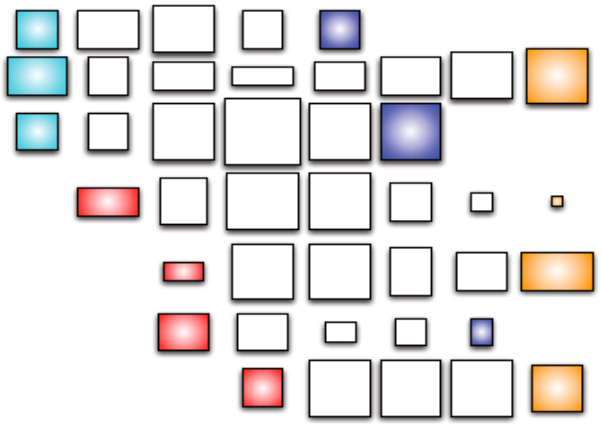
Software Evolution Analysis and Visualization

Harald Gall
University of Zurich
Department of Informatics
http://seal.ifi.uzh.ch

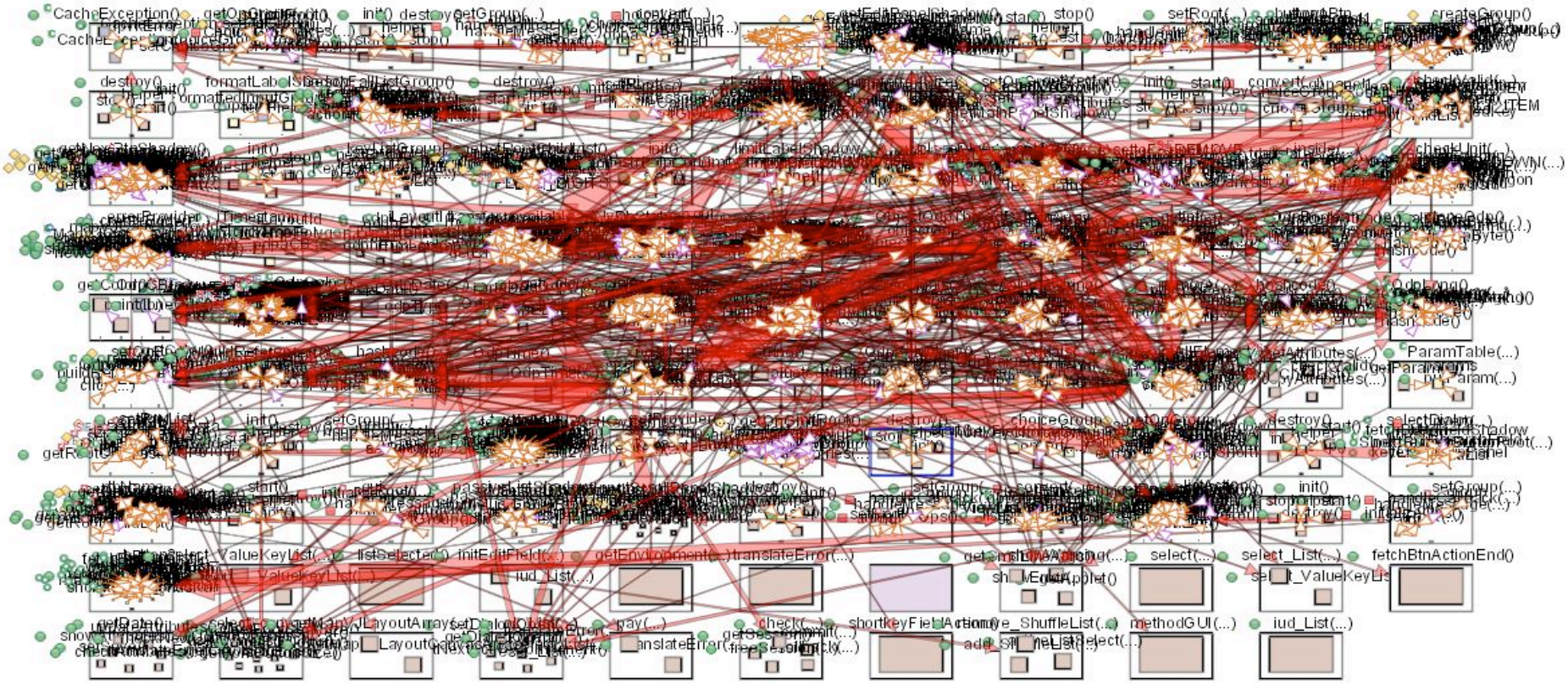


s.e.a.l.
software evolution & architecture lab

"The study of products is vastly more important than the study of production, even for understanding production and its methods."

Karl Popper

On the study of products...



Goals & Questions

- What can we learn about
  - Software and its structure
  - Change impact and propagation
  - Developer contributions and efforts
  - Team structure and social networks
  - Change smells, trends and hot spots
  - Faults and defects

Mining Software Repositories...

- Code base
  - Which entities co-evolve?
  - Do code and comments co-evolve?
- Bugs and Changes
  - Who should fix this bug?
  - How long will it take to fix this bug?
  - Predicting bugs from cached bug history
  - When do changes induce fixes?
- Project and Process
  - Project memory for software development
- Software Expertise
  - Identifying expertise from changes and bug reports

Techniques

Standard Quality Characteristics

Software Quality Models

Software Process

Software Analysis

Software Visualization

Reverse Engineering

Software Analysis

Developer Patterns

Feature Analysis

Architecture Reflexion

Software Evolution Analysis

Software Evolution Metrics

Code Duplication Analysis

Reengineering Patterns

Software Artifacts Analysis

Analyses & Visualizations

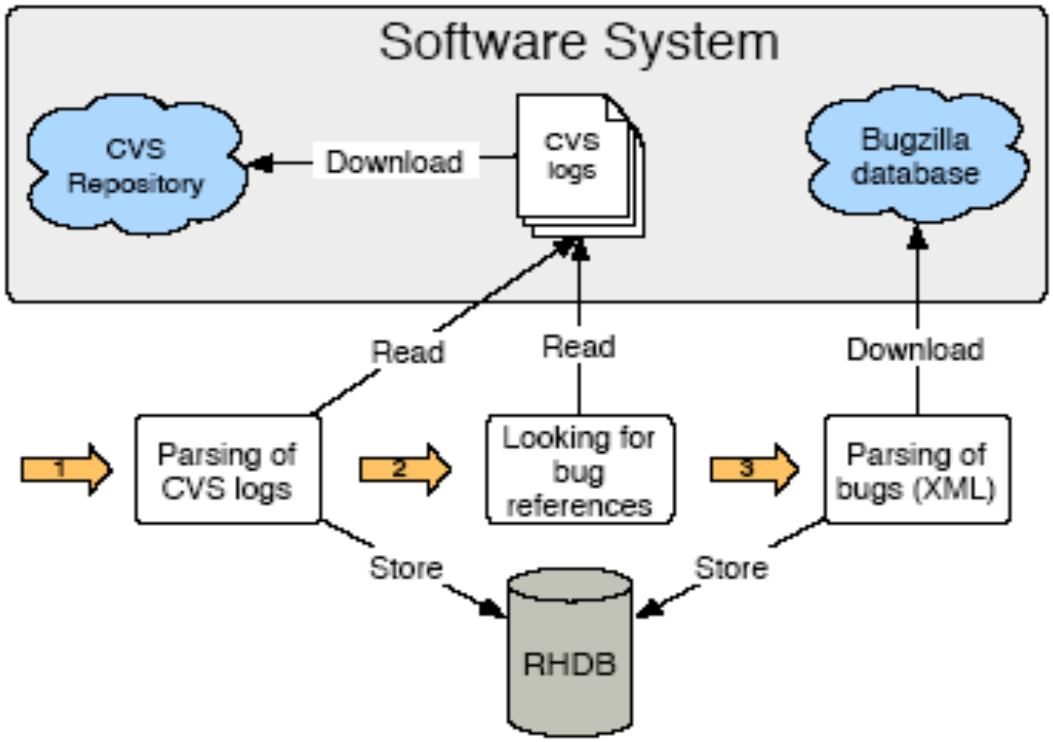
Changes and bug fixes

Developer tasks & patterns

Social networks

s.e.a.l.
software evolution & architecture lab

Release History Database



Related work

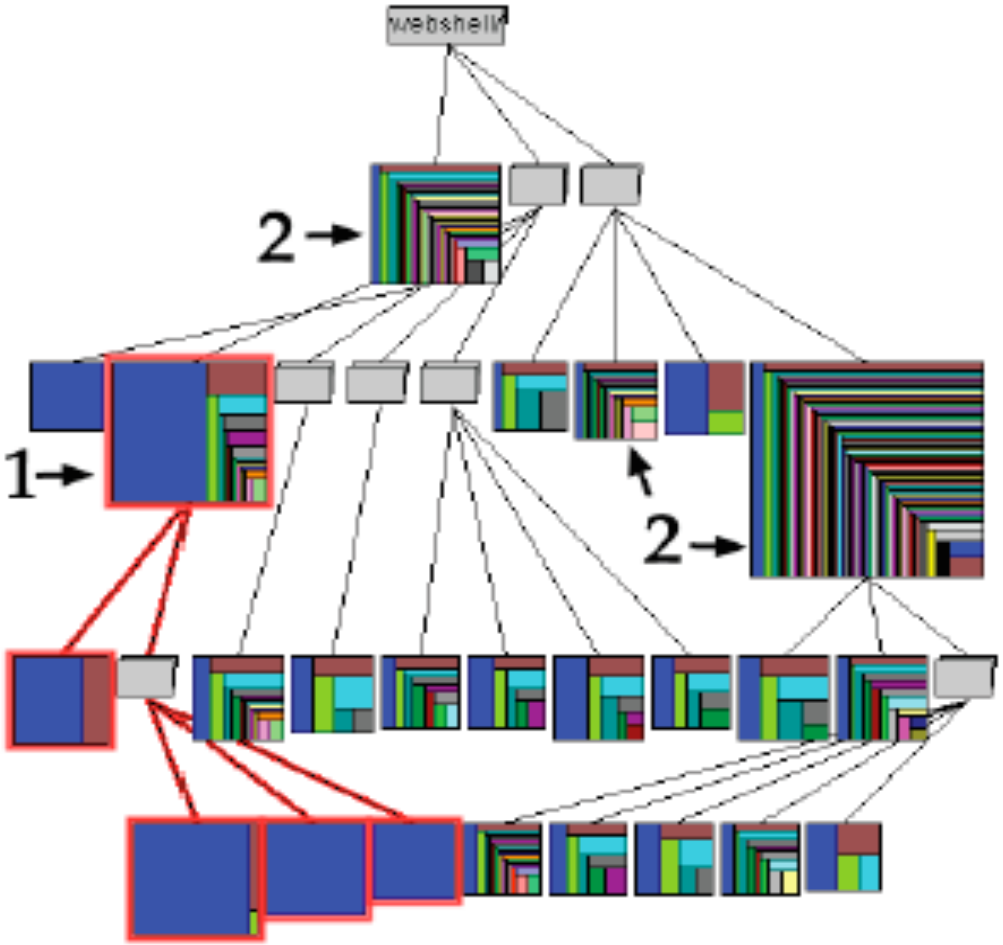
- Hipikat, Cubranic et al.
- softChange, German
- Kenyon, Bevan et al.
- s.e.a.l. Evolizer, Gall et al.

