Speculative analysis and vision brainstorm

CMPSCI 521/621
UMass Amherst, Fall 2012

Any questions about last time?

Mutation testing
Model inference, checking
Bug localization
Symbolic execution

Project ideas assignment

- Due Tuesday Sept 18, 9 AM
  — short write up (no more than 1 page)
- Work individually or in groups of 2
- Presentation (be ready to go Tuesday 9/18)
- Submit 2 things:
  — write up
  — slides


Project idea must include:

- A research question
- The key idea behind technique / tool / experiment
- Evaluation plan

- You are not graded on whether your idea is selected.

Plan for today

- I’ll describe speculative analysis (a dynamic and static analysis technique)
- I’ll demonstrate a couple of speculative analysis tools (examples of past projects)
- You’ll brainstorm (with your neighbors) possible project ideas and we’ll discuss some

Any questions?
Implement a new feature?
Incorporate another developer’s changes?
Fix a bug?
Upgrade a library?
Refactor for code reuse?
Run tests?

Can we predict the future to help make decisions?

Speculative analysis: predict the future and analyze it
Speculative analysis: predict the future and analyze it

current program

speculate

refactor

Quick Fix Scout

Collaborators: Kıvanç Muşlu, Reid Holmes, Michael D. Ernst, and David Notkin

Eclipse provides Quick Fixes to resolve compilation errors.
But Eclipse can't tell which fix is best.

We can speculatively apply each fix to find out how many errors remain.

Sometimes, local fixes cannot resolve an error.

Speculation can discover remote fixes that resolve errors.

Complex error dependencies

http://quick-fix-scout.googlecode.com

http://quick-fix-scout.googlecode.com
Complex error dependencies

```java
public class ExceptionObject {
    public void exceptionMethod() {
        throw new Exception();
    }
}

public class SafeObject {
    public void safeMethod() {
        try {
            ExceptionObject eo = new ExceptionObject();
            eo.exceptionMethod();
        } catch (MyException e) {
            // (1) fix/eliminate declaration of exception
            // (2) replace catch clause with throw
            // (3) remove catch clause
        }
    }
}
```

http://quick-fix-scout.googlecode.com

Speculative analysis for Quick Fix

- Speculative analysis is predictive.

Continuous development
- compilation [Childers et al. 2003; Eclipse 2011]
- execution [Henderson and Weiser 1985; Karinthi and Weiser 1987]
- testing [Saff and Ernst 2003, 2004]
- version control integration [Guimarães and Rito-Silva 2010]

Exploring the future
Proactive detection of collaboration conflicts

Collaborators: Reid Holmes, Michael D. Ernst, and David Notkin

Version-control terminology

Proactive conflict detection applies to both centralized and distributed version control.

<table>
<thead>
<tr>
<th>distributed (hg, git)</th>
<th>centralized (cvs, svn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>local commit:</td>
<td>commit</td>
</tr>
<tr>
<td>incorporate:</td>
<td>pull and push</td>
</tr>
<tr>
<td></td>
<td>save</td>
</tr>
<tr>
<td></td>
<td>update and commit</td>
</tr>
</tbody>
</table>

The Gates conflict

The information was all there, but the developers didn’t know it.
The information was all there, but the developers didn't know it.
The Gates conflict

The information was all there, but the developers didn’t know it.

What could well-informed developers do?

- avoid conflicts

What could well-informed developers do?

- avoid conflicts
- become aware of conflicts earlier

Introducing Crystal: a proactive conflict detector

DEMO

Introducing Crystal: a proactive conflict detector

DEMO

Speculative analysis in collaborative development

local commit
speculate
incorporate from master
incorporate from Melinda
incorporate to master

current program
analyze
merge
collaborative relationships

test
inform developer

http://crystalvc.googlecode.com
Reducing false positives in conflict prediction

Collaborative awareness
- Palantir [Sarma et al. 2003]
- FASTDash [Biehl et al. 2007]
- Syde [Hattori and Lanza 2010]
- CollabVS [Dewan and Hegde 2007]
- Safe-commit [Wloka et al. 2009]
- SourceTree [Streeting 2010]

Crystal analyzes concrete artifacts, eliminating false positives and false negatives.

Utility of conflict detection

- Are textual collaborative conflicts a real problem?
- Can textual conflicts be prevented?
- Do build and test collaborative conflicts exist?

Are textual collaborative conflicts a real problem?

