Dependencies and Reflections in Collaborative Software Engineering

David Redmiles

with Cleidson de Souza, Stephen Quirk, Erik Trainer

redmiles@ics.uci.edu

Background

- A part of a multi-investigator National Science Foundation project managed through ISR.
- With cooperation from NASA, Boeing, and IBM.
- Leveraging funding from IBM through the Eclipse Innovation Program, by the Brazilian Government through a CAPES student grant, and a UCI-ICS Collaborative Research Initiation Award.
- And continuing with support from the National Science Foundation, UCI-ICS, and Hitachi.

Dependencies in Collaboration

- Communication and coordination breakdowns have negative effects on collaborative software development projects especially when team members are not collocated.
  - E.g., globally distributed software development
- Many breakdowns arise over dependencies.
  - I.e.,dependencies among people and between people and artifacts.
- Traditionally, software engineering techniques support managing these dependencies through formal techniques.
- In reality, participants must additionally employ informal techniques to manage dependencies (and navigate the formal techniques).

Awareness and Reflections of Dependencies

- Many informal activities pertain to maintaining awareness.
- Therefore we seek to support coordination by reflecting social and technical dependencies helping software developers to maintain awareness of and easily discover these dependencies.
- Our implementation relies primarily on visual interfaces.

Related Work

- Some are using the term socio-technical congruence (e.g. ICSE 2008 workshop by Herbsleb et al.)
  - Dependencies in the source-code (technical) create dependencies between people writing that source-code (social)
- Explicit relationship between dependencies and coordination
  - By minimizing dependencies, reduce required communication/coordination (Conway 1968, Parnas 1972, Sosa 2002, Grinter 2003, de Souza, Redmiles 2004, Curtis 1988, Herbsleb and Grinter 1999)

Approach

- Field study (ethnography) of software organizations
- Grounded theory analysis
- Development of stereotypical scenarios
- Development of software visualizations (2 versions *and counting)*

Site 1 - MVP

- 34 software engineers in 2 sub-teams
  - Developers
  - Quality assurance (V&V)
- Work together for about 9 years.
- Do not need to interact with external teams.
- A non-modular software
  - Changes in one part can affect almost any other part.

Site 2 - MCW

- 57 software developers in 5 sub-teams
  - Client, server, test, infrastructure, and leads
- Work together for about 9 months
- Do need to interact with several external teams;
  - Part of a large organization implementing a software reuse program
- A modular software based on S.E.’s best practices (APIs, layers, etc)

Data Collection

- Semi-Structured Interviews
  - MVP: 8 informants
  - MCW: 15 informants
  - The interview guide was reused
- Non-participant Observation (Shadowing)
  - MVP: 8 weeks
  - MCW: 11 weeks

Actual Work Practices

- Learning from Email Notifications; Personal Network; Reading Email Notifications; Impact Descriptions; Error-Checking; Back Merges; Partial Check-in's; Being aware by attending meetings, engaging in communication; Grouping requirements; Informal Code Reviews; Holding onto Check-in's; Notifications;
- API design review meetings; Sending Email Notifications; Pre-Testing Activities; Build Document; "Exporting" Developers; Problem Reports; Formal Code Reviews
- The reference architecture and APIs; Handling External Dependencies: APIs and Adaptors

Scenarios (1-2)

- Awareness of Evolving Dependencies
  - Managers' lack of awareness of developers' social dependencies
    - Gauging integration progress between team members
    - Assessing the likelihood of meeting deadlines
  - Developers' lack of awareness of evolving technical dependencies
    - Whether an API is "being exercised"
    - Planning for last minute changes, re-designs

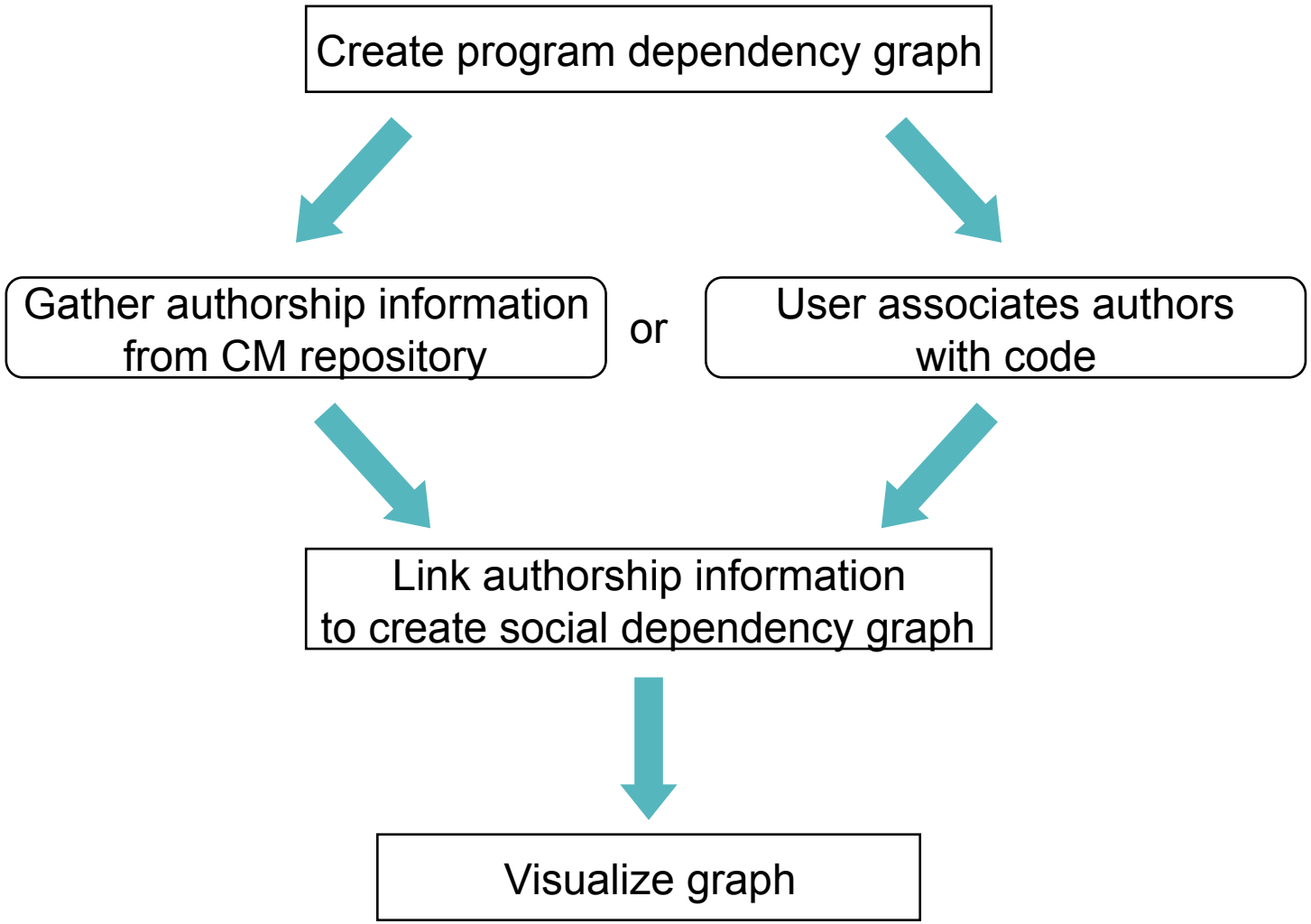
Scenarios (3-4)