The *gestalt* of Fractal Figures

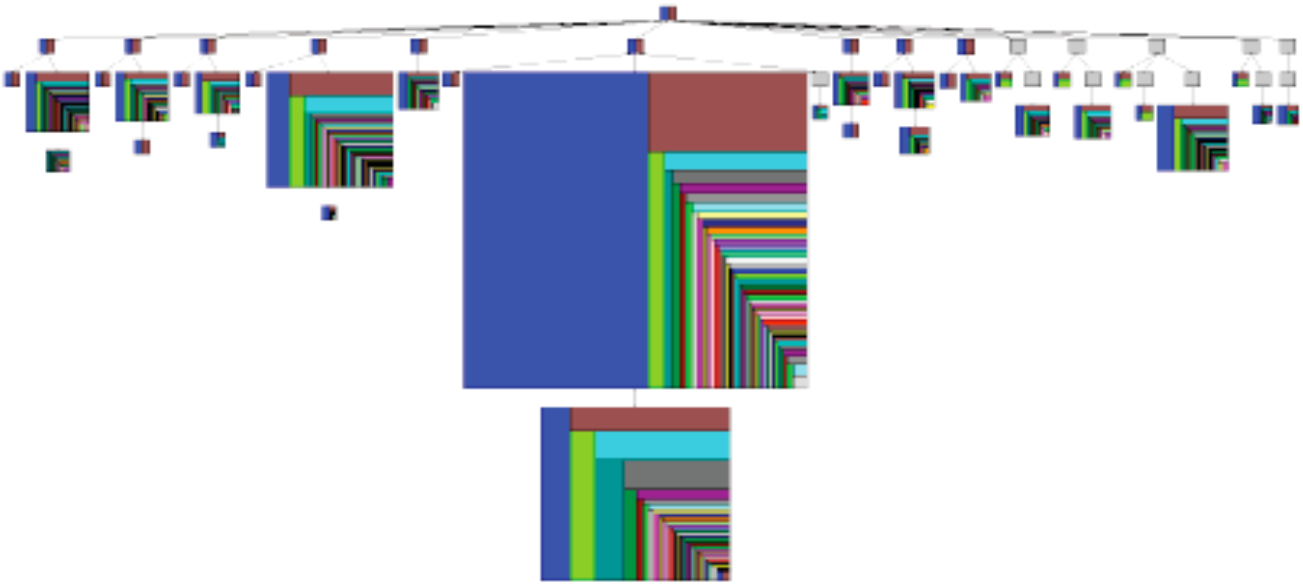
(a) One developer (b) Few balanced developers (c) One major developer (d) Many balanced developers

$$\text{Fractal Value} = 1 - \sum_{a_i \in A} \left(\frac{nc(a_i)}{NC} \right)^2, \quad \text{with} \quad NC = \sum_{a_i \in A} nc(a_i)$$

How many developers per entity?



How many bugs per entity?



Marco D'Ambros, Michele Lanza and Harald C. Gall, Fractal Figures: Visualizing Development Effort for CVS Entities
In Proceedings of International Workshop on Visualizing Software For Understanding and Analysis, 2005.

Who should fix this bug?

- Apply machine learning algorithms to open bug repository
- Learn the kinds of reports that each developer resolves
- A classifier suggests developers who should resolve the bug
- Precision: 57% in Eclipse, 75% in Firefox

Anvik, J., Hiew, L., and Murphy, G. C. 2006. *Who should fix this bug?*
In Proceeding of the 28th international Conference on Software Engineering, May 20 - 28, 2006.

How long will it take to fix this bug?

- Automatically predicting the fixing effort, i.e., the person-hours spent on fixing an issue
- Effort data from JBoss project
- Quality of predictions
  - issues: close to actual effort
  - bugs: beating naive predictions

Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, Andreas Zeller, *How Long Will It Take to Fix This Bug?* In Proceedings of the Fourth International Workshop on Mining Software Repositories, May, 2007.

When do changes induce fixes?

- Fix-inducing changes
- Which change properties may lead to problems?
- How error-prone is my product?
- How can I filter out problematic changes?
- Can I improve guidance along related changes?
- --> Fridays (Eclipse) or Sundays (Mozilla)

	Day of Week							
% of revisions	Mon	Tue	Wed	Thu	Fri	Sat	Sun	avg
$P(fix)$	18.4	20.9	20.0	22.3	24.0	14.7	16.9	20.8
$P(bug)$	11.3	10.4	11.1	12.1	12.2	11.7	11.6	11.4
$P(bug \cap fix)$	4.6	4.8	4.6	5.2	5.6	4.5	4.5	4.9
$P(\neg bug \cap \neg fix)$	74.9	73.5	73.5	70.8	63.4	78.1	76.0	72.7
$P(bug \mid fix)$	25.1	22.9	23.3	23.5	23.2	30.3	26.4	23.7
$P(bug \mid \neg fix)$	8.2	7.1	8.1	8.8	8.7	8.4	8.6	8.1

Table 5: Distribution of fixes and fix-inducing changes across day of week in ECLIPSE

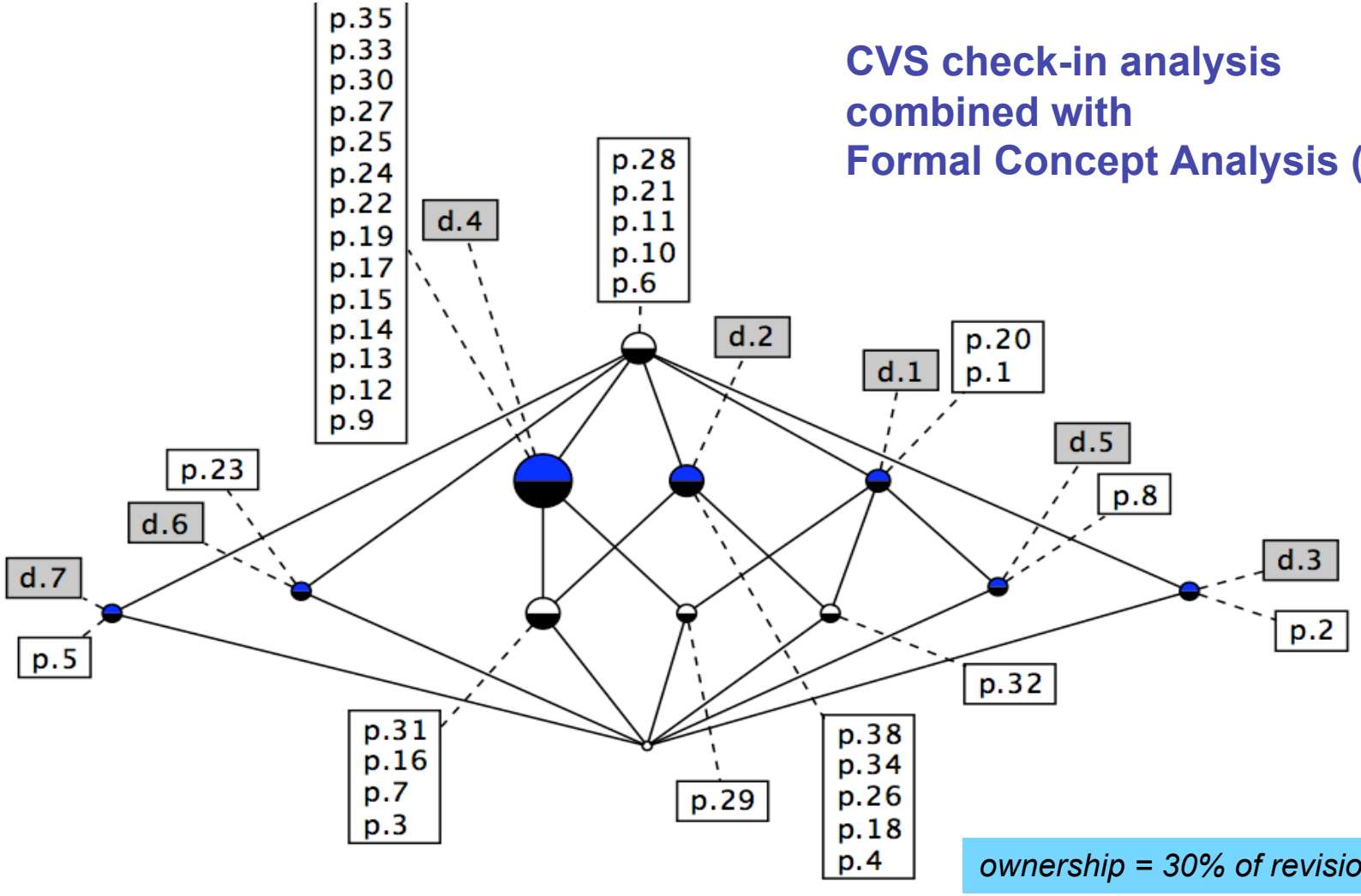
	Day of Week							
% of revisions	Mon	Tue	Wed	Thu	Fri	Sat	Sun	avg
$P(fix)$	42.5	46.5	49.7	45.9	48.4	50.2	61.1	48.5
$P(bug)$	39.1	44.1	41.2	40.8	46.2	44.9	26.4	41.5
$P(bug \cap fix)$	19.4	23.6	22.8	21.6	26.9	19.6	13.2	21.9
$P(\neg bug \cap \neg fix)$	37.8	33.0	31.9	34.9	32.3	24.5	25.7	31.9
$P(bug \mid fix)$	45.7	50.8	45.8	47.1	55.6	39.1	21.6	45.2
$P(bug \mid \neg fix)$	34.1	38.3	36.7	35.5	37.3	50.6	33.9	38.1

Table 6: Distribution of fixes and fix-inducing changes across day of week in MOZILLA

Śliwerski, J., Zimmermann, T., and Zeller, A. 2005. *When do changes induce fixes?* In Proceedings of the 2005 international Workshop on Mining Software Repositories, St. Louis, Missouri, May 2005.

