Lecture 12

Mining Software Repositories, Part 2
Hipikat, Bugcache, Mining Social Network

Announcement

- Project Midpoint Review is coming up in two weeks.
  - You must have preliminary results. (That means you probably need to have a working prototype.)
  - This will count toward your final grade.
- Tool evaluation is due in two weeks.

Today's Agenda

- Quiz
- Presentation: Amal Banerjee
- Hipikat
    - Focusing on its evaluation
- FixCache
- Social Network Mining

Quiz on eRose

- 7-10 minutes
- It will be graded 0-3 point scale.

What kinds of information is available in open source software repositories?

Information in Software Repositories

- Version Control Systems
  - CVS, Clearcase, Subversion, etc
  - Code, file, version number, delta, author, time stamp, change log (commit msg), branch, etc
- Problem Report Databases
  - Bugzilla, GNATS, JIRA, etc.
  - Id, reporter, creation data, phase, component, OS, version, priority, severity, bug assignee, bug description, when fixed, etc.

Information in Software Repositories

- Regression Test
  - Time stamp, # success, # failure
- Build log
- Mailing list
- Newsgroup
- Code inspection or design meeting note, etc.

What’s NOT in software repositories?

What's NOT in software repositories?

- Refactoring information
- Semantics of software changes
- Organizational structure
- Design decisions
- Code navigation history
- Workspace setting
- Editing history/ Transformation history, etc.

What Can We Do with Software Repository Data?

- Identify related changes [Zimmermann et al. 04] [Ying et al. 04]
- Find how to carry out similar tasks or figure out a starting point [Cubranic and Murphy 04]
- Find code examples [Homes and Murphy 05]
- Infer task structure [Kersten and Murphy 05] [DeLine et al. 05]
- Find who should fix this bug [Anvik et al. 05]
- Prove or disprove conventional wisdom about development

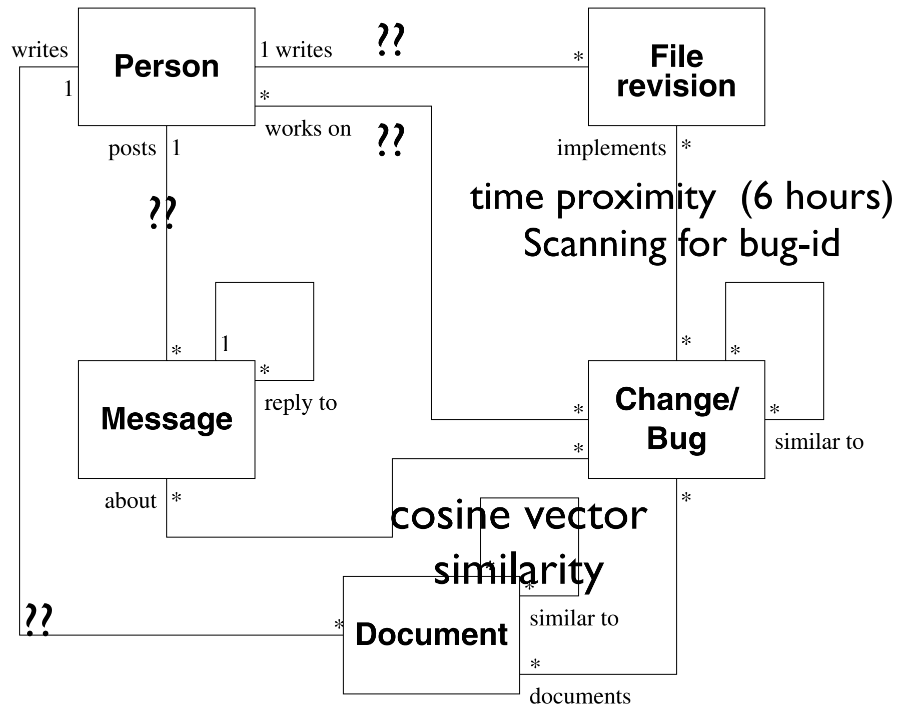
Hipikat

- Motivation: Newcomers to open source projects often rely on heterogeneous software artifact archives to gain implicit group memory (knowledge) about software.
- Hipikat is a recommender system that suggests relevant existing artifacts.

Hipikat Approach

1. Hipikat infers links between the artifacts that may have been apparent at one time to members of the development team but that were not recorded
2. It suggests relevant artifacts.

Associating Artifacts



Evaluation

1. Initial Qualitative Study

2. Case Study

Initial Qualitative Study

- What type of a user study is this?
- What is the purpose of this study?
- Participants:
  - Why did they group subjects into pairs?

Initial Qualitative Study

- Task Design:
  - Which tasks were chosen and why?
  - Why did they randomize the assignment of tools to the changes?
  - Why did they randomize the order in which they asked the pairs to make the change?

Initial Qualitative Study

- Analysis of the comments in the reports + Interview six subjects
- What did they learn from this study?
    - Programmers would like to understand *rationale* of the tool's suggestions.
    - Automatic suggestion => query-based interface

Case Study

- Participant?
- Which task was chosen and why?
  - They chose a completed enhancement to compare their solution with the solution by the Eclipse team.
  - It is somewhat surprising to me that there was a very similar change to this task in Eclipse history.

My general thoughts on Hipikat

- Pro: Hipikat addresses a very important, practical problem using a straightforward approach.
- Con: Hipikat needs to be instantiated for each system
- A clever evaluation: initial assessment => in-depth case study
- An integration & infrastructure implementation focused research