- Finding the right people with whom to talk
  - Developer's finding the "right" developer
    - Programming against "dummy" implementations
    - Want to find who else is implementing the same code, not who designed and checked in the interface
  - Developers finding "similar" developers
    - How to use a particular component
    - Identify similar/overlapping work
    - Who will be affected by changes to a component
    - Leverage the needs of others to request changes to a heavily used component

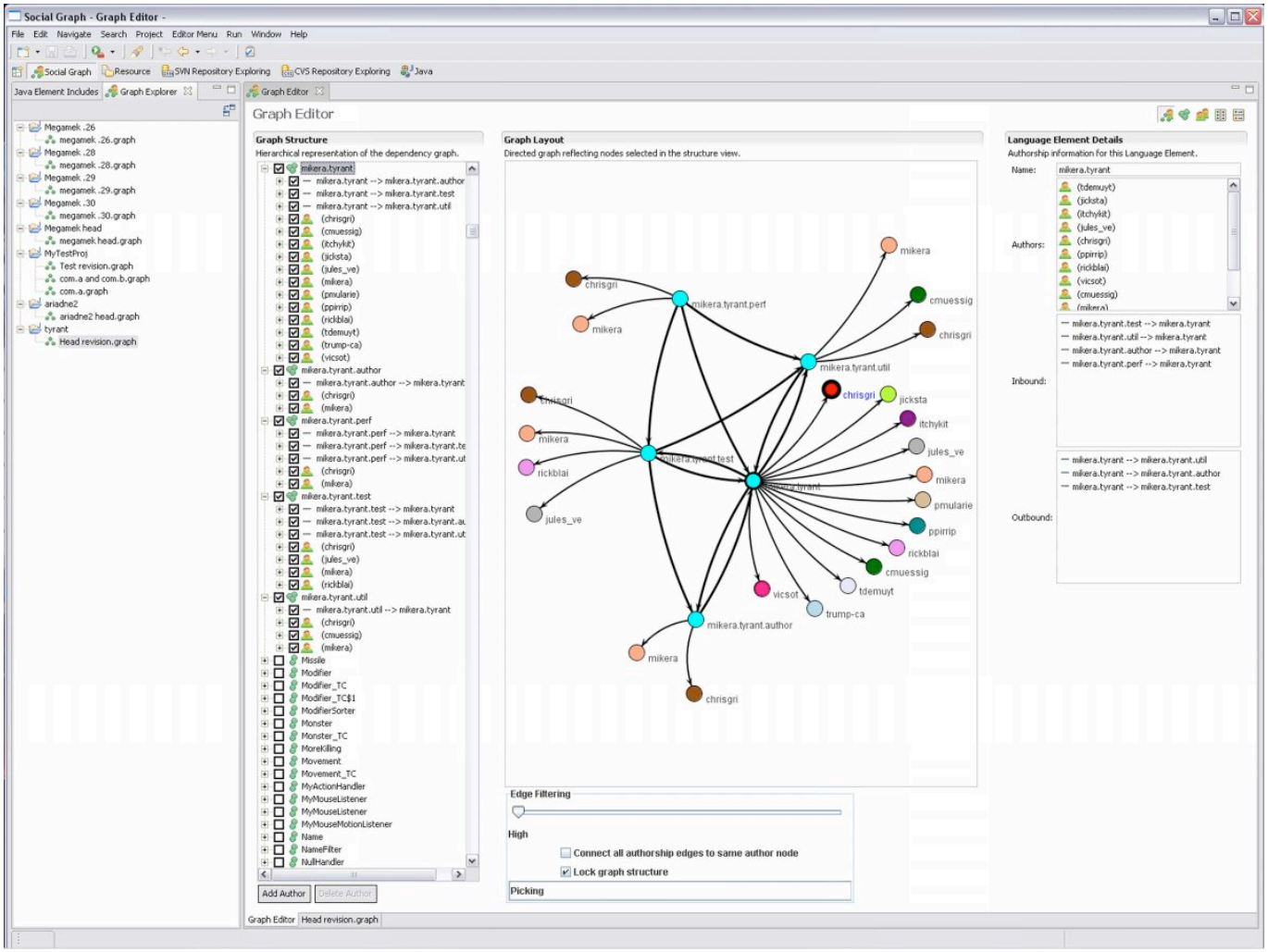
Responding to Scenarios

- Dependency/coordination relationship has not been fully explored
- But it should be!
  - Dependencies can be detected by automated tools
  - Dependencies, and thus coordination, can change
- We need tool support
  - The goal of Ariadne is to fill the acknowledged gap between dependencies and coordination
  - Ariadne automates dependency analysis and collection of authorship information, and generates social networks
  - Eclipse plug-in