Code Ownership & Co-Evolution

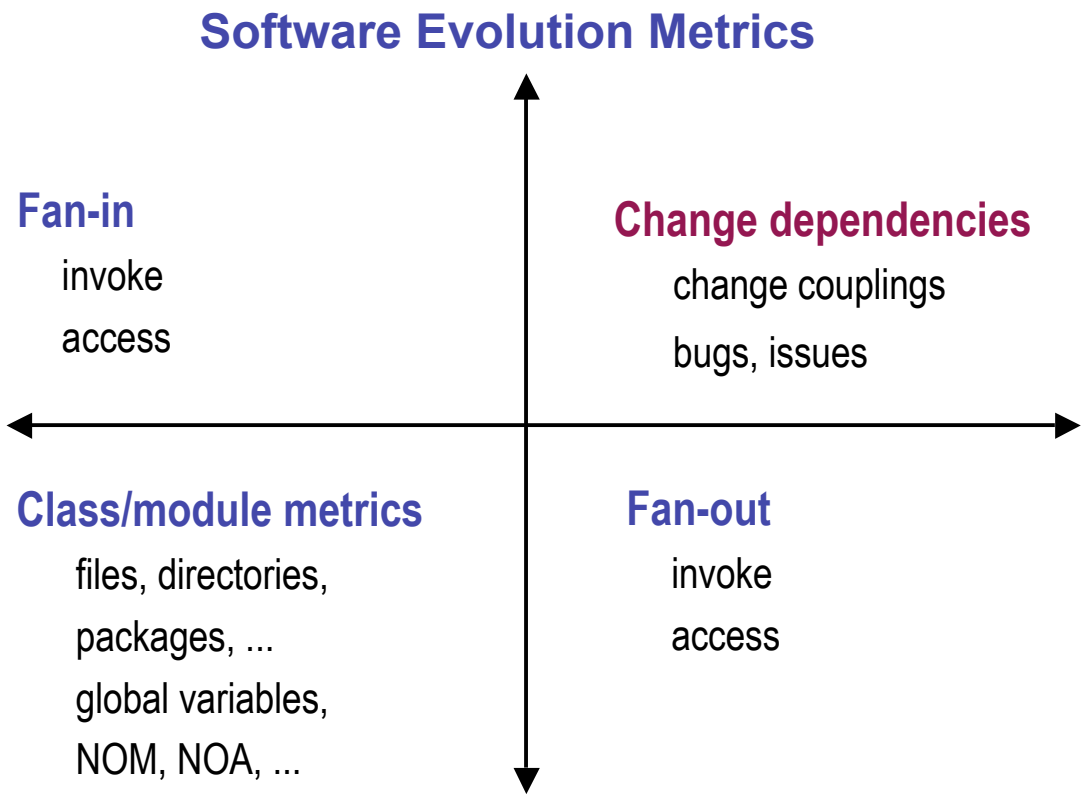
Who is the code owner?



CVS check-in analysis combined with Formal Concept Analysis (FCA)

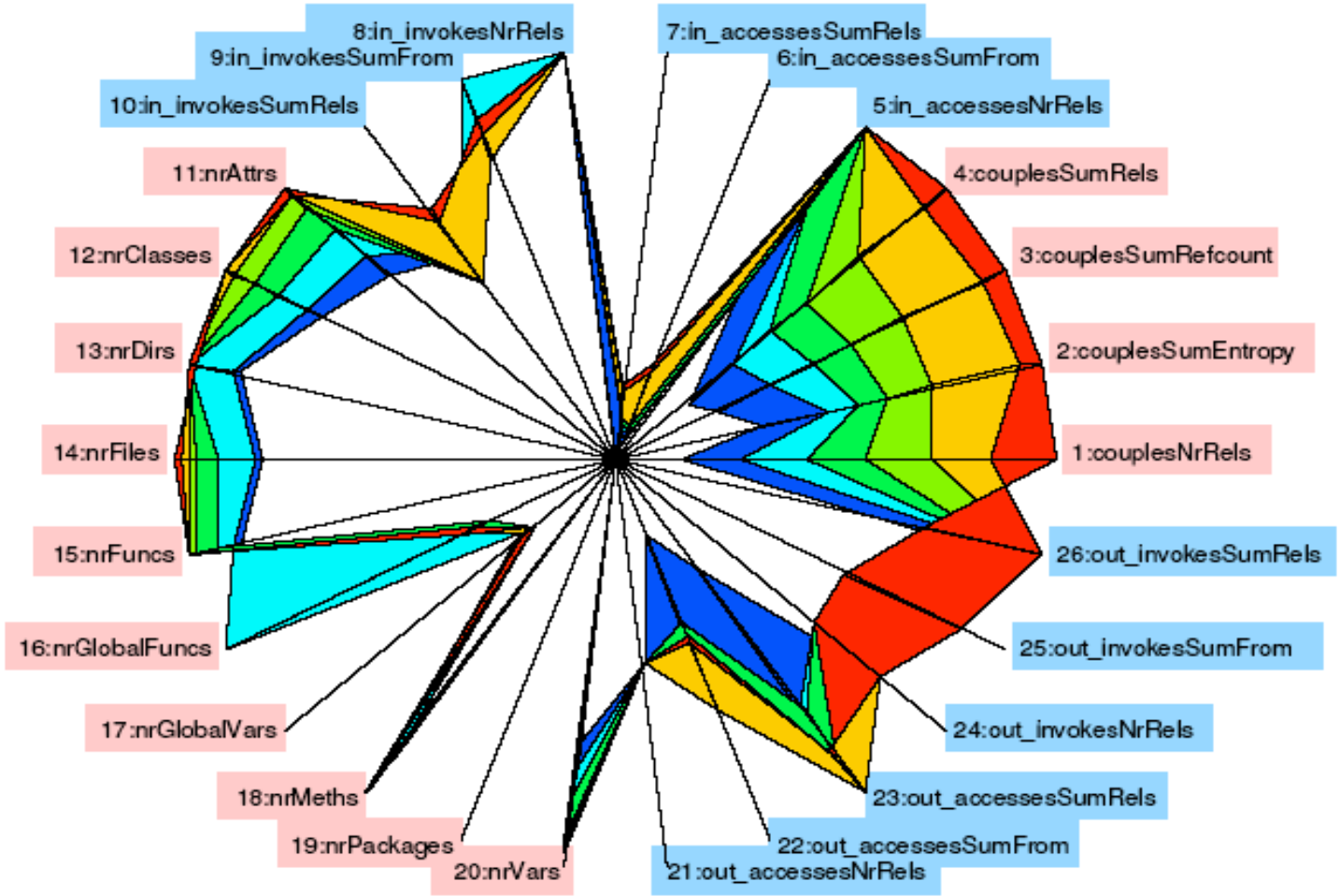
ownership = 30% of revisions

Which entities co-evolve?



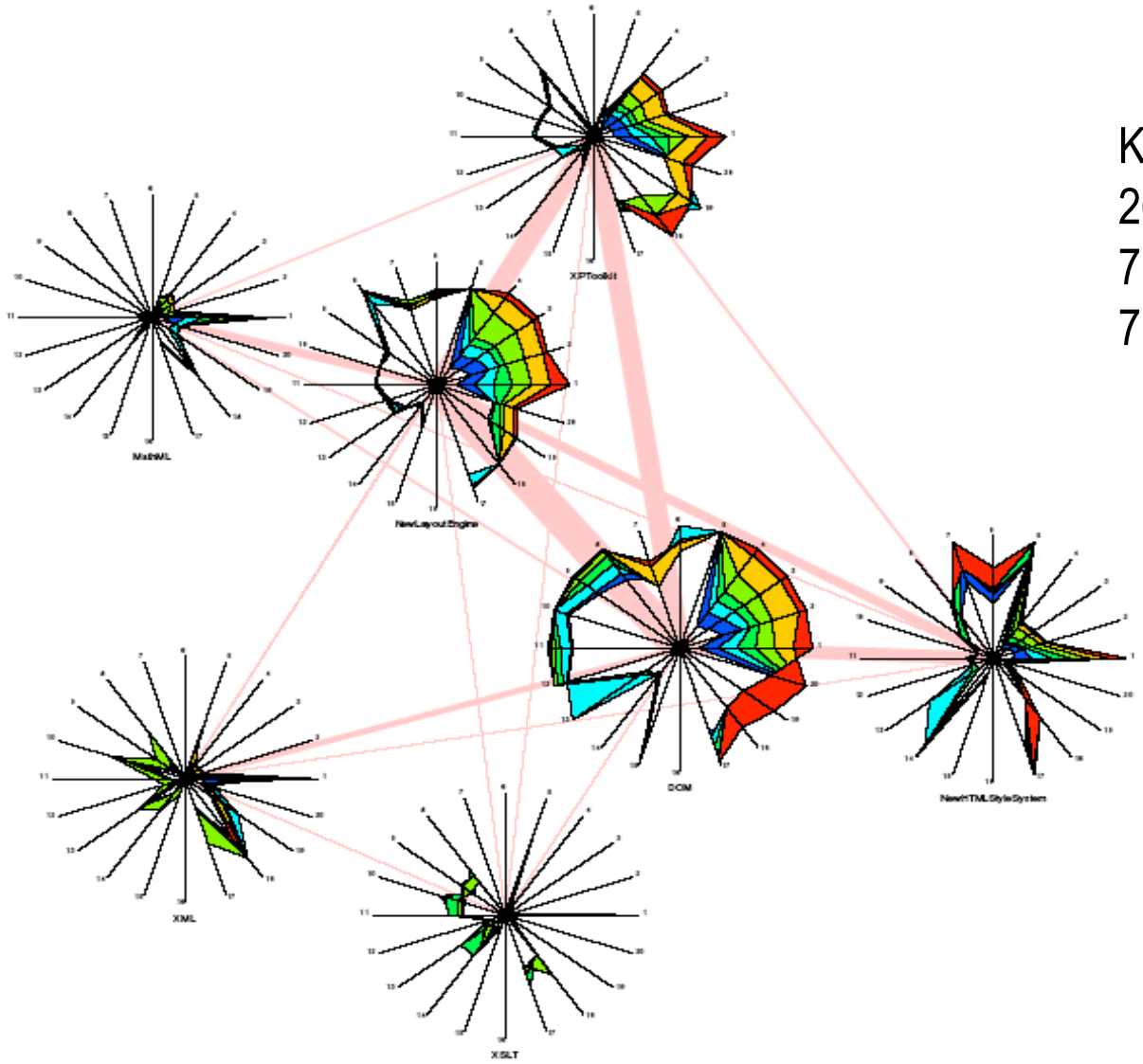
Martin Pinzger, Harald C. Gall, Michael Fischer, and Michele Lanza, *Visualizing Multiple Evolution Metrics*
In Proceedings of the ACM Symposium on Software Visualization, 2005.