Histories of 9 open-source projects:
- size: 26K–1.4M SLoC
- developers: 298
- versions: 140,000

Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

How frequent are textual conflicts?

16% of the merges have textual conflicts.

How long do textual conflicts persist?
Conflicts live a mean of 9.8 and median of 1.6 days.
The worst case was over a year.

How long do textually-safe merges persist?
Textually-safe merges live a mean of 11.0 and median of 1.9 days.
Are textual collaborative conflicts a real problem?

How frequent are textual conflicts?
16% of the merges have textual conflicts.

How long do textual conflicts persist?
Conflicts live a mean of 9.8 and median of 1.6 days. The worst case was over a year.

How long do textually-safe merges persist?
Textually-safe merges live a mean of 11.0 and median of 1.9 days.

Can textual conflicts be prevented?

Where do textual conflicts come from?
93% of textual conflicts developed from safe merges. The information Crystal computes can help prevent conflicts.
Can textual conflicts be prevented?

Where do textual conflicts come from?

93% of textual conflicts developed from safe merges.

The information Crystal computes can help prevent conflicts.

Do build and test collaborative conflicts exist?

<table>
<thead>
<tr>
<th>program</th>
<th>conflicts</th>
<th>safe merges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>textual</td>
<td>build</td>
</tr>
<tr>
<td>Git</td>
<td>17%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Perl5</td>
<td>8%</td>
<td>4%</td>
</tr>
<tr>
<td>Voldemort</td>
<td>17%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Does merged code fail to build or fail tests?

One in three conflicts are build or test conflicts.

Microsoft Beacon

- A centralized version control-based tool.
- Microsoft product groups are using Beacon to help identify conflicts earlier in the development process.

Next steps:
- Measure Crystal’s effect on conflict frequency and persistence
- Evaluate qualitative effects on user experience
- Identify what helps and what does not

Additional collaborators: Kvanç Muşlu, Christian Bird, Thomas Zimmermann

Contributions of speculative analysis

Expanding the space of speculative analysis

Identify a domain with:
- likely, automatable developer actions
- informative, efficient analyses
- inferable developer intent

Next speculations:
- automated fault removal
- target platform deployment
- test generation and augmentation

Improving developer awareness when making decisions
- compute precise, accurate information
- convert a pull mechanism to a push one
Expanding the space of speculative analysis

Identify a domain with:
- likely, automatable developer actions
- informative, efficient analyses
- inferable developer intent

Next speculations:
- automated fault removal
- target platform deployment
- test generation and augmentation
Expanding the space of speculative analysis

Identify a domain with:
- likely, automatable developer actions
- informative, efficient analyses
- inferable developer intent

Next speculations:
- automated fault removal
- target platform deployment
- test generation and augmentation
Automating decision making: self-adaptation

What part of that would have been a good project?

- **Idea 1: Quick Fix Scout**
  - Identify quick fixes as the speculative actions and compilation errors as the analysis function
  - Build the Eclipse plug-in
  - Do a small, qualitative evaluation on a few users

Research question: Does precomputing effects of quick fix suggestions affect developer behavior?

What part of that would have been a good project?

- **Idea 2: Conflict frequency**
  - Collect ~8 large, open-source, with-tests programs with their histories from [github.com](http://github.com)
  - Analyze the frequency and duration of textual, compile, and test conflicts

Research question: How often do textual, compile, and test conflicts occur in open-source development and how long do they last?

What part of that would have been a good project?

- **Idea 3: Crystal**
  - Identify version control operations as the speculative actions and conflicts as the analysis function
  - Build Crystal
  - Do a small, qualitative evaluation on a few users

Research question: Does precomputing conflicts and making developers aware of them reduce the frequency and duration of conflicts?

What part of that would have been a good project?

- **Idea 4: Quick Fix Scout evaluation**
  (suppose Quick Fix Scout already exists)
  - Perform a controlled experiment on ~40 developers (students well familiar with Eclipse)
  - Build an Eclipse plug in to log (record) developer actions
  - Give each developer a set of small programming tasks and ask him/her to resolve compilation errors
  - Half the time, have the developer use QFS, the other half, just QF
  - Measure task completion times and analyze it for statistical improvement

Research question: Does QFS reduce the time it takes developers to resolve compilation errors?
Brainstorm topic ideas

• Get a piece of paper
• Turn to your neighbor
• Discuss possible research projects
  – brainstorm, write down all kinds of crazy ideas
• When the ideas stop flowing, discuss the ones you like best in some depth
• Be prepared to tell us your most promising idea