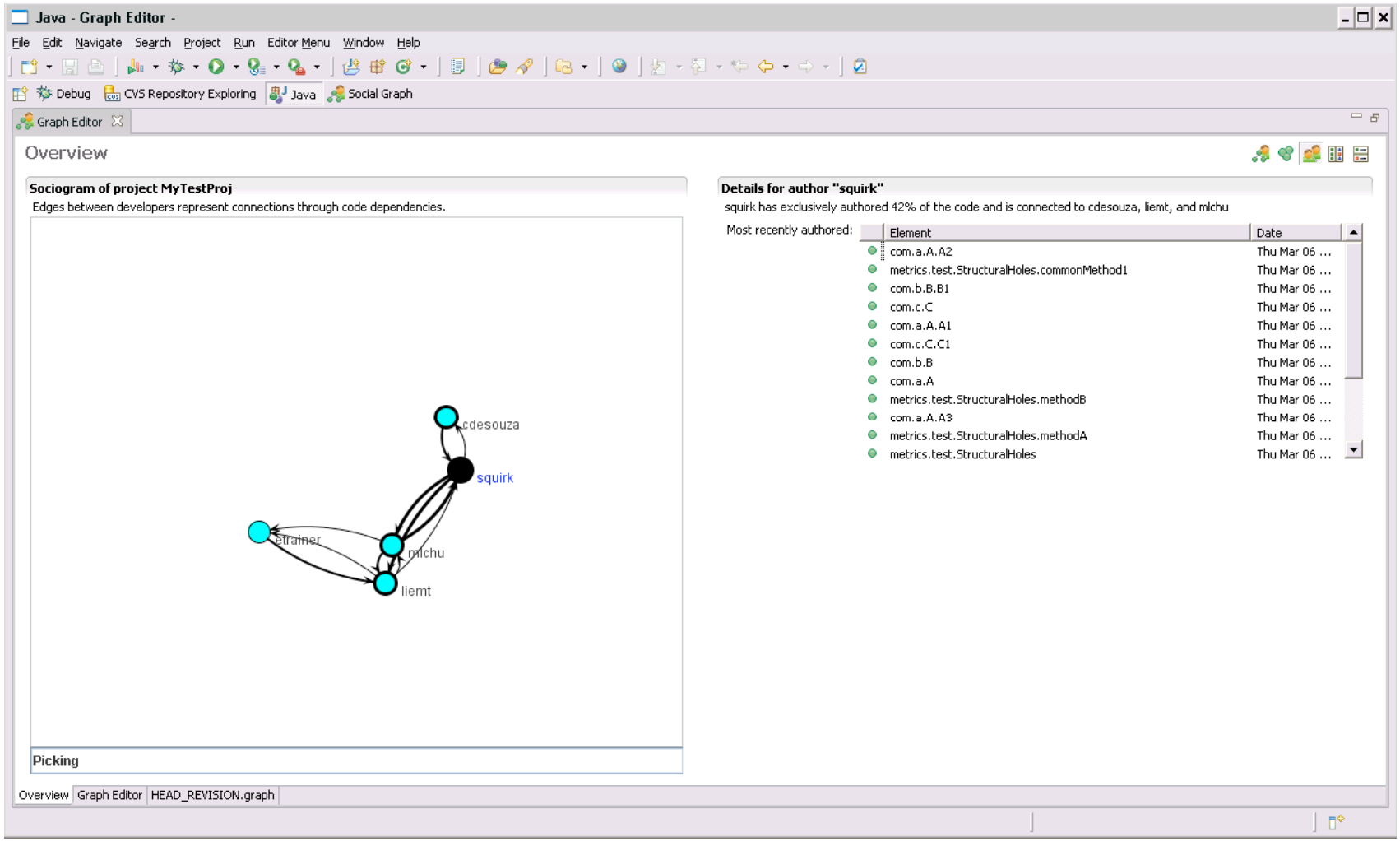
Automated Process



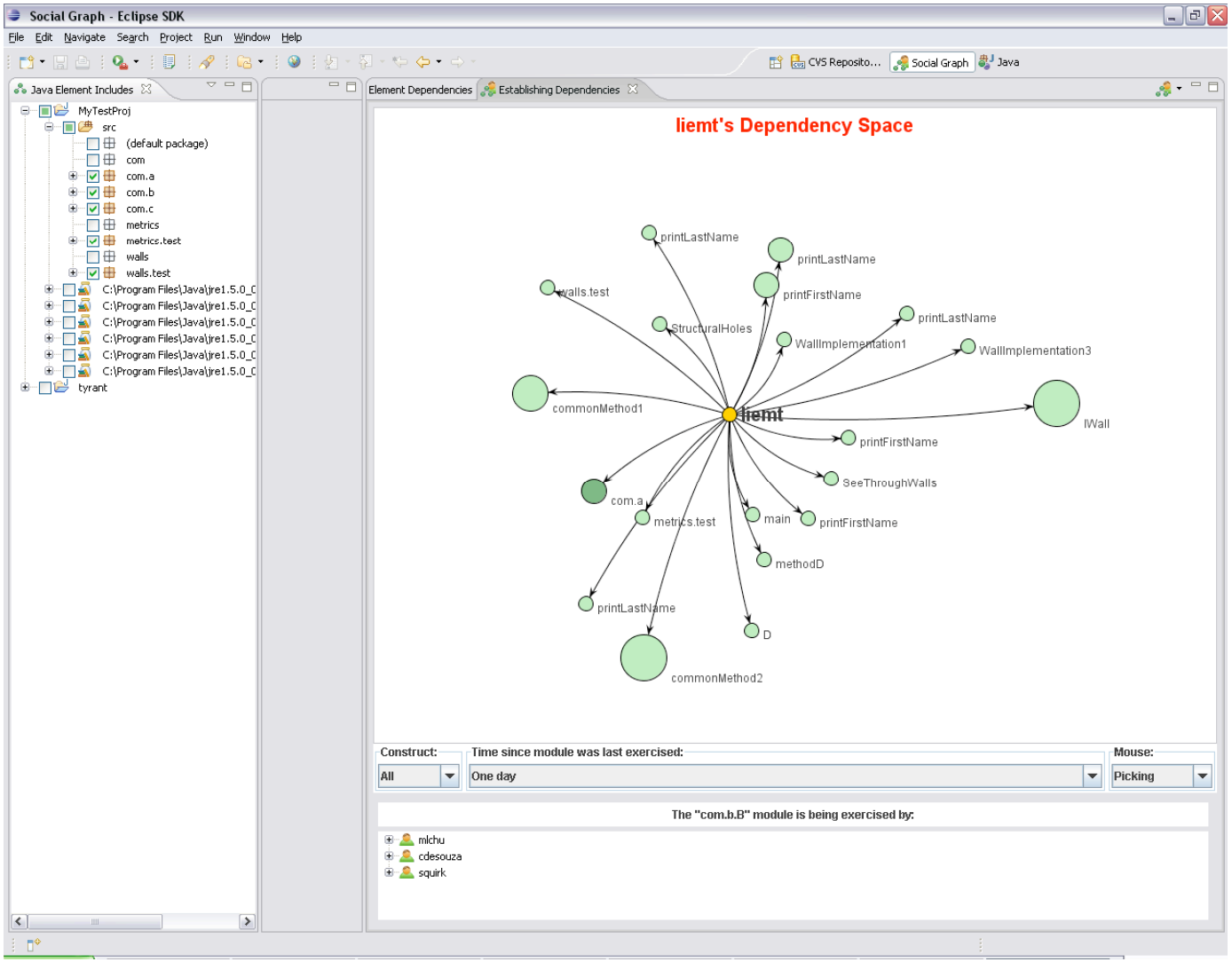
Ariadne - Social and Technical Dependencies among Developers and Components