Mozilla Module DOM: 0.92 -> 1.7



DOM

Multiple Evolution Metrics



Kiviat graph:
26 metrics
7 Mozilla modules
7 subsequent releases

ChangeDistilling

Source Code Change Extraction

s.e.a.l.
software evolution & architecture lab

Change Analysis

- Current change history analysis rely on versioning systems (e.g., CVS)
- Extracting source code changes by means of text diffs has problems
  - determine enclosing entity (e.g., method)
  - kind of statement which changed (e.g., return statement)
  - kind of change (i.e., insert, delete, move, update)

Change Analysis

CVS diff

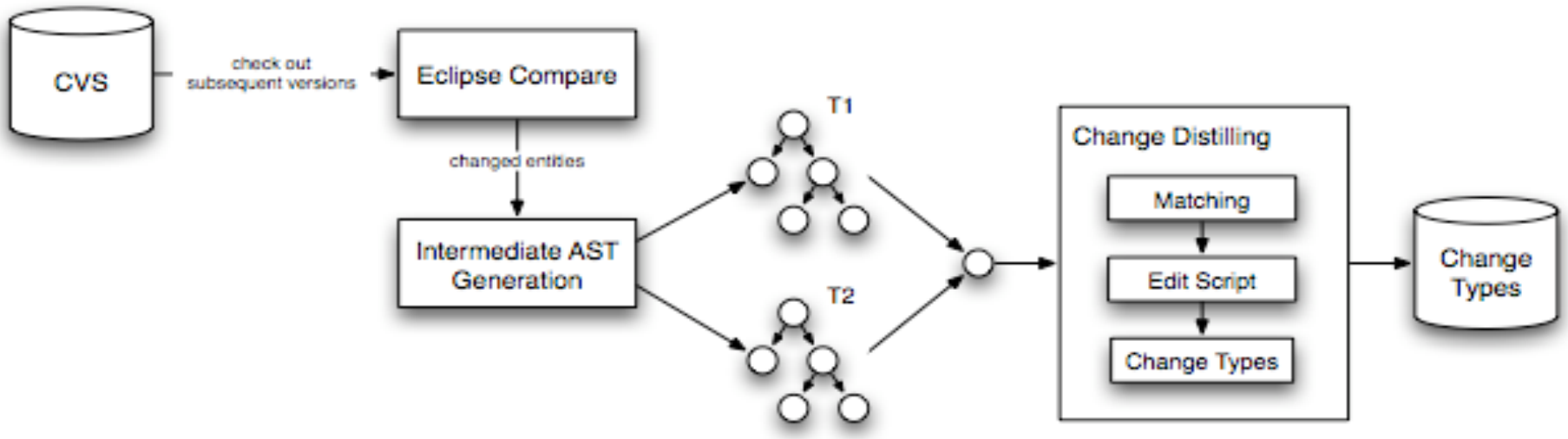
```
1967,1970c1964,1965
<     if (d != null) {
<         d.foo();
<         d.bar();
<     }
---
>     d.foo();
>     d.bar();
```

- 3 Body changes
- 2 Statement parent changes
- 1 Statement delete
- Change significance?

CVS log: “lines: +2 -4”

Change Distilling

- Identifying change types and change patterns
- Eliminate mass changes and other noise



Examples of Change Types

- Classification of 35 change types

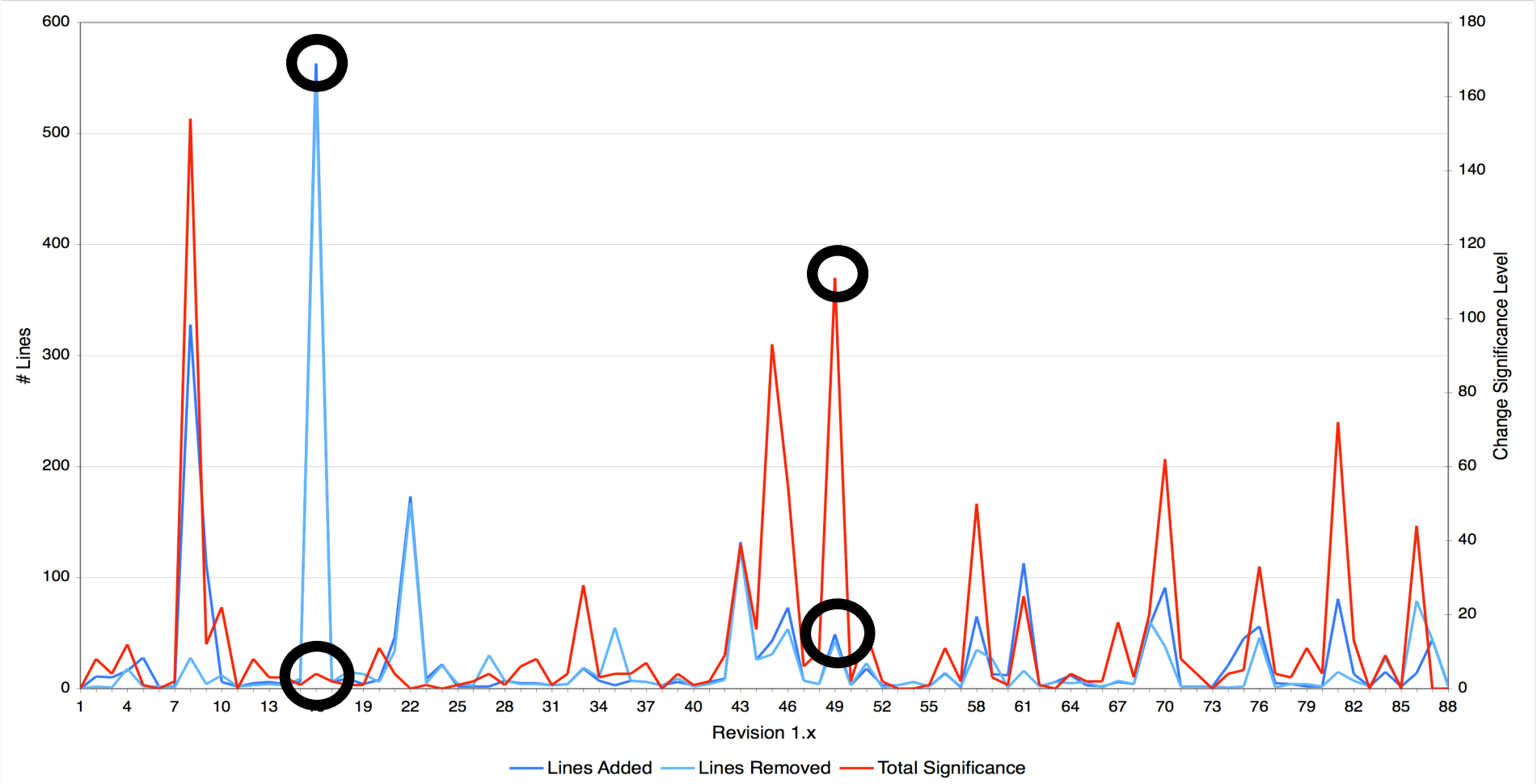
Body part changes:	***significance:***
Additional Object State	low
Condition Expression Change	medium
Removed Functionality	crucial

Declaration part changes:	***significance:***
Final Modifier Delete	low
Parameter Renaming	medium
Return Type Update	crucial

Potential of Change Type Analysis

- Stability of interfaces
- Change impact
- Code and Comments
- Changes due to bug fixes
- Many or significant changes

Which changes are significant?



Example from ArgoUML

Developer Networks

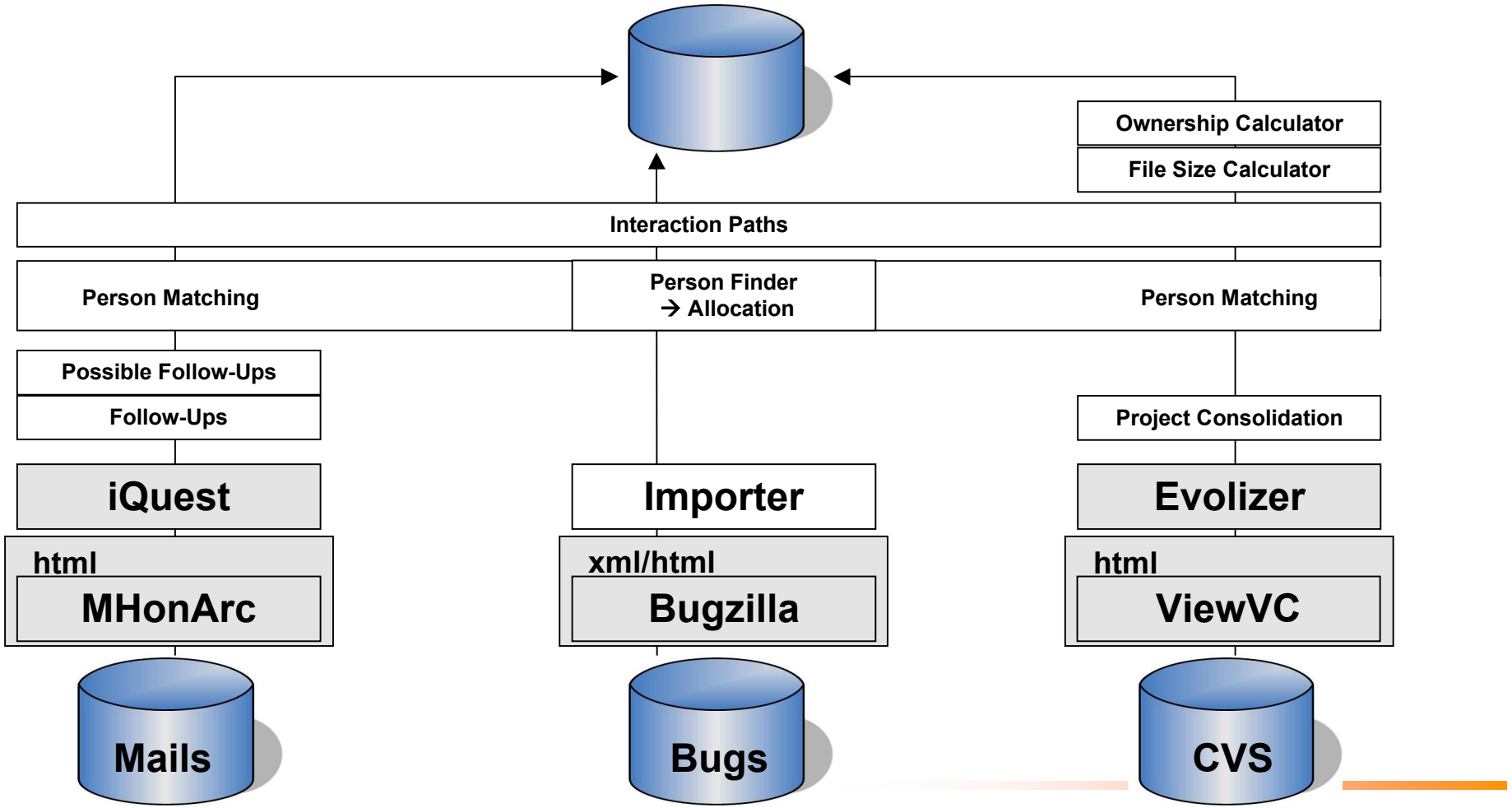
Communication Structures in
Software Teams

s.e.a.l.
software evolution & architecture lab

What are the developer networks?

