Speculative Analysis

Course updates
• Project plan assignment due April 11
• Midterm next week, April 13
• We’ll start review today, finish next Tuesday
• Homework 4 extended until April 20, 9 AM EDT

Today’s plan
• Brief description of midterm + topics covered
• Lecture on speculative analysis
  (last lecture covered by midterm)

What’s the midterm like?
• Some true/false questions
• Some multiple choice questions
• Some reasoning questions

On Tuesday
• we’ll do some sample questions
• I’ll let you ask questions about midterm topics
• if (more questions)
  answer questions
  else
  talk about software architecture

Topics to be covered
• Dynamic analysis
  – Daikon and Purify
• Software development lifecycle
  – ad hoc, code and fix, waterfall, spiral, staged, scrum
• Testing and automated test generation
  – revealing domains, Korat, Chronicler and BugRedux (field failures), SPLat,
    mobile testing and recovery, mutation testing, delta debugging
Topics to be covered

• Software privacy and reliability
  – SoZware and smart redundancy

• Automated Bug Fixing
  – redundant methods, GenProg, Par, staged repair,
    SemFix, DirectFix, Angelix, ClearView,
    app method substitutions, program boosting (crowd)
    quality of repair

• Speculative Analysis
  – Quick fix scout, Crystal, CodeHint,
    CodebaseReplication

• Refactoring
DECISION MAKING
Implement a new feature?

Incorporate another developer’s changes?

Fix a bug?

**DECISION MAKING**

Upgrade a library?

Refactor for code reuse?

Run tests?
Decision making

Implement a new feature?

Incorporate another developer’s changes?

Fix a bug?

Upgrade a library?

Refactor for code reuse?

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
Speculative analysis: predict the future and analyze it

speculate

current program
Speculative analysis: predict the future and analyze it

- 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: predict the future and analyze it

- 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: predict the future and analyze it

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: predict the future and analyze it

- Speculative analysis is possible in certain domains.
- Computationally feasible with the right tools and methods.
- Can help developers by providing insights without overwhelming them.
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?
Quick Fix Scout

Collaborators: Kivanç 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) {
        }
    }
}
```

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

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

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

http://quick-fix-scout.googlecode.com
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) {}
    }
}
```

http://quick-fix-scout.googlecode.com
Speculative analysis for Quick Fix

speculate

current program

analyze
compile

inform developer
# of resulting compilation errors

Decision making
Quick Fix Scout
Crystal
Future: understanding behavior
Exploring the future

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

- delta debugging
- continuous testing
- automated debugging

Continuous development

Compilation [Childers et al. 2003; Eclipse 2011]

Execution [Henderson and Weiser 1985; Karinthi and Weiser 1987]

Testing [Saand Ernst 2003, 2004]

Version control integration [Guimarães and Rito-Silva 2010]
Exploring the future

past version of the program

present version of the program

future version of the program

- mining software repositories
- regression testing
- delta debugging
- continuous testing
- automated debugging
Exploring the future

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

- mining
- regression testing
- delta debugging
- continuous testing
- automated debugging

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

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
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>commit</td>
<td>pull and push</td>
<td>update and commit</td>
</tr>
<tr>
<td>incorporate</td>
<td>commit</td>
<td>save</td>
</tr>
</tbody>
</table>
The Gates conflict
The Gates conflict
The Gates conflict

![Diagram with M, T, and W labels]

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

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

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

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

- **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]
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.

Conflicts live a mean of 9.8 and median of 1.6 days.
The worst case was over a year.

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.
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.

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.
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?
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.
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: Kıvanç 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
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
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
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

running system
Automating decision making: self-adaptation

- **specification**
- **generate adaptations**
- **potential systems**
- **running system**
Automating decision making: self-adaptation

generate adaptations

running system

observe

specification

potential systems

analysis
Automating decision making: self-adaptation

- **specification**
- **potential systems**
- **running system**
- **observe**
- **employ adaptation**
- **decide**
- **analysis**
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