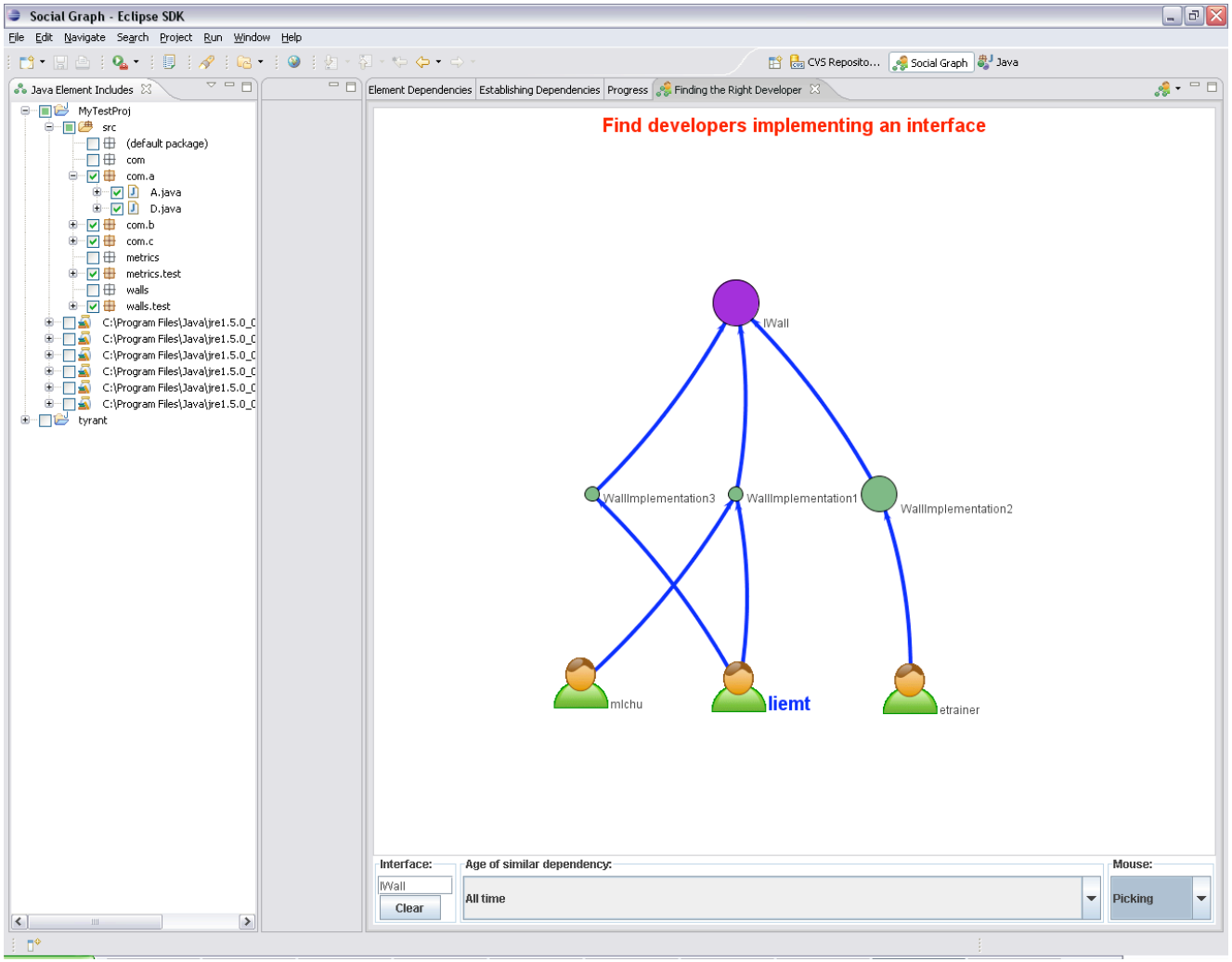
Managers' Lack of Awareness



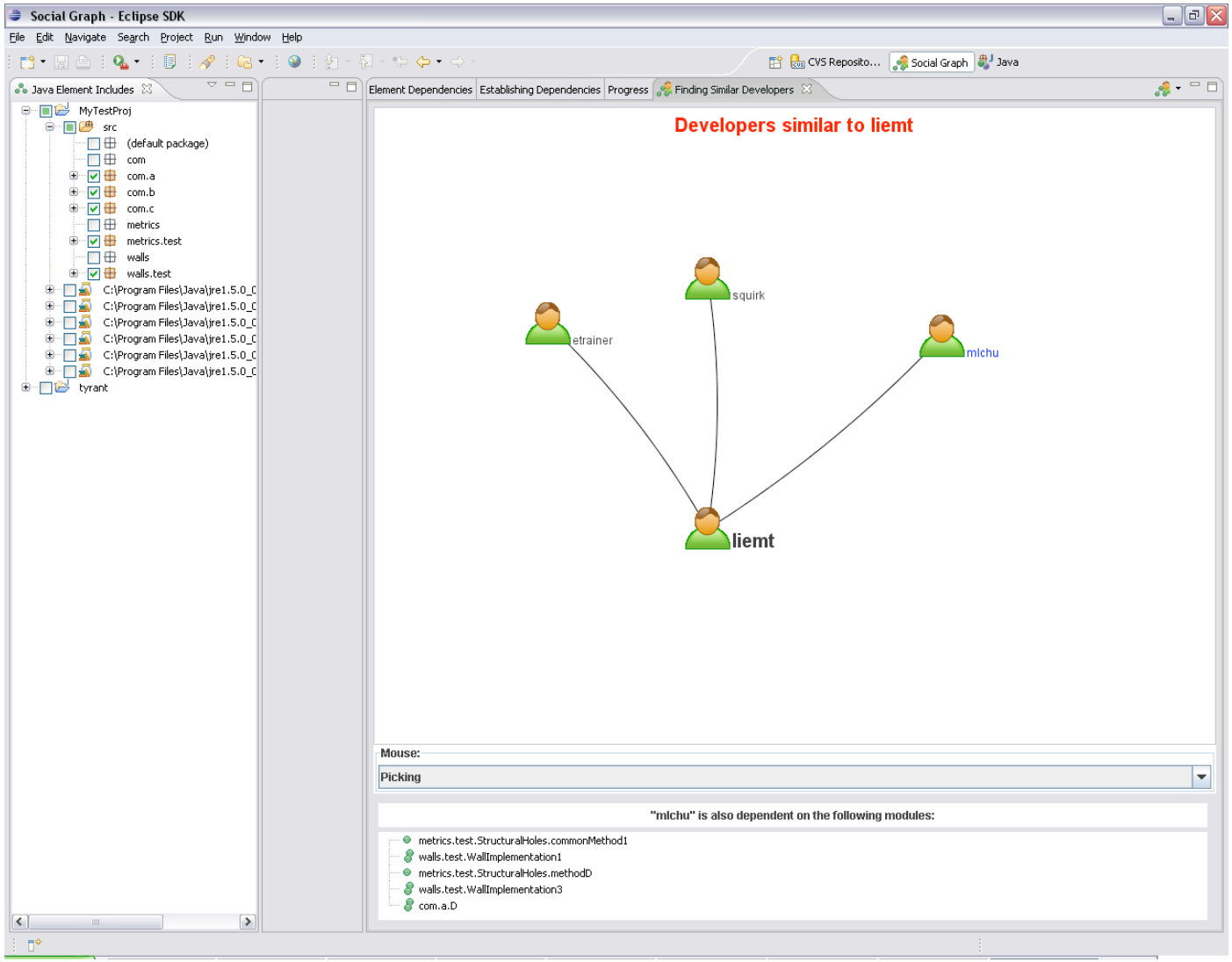
Developers' Lack of Awareness



Finding the “Right” Developer



Finding Similar Developers



Feedback

- Feedback from open-source developers, running the tool on whole software project rather than subsets
- Problems with social network graphs
  - Layouts are not “geographically” consistent from analysis to analysis
  - Graphs do not scale well without smart filtering/zooming
  - Difficult to show social and technical dependencies together