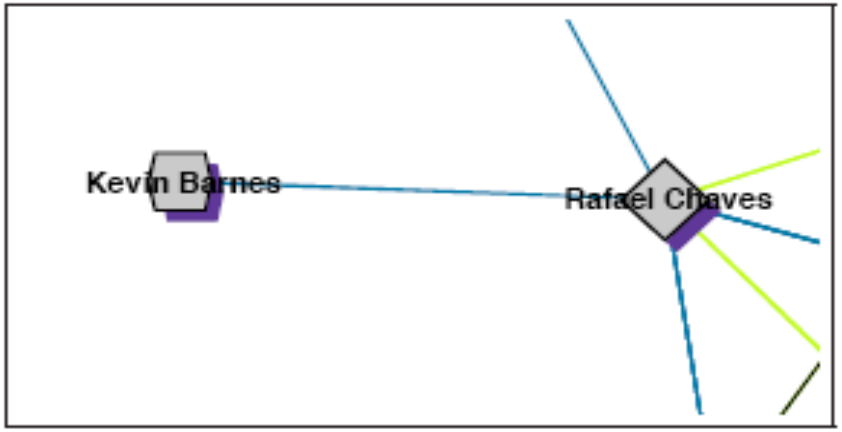
- Conway‘s law
- Inter-team collaboration
- Ownership changes
- Key personalities in social networks
  - *connectors vs. communicators*
  - *gatekeepers, influencers, innovators, leaders and communicators as trendsetters*
- Information for project manager vs. newcomer

Integration of Data Sources



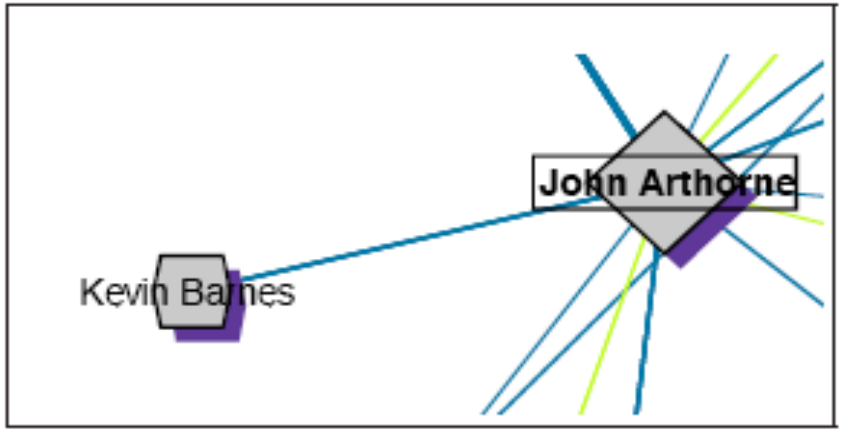
Scenario: newcomer Kevin

(a)



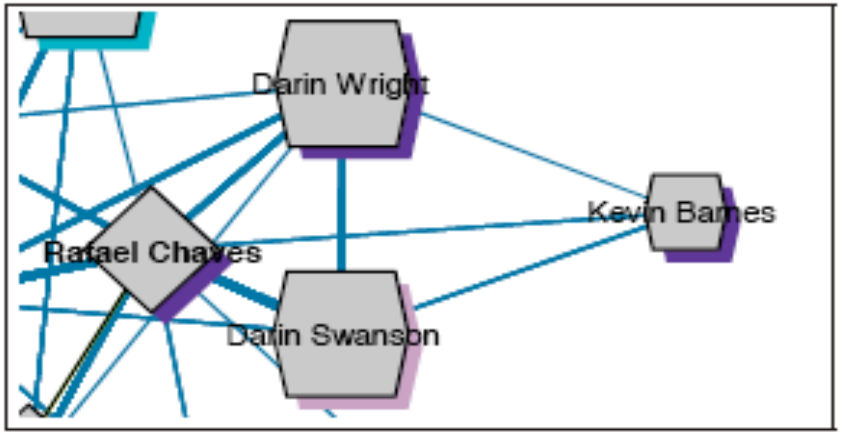
April 2004

(b)



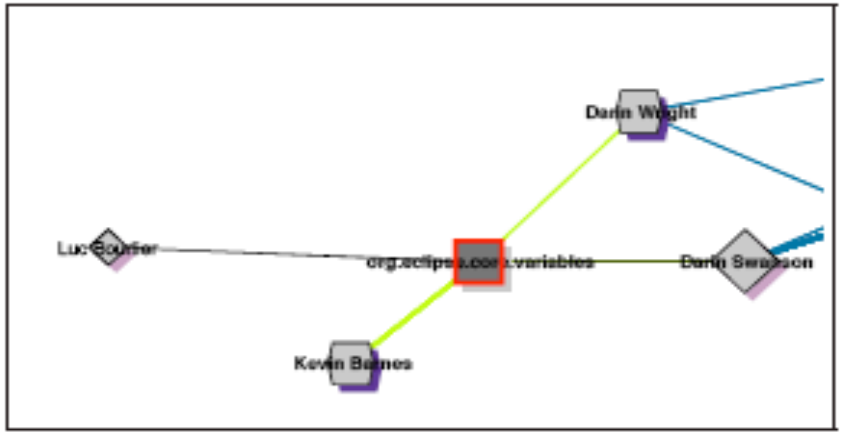
April 2004

(c)



June 2004

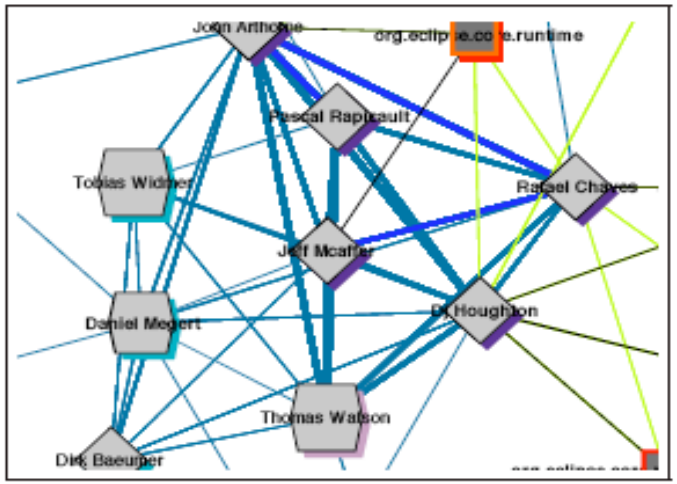
(d)



June 2004

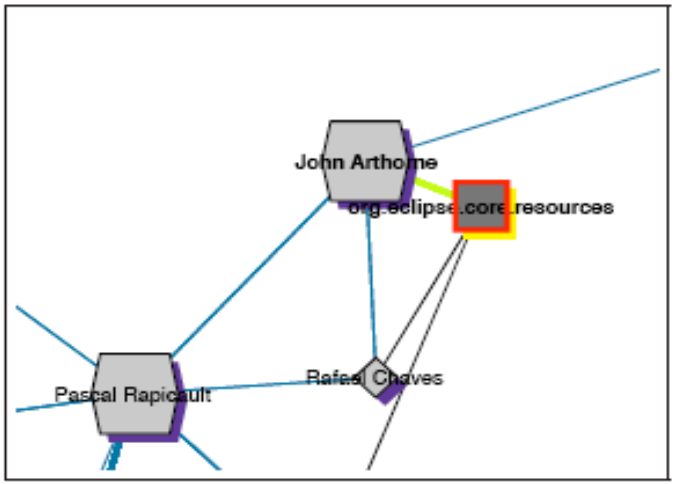
Scenario: key person Rafael leaving

(a)



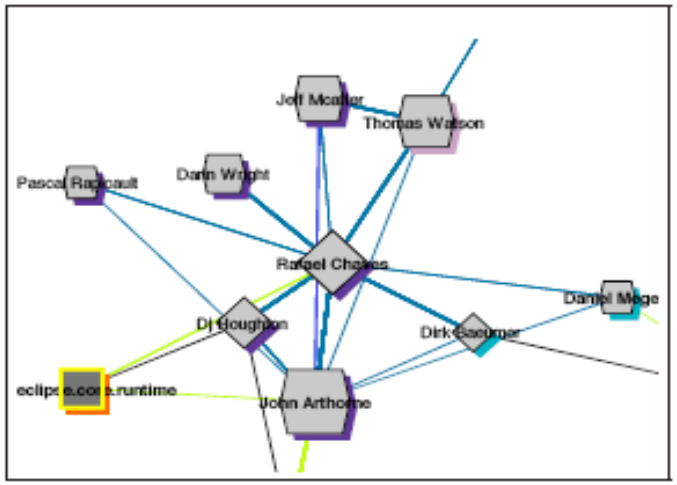
April 2005

(b)



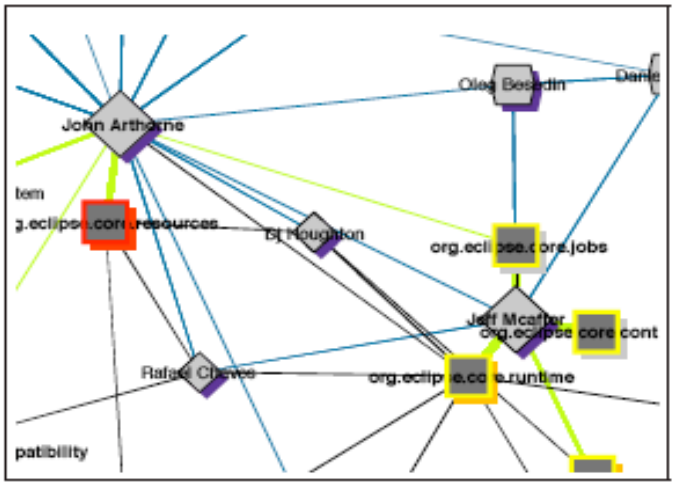
July 2005

(c)



July 2005

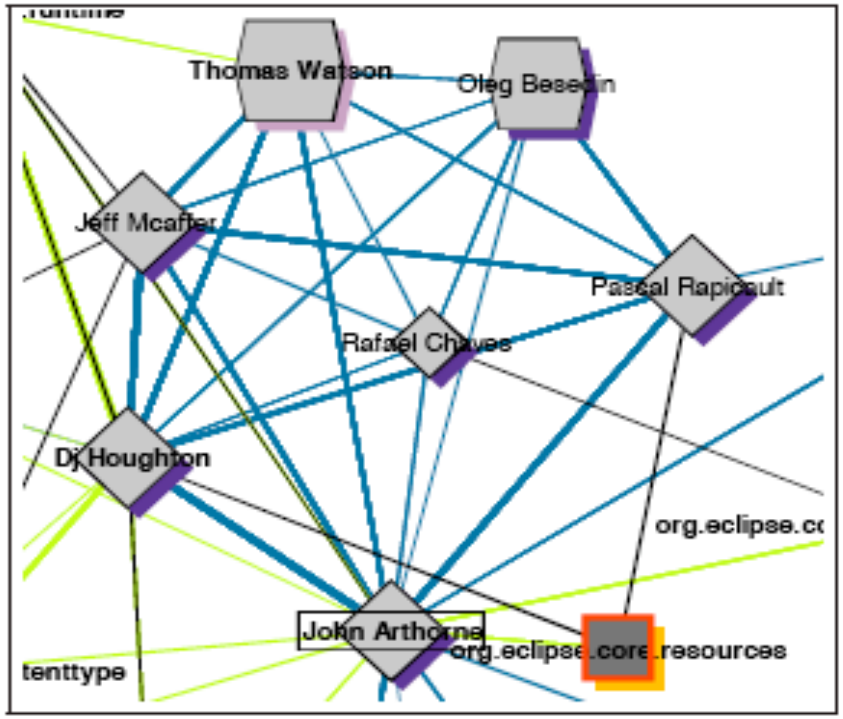
(d)



