Speculative Analysis

Homework 2
- On bug localization
- Due Tuesday (fake Monday), Oct 14, 9 AM on moodle

Research Projects
- Has everyone submitted project ideas / paper selections?
- Wednesday, we will have research idea presentations
- I will present several as well
- Students will have until Friday 10/10 to form groups
**DECISION MAKING**

Developers often make decisions based on experience and intuition.

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

- Current program

speculate
Speculative analysis: predict the future and analyze it

Speculate

current program

speculate

refactor

refactor

refactor

refactor

analyze

execute test suite

inform developer

# of resulting test failures

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
Complex error dependencies

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

```java
public class SafeObject {
    public void safeMethod() {
        try {
            ExceptionObject eo = new ExceptionObject();
            eo.exceptionMethod();
        } catch (MyException e) {
            // Handle exception
        }
    }
}
```

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

Speculative analysis for Quick Fix

Exploring the future

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

<table>
<thead>
<tr>
<th>past version of the program</th>
<th>present version of the program</th>
<th>future version of the program</th>
</tr>
</thead>
<tbody>
<tr>
<td>regression testing</td>
<td>continuous testing</td>
<td>speculative analysis</td>
</tr>
<tr>
<td>debugging</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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]

Speculative analysis is **predictive**.

### Version-control terminology

- **Proactive detection of collaboration conflicts**

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

- **The Gates conflict**

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

  ![Diagram](image-url)
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
- Incorporate from Melinda
- 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]
- 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–1.4MSLoC
- 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>Git</td>
<td>17%</td>
<td>4%</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>

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

Contributions of speculative analysis

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
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
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

employ adaptation

decide

Future research: automation

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