- Explore alternate visualizations and evaluations

Goals for Alternate Visualization

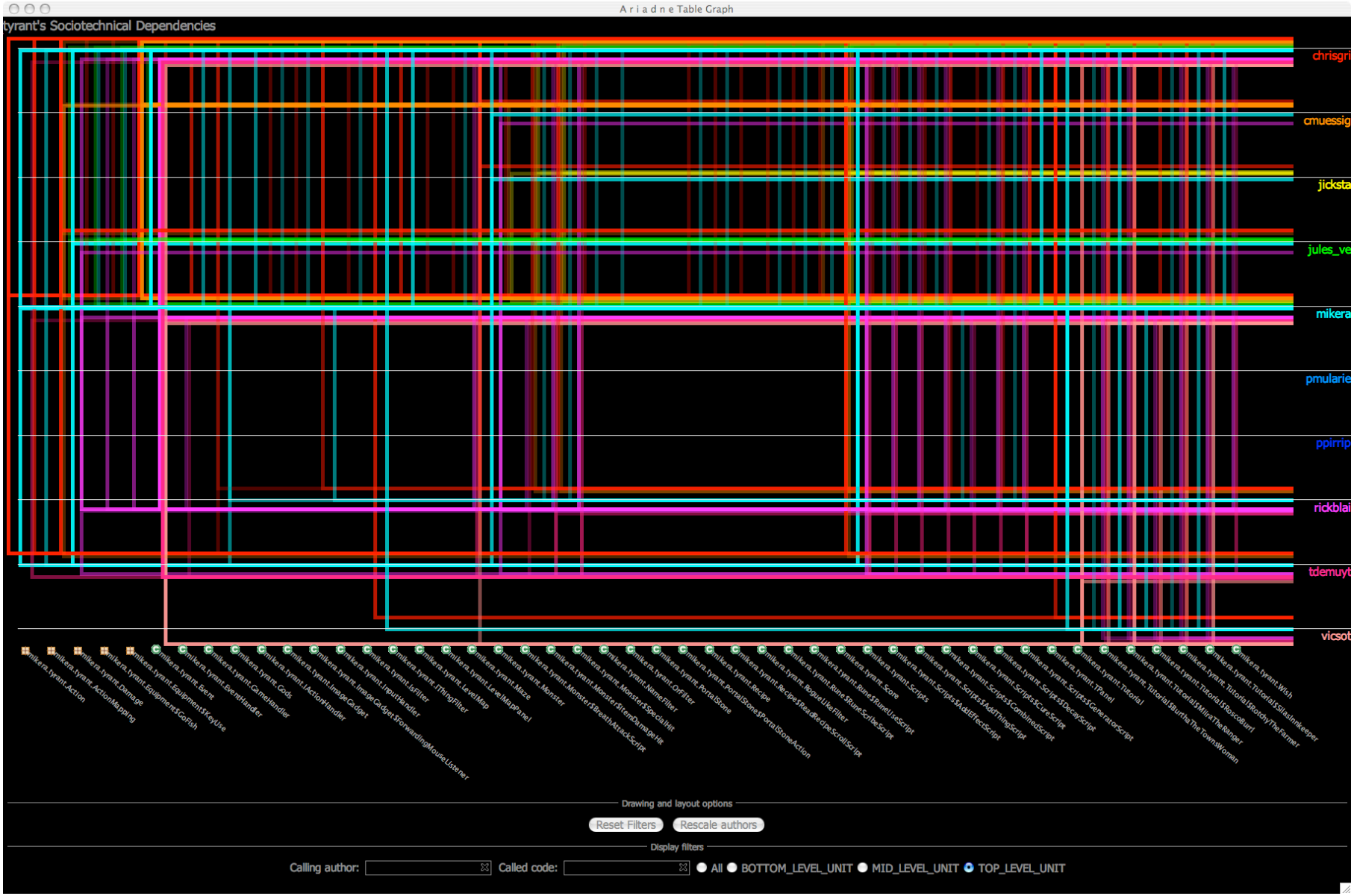
- Instead of four different visualizations, one
- Preserve ease of identify connections in social network graphs
- Consistent layouts
- Showing many data at once, more scalable
- Evaluation using methods appropriate for visual interfaces
  - Lewis, Polson et al.'s Cognitive Walkthrough, Tufte's Information Visualization Principles, Nielsen's Heuristic Evaluation
- Evaluation with real data sets
- Evaluation with end users