November 2005

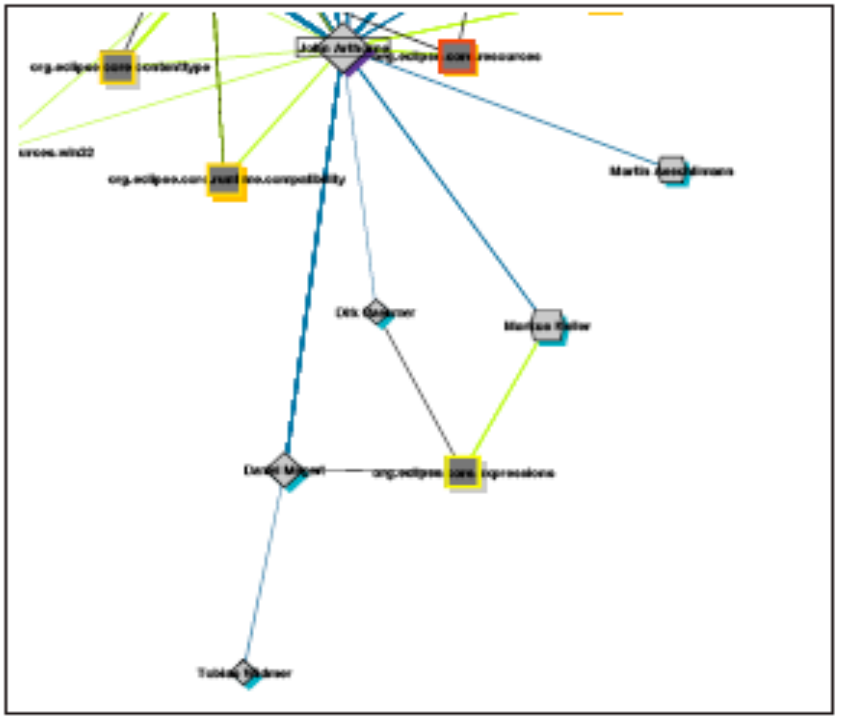
Scenario: distributed teams

(a)



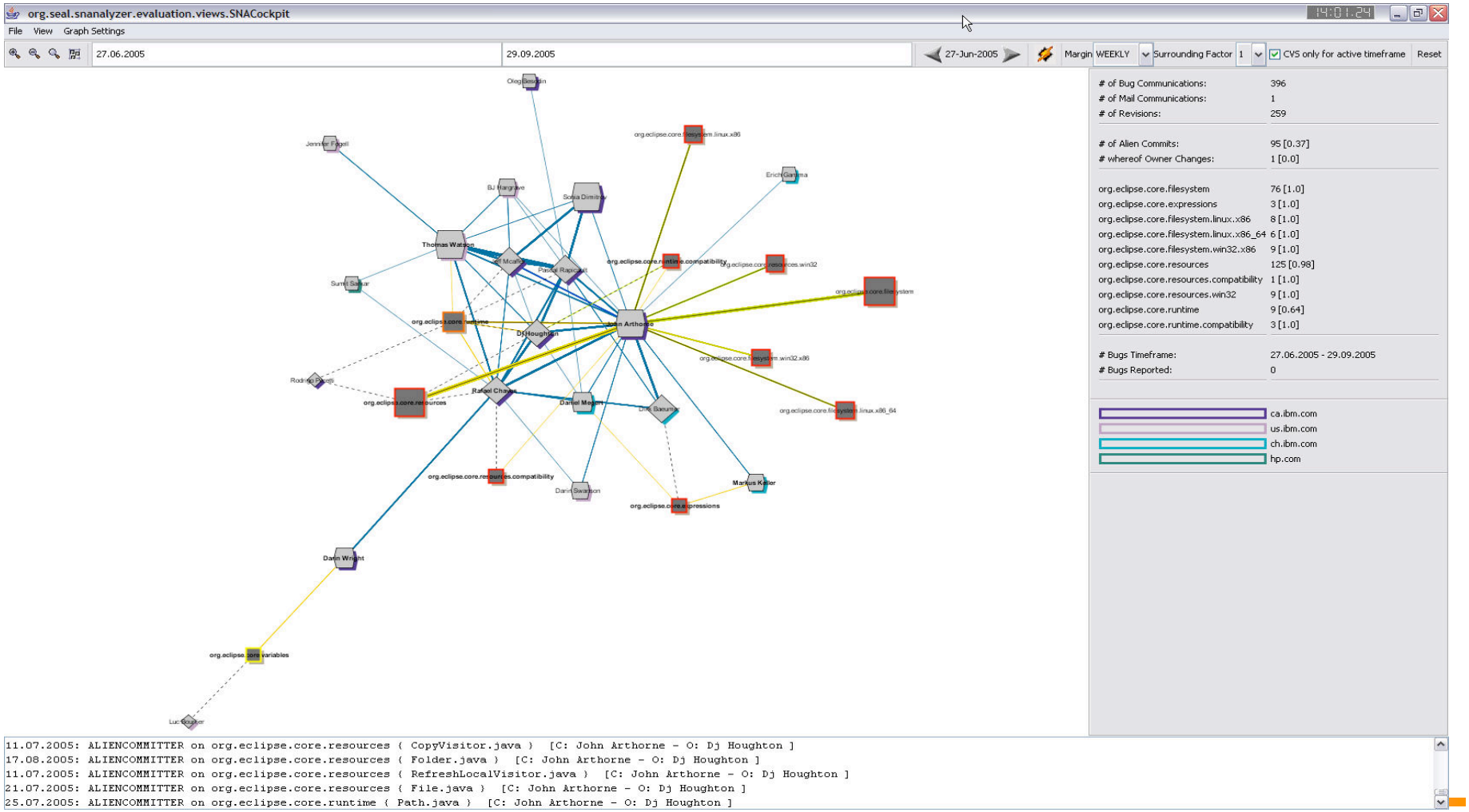
Sub-Group @us.ibm.com

(b)



Sub-Group @ch.imb.com

SNA Cockpit


org.seal.snanalyzer.evaluation.views.SNACockpit
File View Graph Settings
27.06.2005
29.09.2005
27-Jun-2005
Margin WEEKLY
Surrounding Factor 1
CVS only for active timeframe
Reset
# of Bug Communications: 396
# of Mail Communications: 1
# of Revisions: 259
# of Alien Commits: 95 [0.37]
whereof Owner Changes: 1 [0.0]
org.eclipse.core.filesystem 76 [1.0]
org.eclipse.core.expressions 3 [1.0]
org.eclipse.core.filesystem.linux.x86 8 [1.0]
org.eclipse.core.filesystem.linux.x86_64 6 [1.0]
org.eclipse.core.filesystem.win32.x86 9 [1.0]
org.eclipse.core.resources 125 [0.98]
org.eclipse.core.resources.compatibility 1 [1.0]
org.eclipse.core.resources.win32 9 [1.0]
org.eclipse.core.runtime 9 [0.64]
org.eclipse.core.runtime.compatibility 3 [1.0]
Bugs Timeframe: 27.06.2005 - 29.09.2005
Bugs Reported: 0
ca.ibm.com
us.ibm.com
ch.ibm.com
hp.com
11.07.2005: ALIENCOMMITTER on org.eclipse.core.resources (CopyVisitor.java) [C: John Arthorne - O: Dj Houghton]
17.08.2005: ALIENCOMMITTER on org.eclipse.core.resources (Folder.java) [C: John Arthorne - O: Dj Houghton]
11.07.2005: ALIENCOMMITTER on org.eclipse.core.resources (RefreshLocalVisitor.java) [C: John Arthorne - O: Dj Houghton]
21.07.2005: ALIENCOMMITTER on org.eclipse.core.resources (File.java) [C: John Arthorne - O: Dj Houghton]
25.07.2005: ALIENCOMMITTER on org.eclipse.core.runtime (Path.java) [C: John Arthorne - O: Dj Houghton]

Evolizer

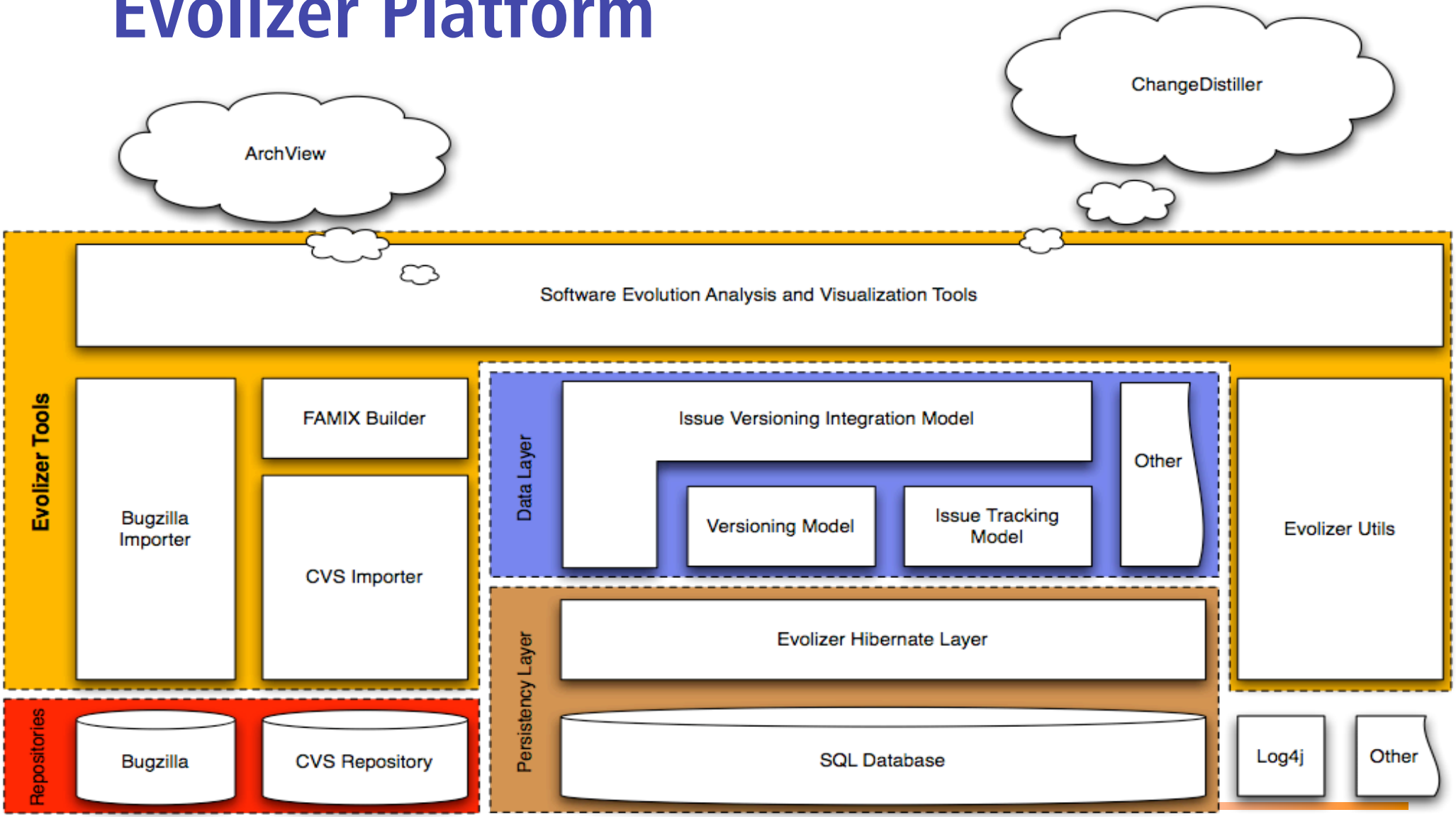
A platform for harvesting and provisioning of software evolution data

s.e.a.l.
software evolution & architecture lab

The Architecture of Evolizer

- Plug-in architecture
- Layers
  - Repositories
  - Data importers
  - Data integrators
  - Data providers
  - Data consumers

