Speculative Analysis

Homework 3
- Will cover model inference
- Wednesday, October 28

BRING A LAPTOP TO CLASS!

Literature review
- for those doing research projects
- due November 2nd
Developers often make decisions based on experience and intuition.

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

Speculative analysis: predict the future and analyze it

Speculative analysis: predict the future and analyze it

Speculative analysis: research questions

Are there domains for which speculative analysis is possible?

Can speculative analysis be made computationally feasible?

Can speculative analysis help, and not overwhelm, developers?

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.

http://quick-fix-scout.googlecode.com
Exploring the future

Speculative analysis for Quick Fix

- `public class ExceptionalObject { public void exceptionMethod() { throw new Exception(); } }`
- `public class SafeObject { public void safeMethod() { try { ExceptionalObject eo = new ExceptionalObject(); eo.exceptionMethod(); } catch (Exception e) { } }`

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

Continuous development
- version control integration [Guimarães and Rito-Silva 2010]
- testing [Saad and Ernst 2003, 2004]
- 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
Exploring the future

- past version of the program
- present version of the program
- future version of the program

Continuous development

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

Speculative analysis is predictive.

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.

- distributed (hg, git)
- local commit: commit
- incorporate: pull and push
- centralized (cvs, svn)
- save
- update and commit

The Gates conflict

- 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.

The Gates conflict

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

What could well-informed developers do?

- avoid conflicts
- become aware of conflicts earlier

Introducing Crystal: a proactive conflict detector

DEMO

http://crystalvc.googlecode.com
### Speculative analysis in collaborative development

- **Speculate**
  - local commit
  - incorporate from Melinda
- **Incorporate**
  - incorporate from master
  - incorporate to master
- **Current program**
- **Analyze**
  - merge
  - compile
  - test
- **Inform developer**
  - collaborative relationships

---

### Reducing false positives in conflict prediction

#### Collaborative awareness
- Palantir [Sarma et al. 2003]
- FASTDash [Biehl et al. 2007]
- Syde [Hattori and Lanza 2010]
- CollaborativeVS [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–1.4MSLoC
- **Developers**: 298
- **Versions**: 140,000

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

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>Per15</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

Improving developer awareness when making decisions
- Compute precise, accurate information
- Convert a pull mechanism to a push one

Additional collaborators: Kıvanç Muşlu, Christian Bird, Thomas Zimmermann
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
- code parallelization
- 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
- code parallelization
- test generation and augmentation
Automating decision making: self-adaptation

- Specification
- Generate adaptations
- Potential systems
- Running system
- Observe
- Analyze
- Employ adaptation
- Decide
- Future: understanding behavior

Future research: automation

- Automating decision making: removing the developer
- Using new automation to enrich speculative analysis
- Bridging requirement specification and behavioral model inference