Midterm Review and Speculative Analysis

Course updates
- Homework 3 due April 18
- Project Plan due April 20*

* If you need more time (until April 24), just ask

Today’s plan
- Brief description of midterm + topics covered
- Lecture on speculative analysis

What’s the midterm like?
- True / False questions
- Multiple choice questions
- Some reasoning questions (also multiple choice)

Topics to be covered
- Dynamic analysis
  - Daikon and Purify
- Testing and automated test generation
  - Revealing domains, Korat, Chronicler, and BugRedux (field failures), mutation testing, delta debugging
- Testing
  - Coverage, revealing subdomains, black-box vs. glass, regression testing
- Automated program repair
  - How it works, what can go wrong
- Bias in software
  - Themis
- Speculative Analysis
  - Quick fix scout, Crystal, CodeHint, CodebaseReplication
- Visual data communication
- Machine Learning for Systems
- Provably sound synthesis for side-channel attacks
- Inclusive open-source projects
- Gradual verification
- Specification synthesis
- Trojans in DNNs
True / False Example
Automatically predicting collaboration conflicts, if applied properly, would eliminate the need for resolving conflicts, which would greatly improve software development productivity.

Multiple Choice Example
Rational Purify can find the following types of bugs (check all that apply):
A. Writing past the end of an array
B. Reading past the end of an array
C. Writing past the end of the first object in an array of objects
D. Null pointer exceptions
E. Using a different method than the developer intended

Reasoning
• Reasoning are the harder questions that require abstraction and application of what you learnt.
• Reasoning questions will largely cover the homework assignments
Implement a new feature?

Incorporate another developer’s changes?

Fix a bug?

Refactor for code reuse?

Upgrade a library?

Run tests?

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

- Current program
  - Speculate
  - Refactor
- Analyze
  - Execute test suite
  - Inform developer
    - # of resulting test failures

Are there domains for which speculative analysis is possible? Can speculative analysis be made computationally feasible? Can speculative analysis help, and not overwhelm, developers?
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.
Complex error dependencies

```java
public class ExceptionalObject {
    public void exceptionalMethod() {
        throw new MyException();
    }
}
```

```java
public class SafeObject {
    public void safeMethod() {
        try {
            ExceptionalObject eo =
                new ExceptionalObject();
            eo.exceptionalMethod();
        } catch (MyException e) {
        }
    }
}
```

https://github.com/brunyuriy/quick-fix-scout
Complex error dependencies

```java
public class ExceptionalObject {
    public void exceptionalMethod() {
        throw new MyException();
    }
}
```

```java
public class SafeObject {
    public void safeMethod() {
        try {
            ExceptionalObject eo = new ExceptionalObject();
            eo.exceptionalMethod();
        } catch (MyException e) {
        }
    }
}
```

https://github.com/brunyuriy/quick-fix-scout
Complex error dependencies

```java
public class ExceptionalObject {
    public void exceptionalMethod() {
        throw new MyException();
    }
}
```

```java
public class SafeObject {
    public void safeMethod() {
        try {
            ExceptionalObject eo =
                new ExceptionalObject();
            eo.exceptionalMethod();
        } catch (MyException e) { }
    }
}
```

https://github.com/brunyuriy/quick-fix-scout
Speculative analysis for Quick Fix

- **speculate**
  - quick fix
  - quick fix
  - quick fix
  - quick fix
  - quick fix

- **current program**

- **analyze**
  - compile

- **inform developer**
  - # of resulting compilation errors
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]

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.

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

https://github.com/brunyuriy/crystalvc
Speculative analysis in collaborative development

Speculate
- local commit
- incorporate from Melinda
- incorporate from master
- incorporate to master

Current program

Analyze
- merge
- compile
- test
- ...

Inform developer
- collaborative relationships
Reducing false positives in conflict prediction

Collaborative awareness

- Palantír [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?

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.
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.
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>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: Kivanç Muşlu, Christian Bird, Thomas Zimmermann
Contributions of speculative analysis

- Past version of the program:
  - Mining software repositories
  - Regression testing

- Present version of the program:
  - Delta debugging
  - Continuous testing

- Future version of the program:
  - Automated debugging
  - Speculative analysis

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
- 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
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

- **generate adaptations**
- **running system**
- **observe**
- **decide**
- **employ adaptation**
- **potential systems**
- **analysis**
- **specification**
Future research: automation

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