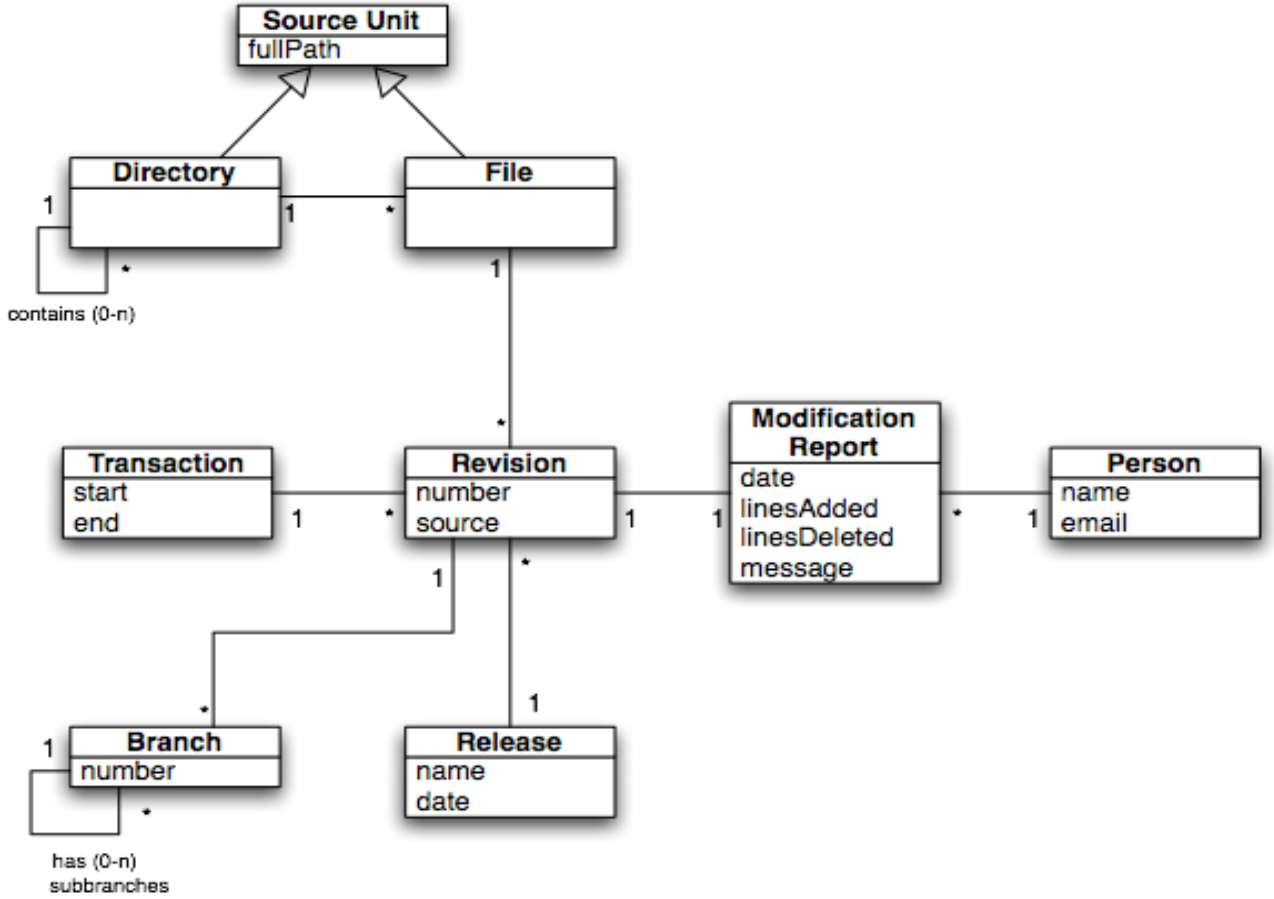
Evolizer Platform



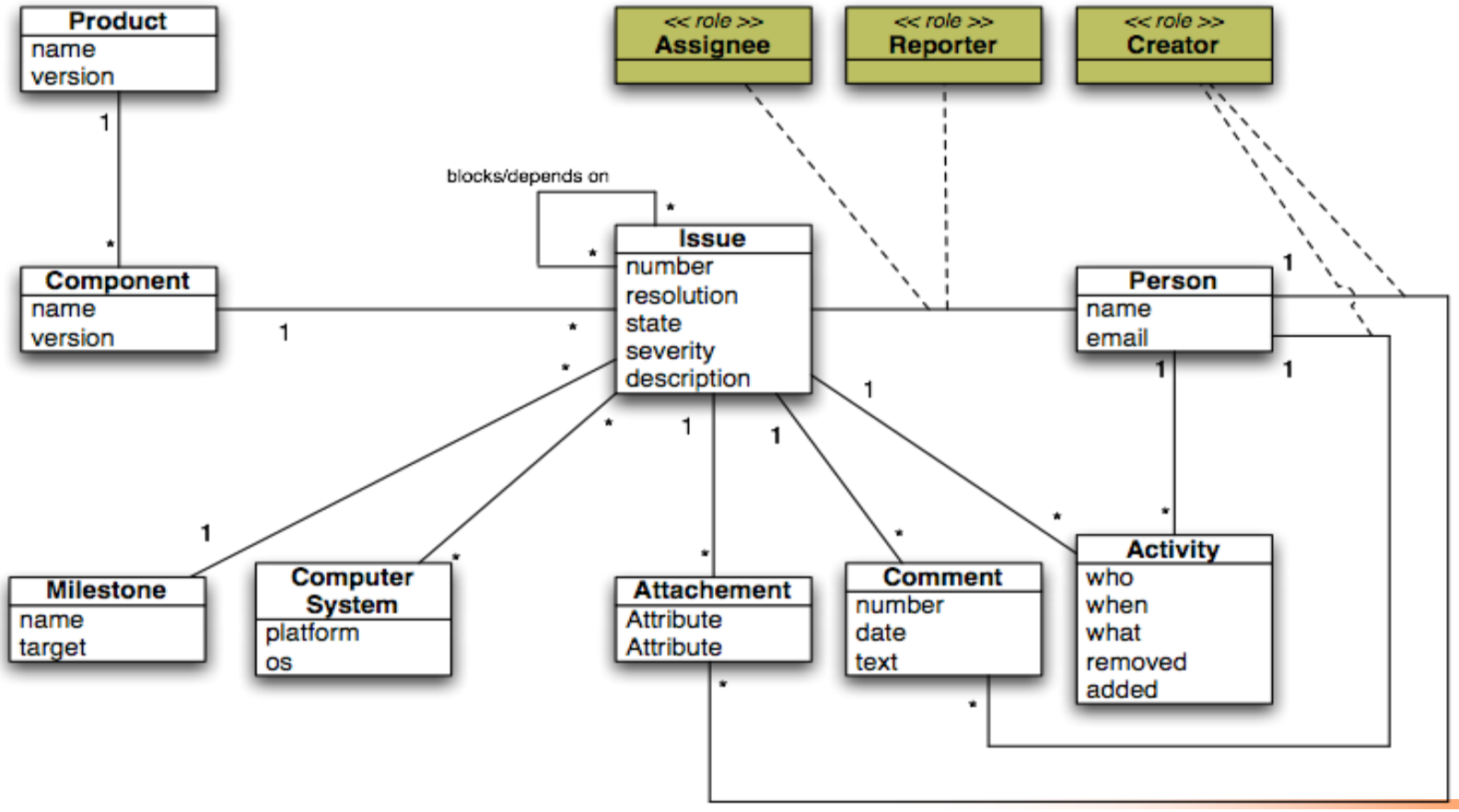
Data Models in Evolizer

- Data Models provide an interface to the information harvested from a software repository
  - One model per repository
  - Models can integrate other models

