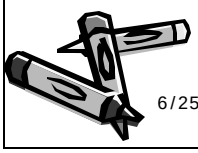


43 monthly revisions: 1998-03-31 - 2002-03-05



83 files

23603 files



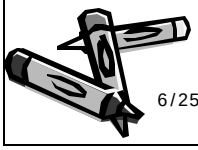
Evolution by module (directories)

Observations about Mozilla evolution

- Growing…
- Adding “new stuff”
- Not stabilized yet
- Looks different from TSS
- Architecture is “stable”?

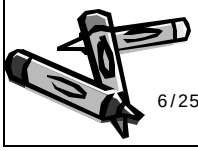
Speculations

- Evolution goes with any software
- Good architecture helps evolution
- Web software will be more volatile
- Traditional software should be "stable"



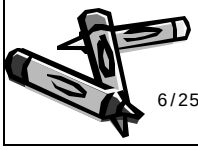
Requirements

- Recent experience: 60% of requirements had to be changed!
- Requirements evolution?



Conclusions

- The architect's problem is software *evolution*
- Architectural "evolvability" should be validated *retrospectively*
- Tools must track software evolution: i.e. *span* several releases
- It is not hard to do ☺



Post-conclusions

- Will component-based development change the rules of the game?
  - Maintenance problems go away: we just unplug and insert new component
- Are the requirements of Internet-based software different?
  - Applications evolve by negotiating on-line for the right component to adopt