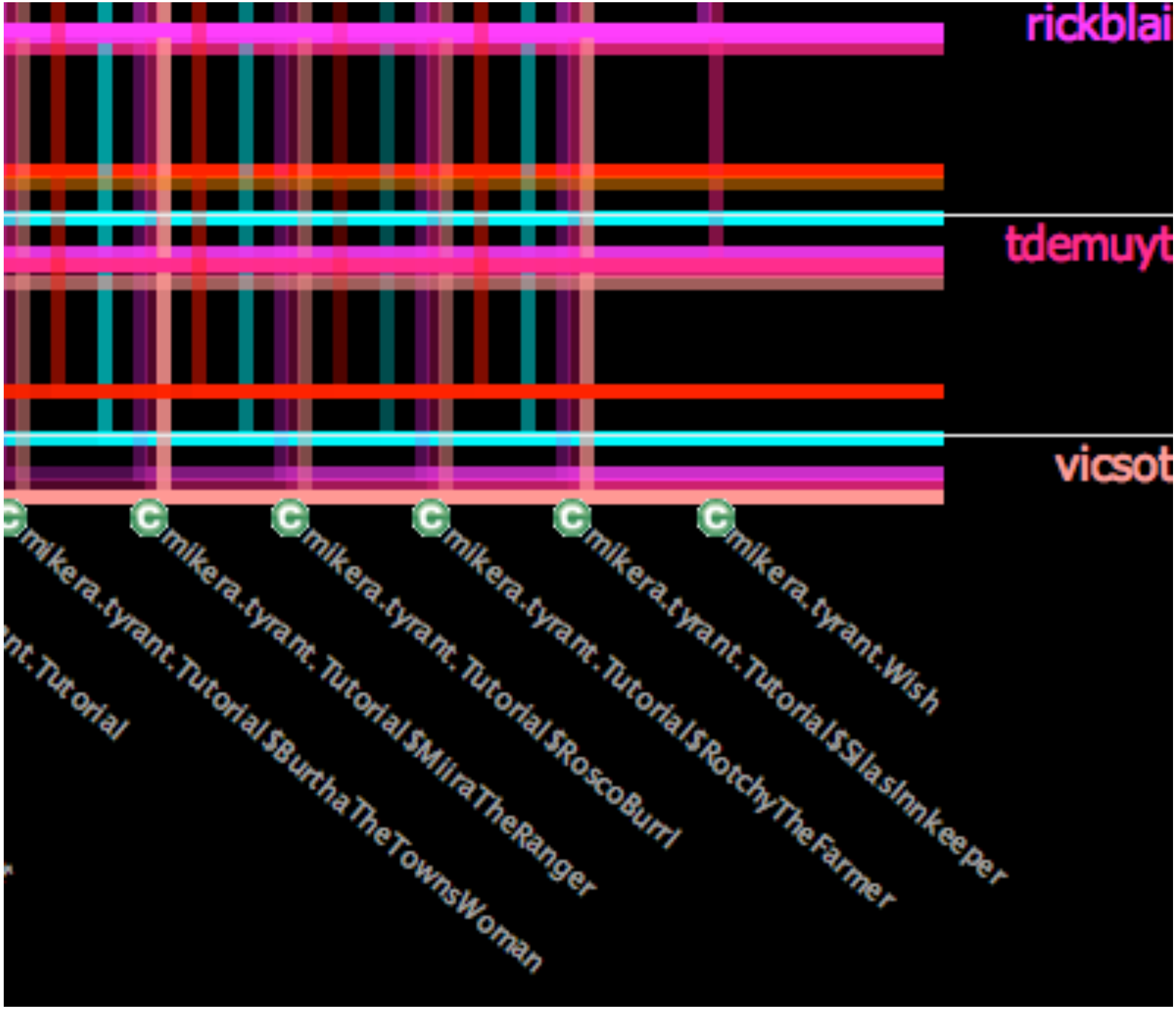
Content – Problem Context

Photo googled at http://jurmo.us/2007/03/04/work-20-the-empty-cubicle/
Original source from http://www.ebertfest.com/seven/playtime.htm