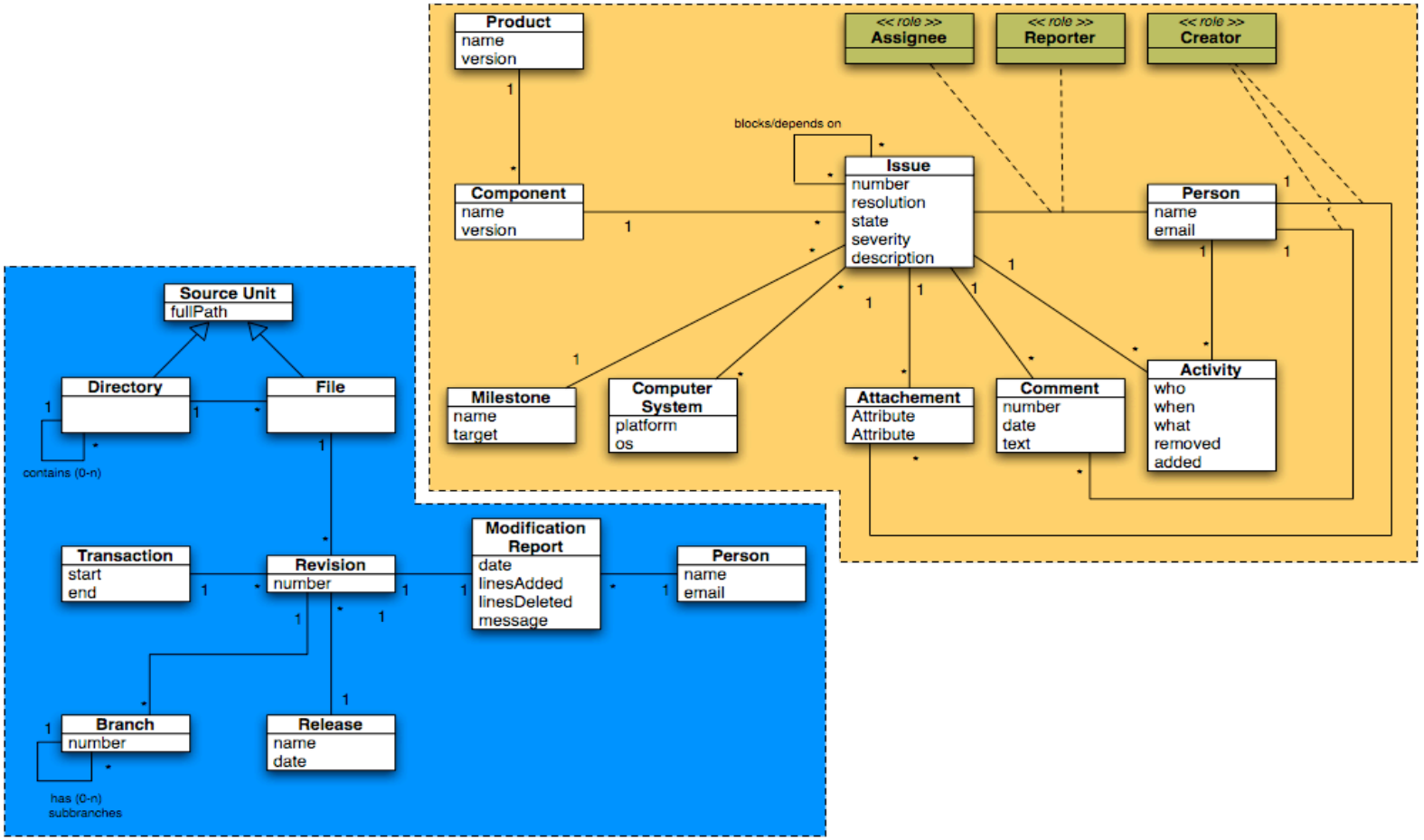
Version Control Model



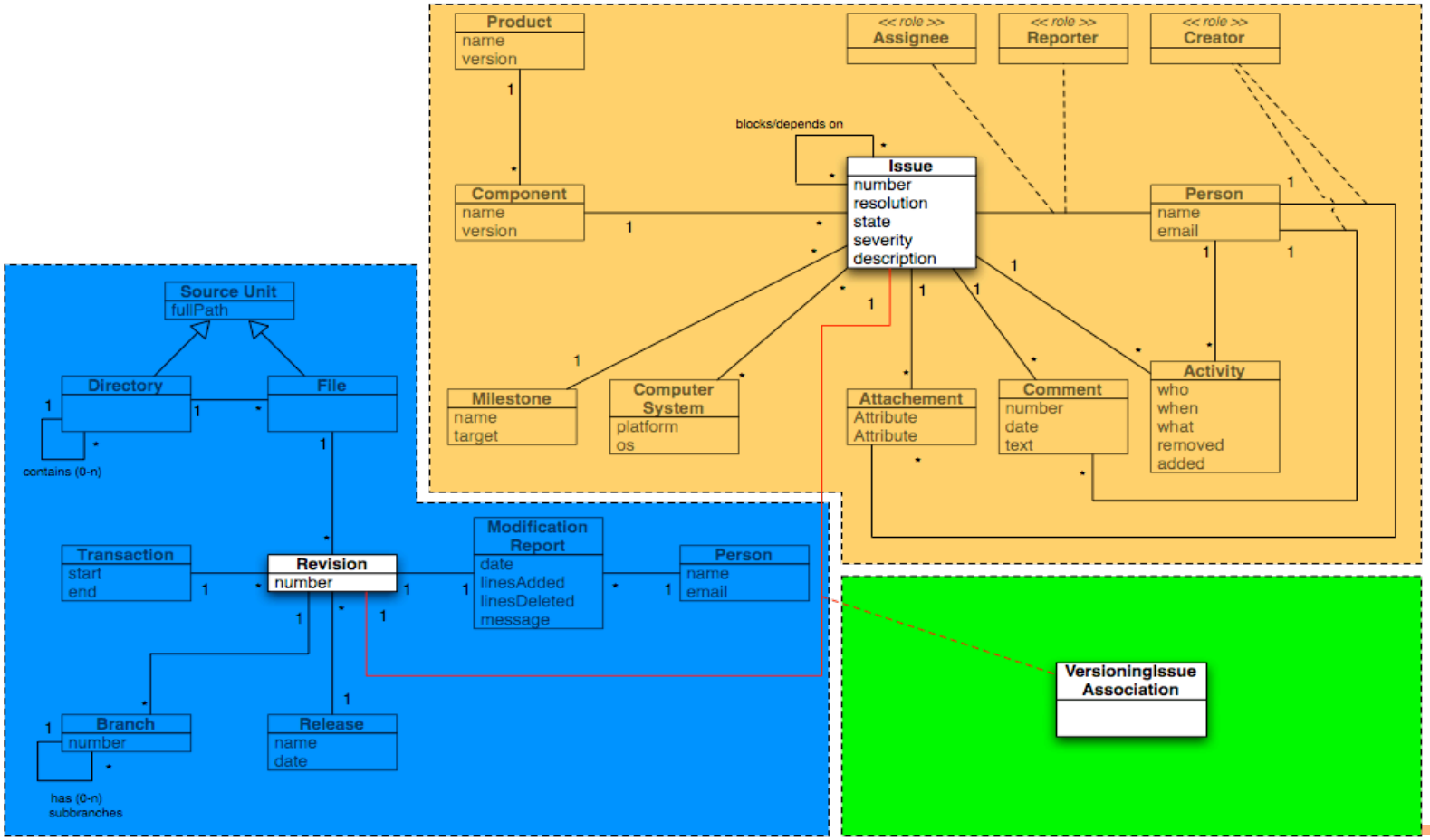
Bug Tracking Model



Bridging the Gap



Bridging the Gap



Evolizer Tools

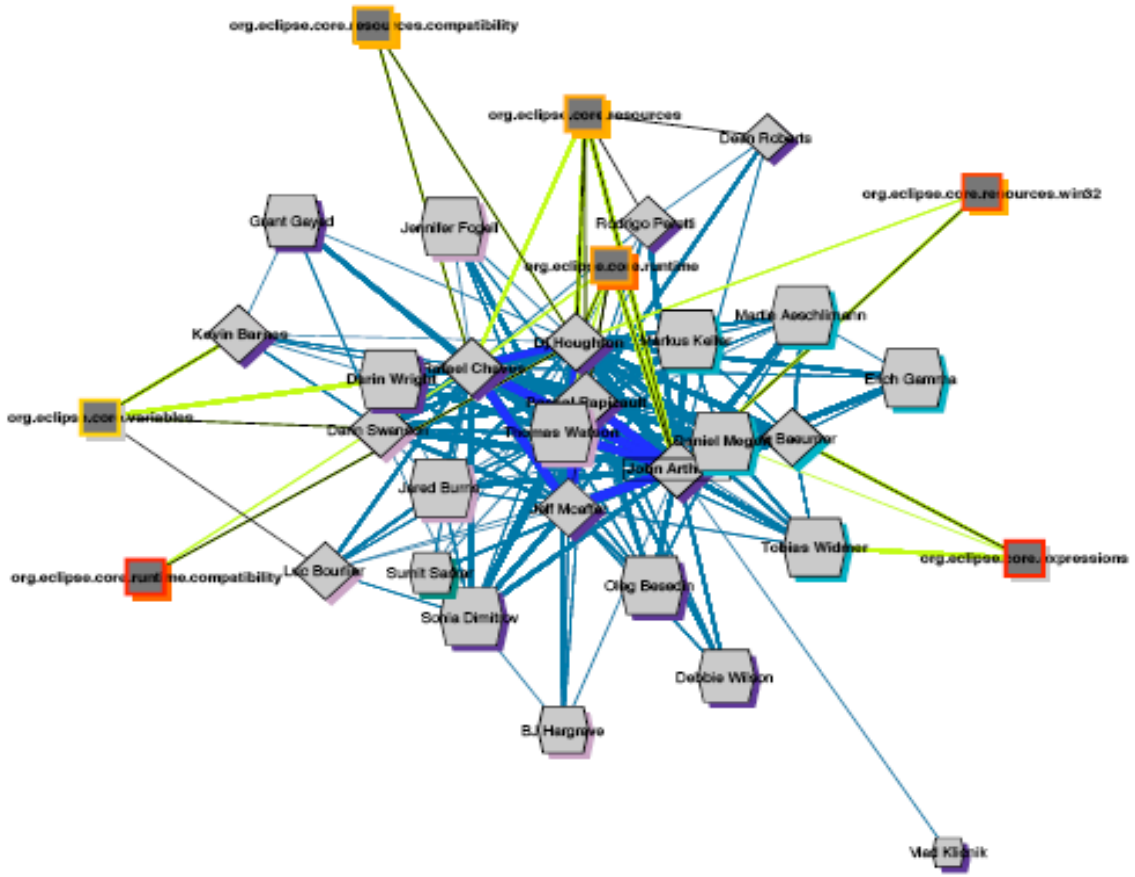
- *ChangeDistiller*: change types and significance
- *ArchView*: evolution metrics
- *SNA Cockpit*: developer networks
- *Evolution Browser*
- *Comment Analyzer*: code and comments
- *Clone Evolution*

Conclusions

Résumé

- Analyzing *software evolution* is a multi-source/-view/-dimension/-stakeholder challenge
  - Technical: resides in modeling and handling various kinds of information
  - Conceptual: answering interesting questions and presenting the results (visually)
- Mining software repositories has been embraced by both the software evolution and the empirical software engineering community
- Social networks are a key

Developers, developers, developers



References /1

- Anvik, J., Hiew, L., and Murphy, G. C., Who should fix this bug?. In Proceeding of the 28th international Conference on Software Engineering, Shanghai, China, May 2006
- John Anvik , Gail C. Murphy, Determining Implementation Expertise from Bug Reports, Proceedings of the Fourth International Workshop on Mining Software Repositories, May 2007
- Ivan T. Bowman, Richard C. Holt, Reconstructing Ownership Architectures To Help Understand Software Systems, Proceedings of the 7th International Workshop on Program Comprehension, May 1999
- G. Canfora and L. Cerulo. How software repositories can help in resolving a new change request. In Workshop on Empirical Studies in Reverse Engineering, 2005
- D. Čubranić and G. C. Murphy. Automatic bug triage using text classification. In Proceedings of Software Engineering and Knowledge Engineering, pages 92--97, 2004
- Davor Cubranic, Gail C. Murphy, Janice Singer, Kellogg S. Booth, Hipikat: A Project Memory for Software Development, IEEE Transactions on Software Engineering, v.31 n.6, p.446-465, June 2005
- M. Fischer, M. Pinzger, and H. Gall, Populating a Release History Database from Version Control and Bug Tracking Systems, Proc. of 19th International Conference on Software Maintenance (ICSM 2003), Amsterdam, Netherlands, IEEE CS, 2003
- Fluri, B., Wuersch, M., Gall, H.C.: Do code and comments co-evolve? on the relation between source code and comment changes. In: Proc. Working Conf. Reverse Eng., IEEE CS, 2007
- Beat Fluri, Michael Würsch, Martin Pinzger, Harald C. Gall, Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction, IEEE Transactions in Software Engineering (TSE), vol. 33, no. 11, November 2007

References /2

- Kim, S. and Ernst, M. D. 2007. Which warnings should I fix first?. In Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC-FSE '07), Dubrovnik, Croatia, September 2007
- Audris Mockus , James D. Herbsleb, Expertise browser: a quantitative approach to identifying expertise, Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), Orlando, Florida, May 2002
- Nachiappan Nagappan, Brendan Murphy, Victor Basili, The Influence of Organizational Structure on Software Quality: An Empirical Case Study, ICSE 2008 (to appear)
- S. Kim, T. Zimmermann, E. J. Whitehead, Jr., and A. Zeller, Predicting Bugs from Cached History, Proc. of the 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, USA, May 2007
- Gursimran Singh Walia, Jeffrey C Carver, Nachiappan Nagappan, The Effect of the Number of Inspectors on the Defect Estimates Produced by Capture-Recapture Models, ICSE 2008 (to appear)
- Cathrin Weiss, Rahul Premraj, Thomas Zimmermann, Andreas Zeller, How Long Will It Take to Fix This Bug?, Proceedings of the Fourth International Workshop on Mining Software Repositories, May 2007
- J. Zliwerski, T. Zimmermann, and A. Zeller, When Do Changes Induce Fixes?, Proc. of Int'l Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, May 2005
- Thomas Zimmermann, Nachiappan Nagappan, Predicting Defects using Social Network Analysis on Dependency Graphs, ICSE 2008 (to appear)