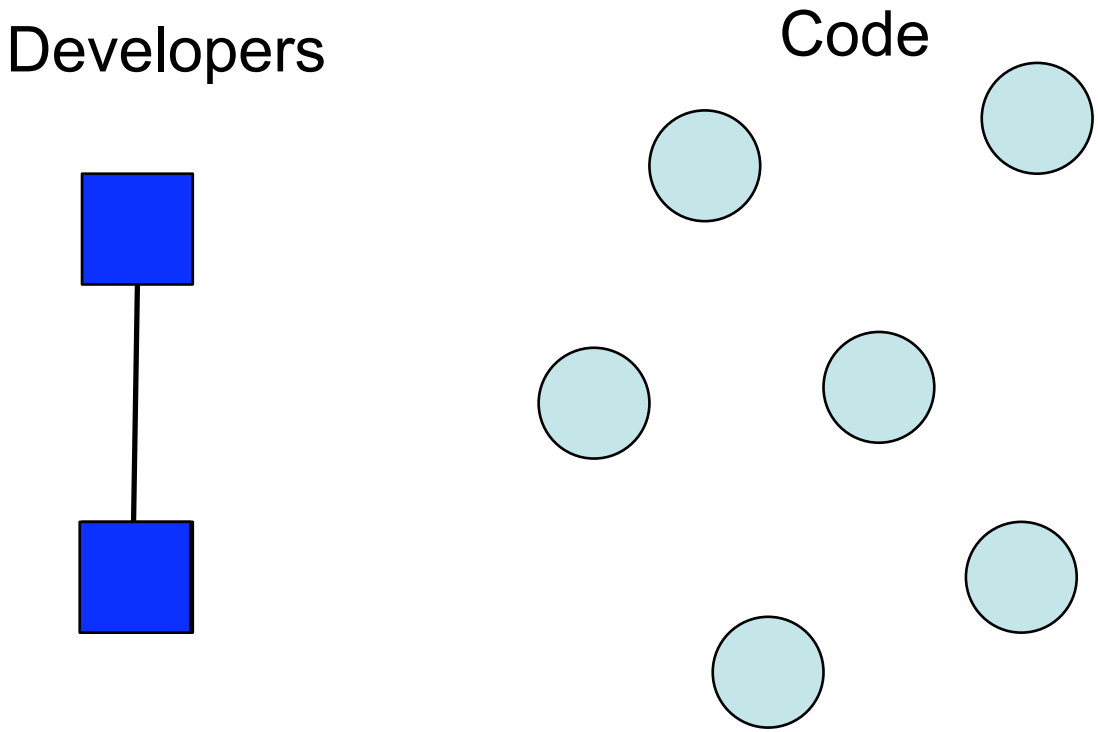
Implementing Ariadne 2.0

Multivariate Analysis

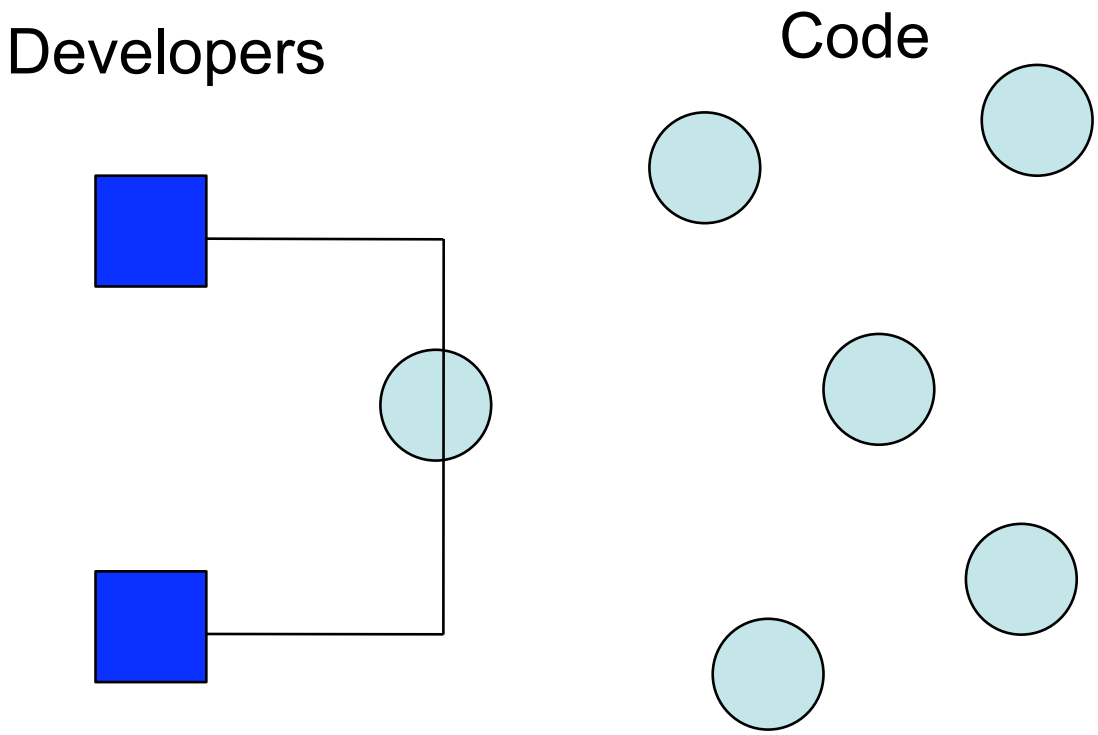



rickblai
tdemuyt
vicsot
mikera.tyrant.Wish
mikera.tyrant.Tutorial$SilasInnkeeper
mikera.tyrant.Tutorial$RotchyTheFarmer
mikera.tyrant.Tutorial$RoscoBurri
mikera.tyrant.Tutorial$MiiraTheRanger
mikera.tyrant.Tutorial$BurthaTheTownsWoman
nt.Tutorial

Progression of Graphs to Brackets (1)

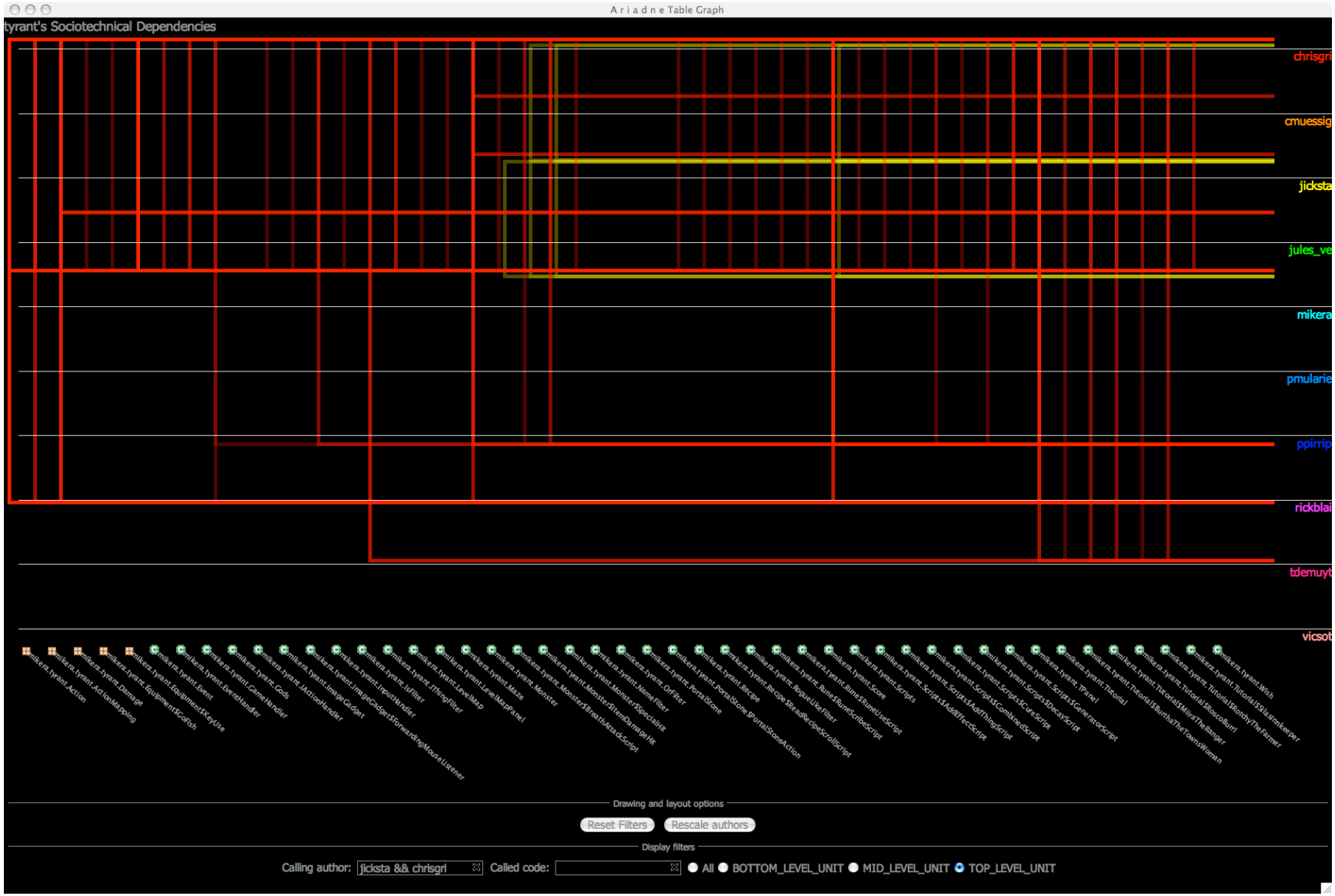


Progression of Graphs to Brackets (2)

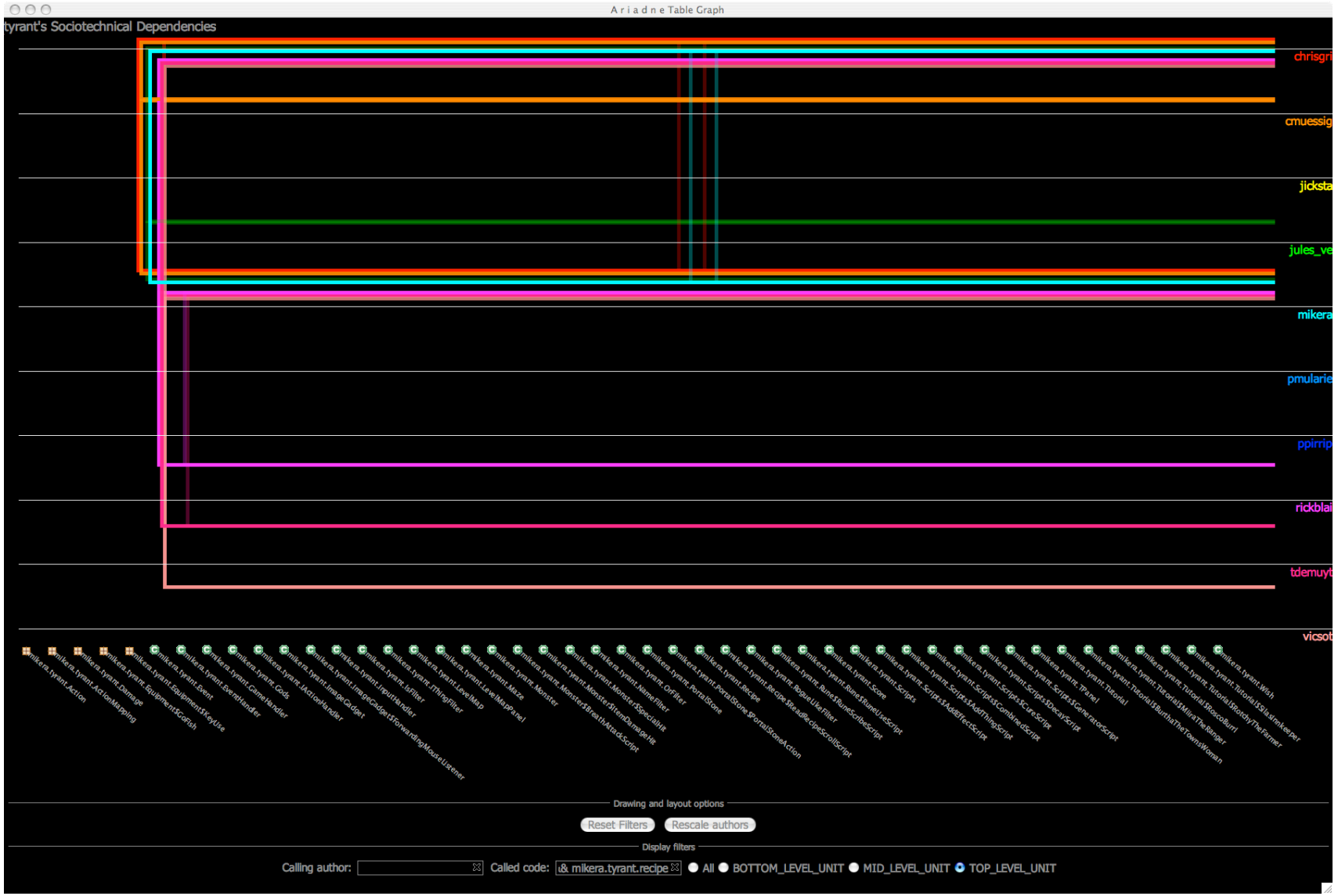


Stretching the string allows us to see through which artifact the dependency is established.

Comparisons / Filter by Author



Comparisons / Filter by Artifact



Results

- Social and technical information is needed and can be combined in visual tools
  - Grounded on ethnographic field studies and user interface evaluation
- Tools can affect self-coordination and performance in aware-critical tasks
  - Individual differences can be overcome

Potential Users

- Project managers or researchers
  - Identify key roles played by developers
    - How do roles change over time
  - Determine coordination needs of team members
- Software developers
  - Identify the “right developer,” or owner
  - Find out which developers have started to integrate their code

Further Details of On-going Work

- Poster: Continuous Coordination within the Context of Cooperative and Human Aspects of Software Engineering
  - Students: Erik Trainer, UC Irvine/ISR, Roger Ripley, UC Irvine/ISR
  - Project Scientist: Ban Al-Ani, UC Irvine, Post-Doc: Anita Sarma, CMU (formerly UC Irvine/ISR)
  - Advisors: André van der Hoek, UC Irvine/ISR, David F. Redmiles, UC Irvine/ISR

Further Reading

- http://awareness.ics.uci.edu:8080/ContinuousCoordination/
- Cleidson R. B. de Souza, Redmiles, D.F. *An empirical study of software developers' management of dependencies and changes,* Proceedings of the 30th International Conference on Software Engineering (ICSE08 - Leipzig, Germany) May 2008, pp. 241-250.
- Cleidson R. B. de Souza, Quirk, S., Trainer, E., Redmiles, D.F. *Supporting collaborative software development through the visualization of socio-technical dependencies,* Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work (GROUP07 - Sanibel Island, FL) November 2007, pp. 147-156.
- Redmiles, D., van der Hoek, A., Al-Ani, B., Hildenbrand, T., Quirk, S., Sarma, A., Silveira Silva Filho, R., de Souza, C., Trainer, E. Continuous Coordination: A New Paradigm to Support Globally Distributed Software Development Projects, WIRTSCHAFTSINFORMATIK, Special Issue on the Industrialization of Software Development, V. 49, 2007, pp. S28-S38.