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

http://people.cs.umass.edu/~brun/class/CS521.621/ideaProposal.pdf

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.

Any questions?

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

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?

Implement a new feature?

Incorporate another developer's changes?

Fix a bug?

DECISION MAKING

Developers often make decisions based on experience and intuition.

Upgrade a library?

Refactor for code reuse?

Run tests?

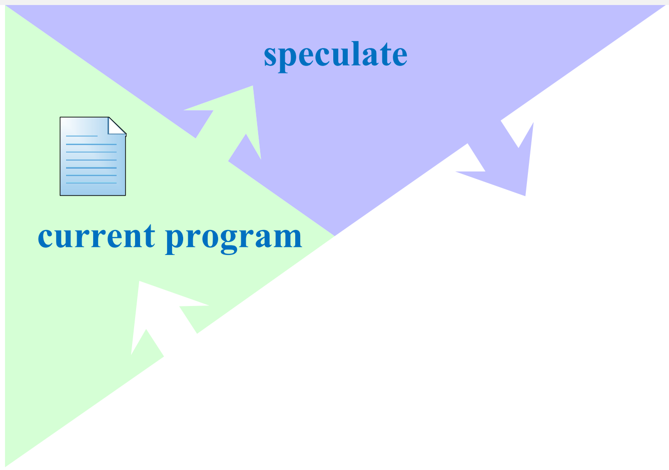
Can we predict the future

to help make decisions?

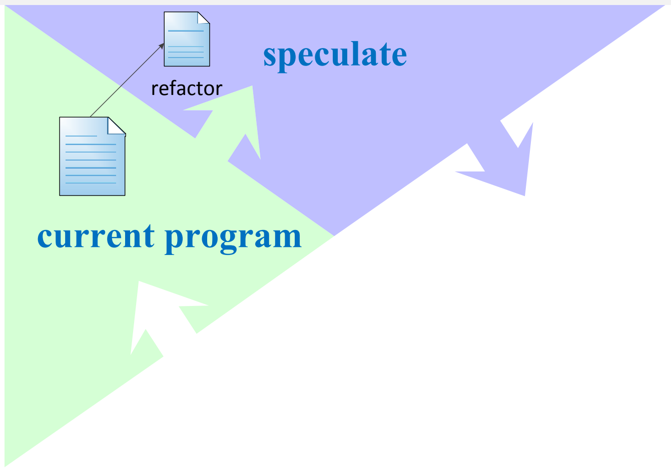
Speculative analysis: predict the future and analyze it

current program

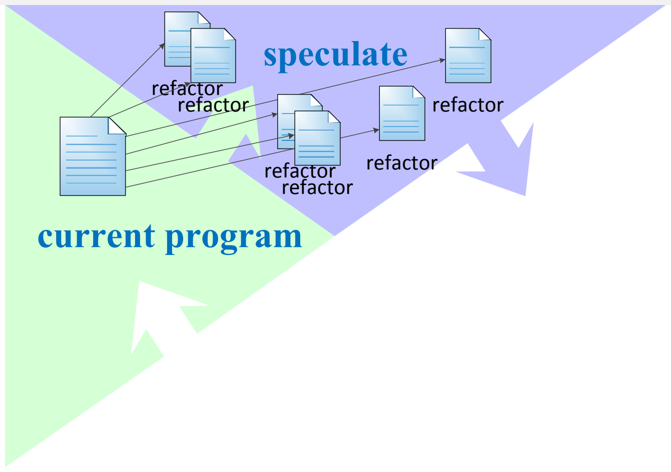
Speculative analysis: predict the future and analyze it



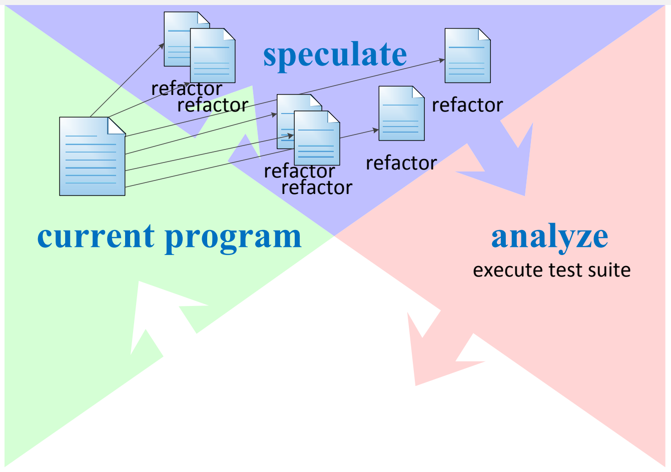
Speculative analysis: predict the future and analyze it



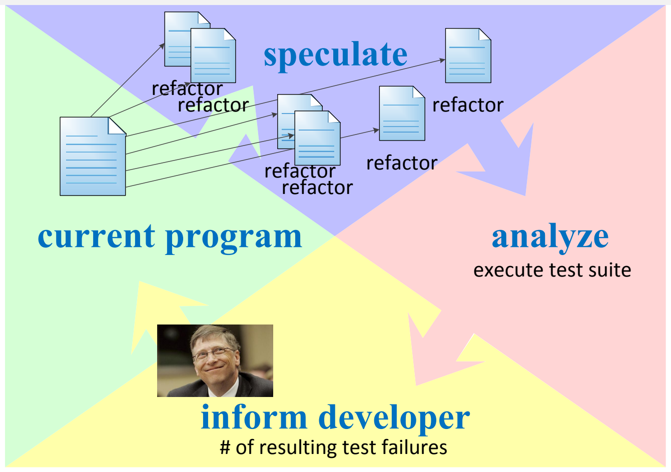
Speculative analysis: predict the future and analyze it



Speculative analysis: predict the future and analyze it

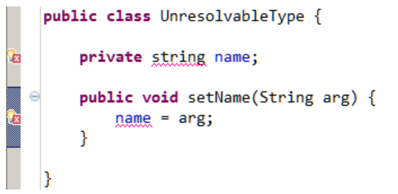


Speculative analysis: predict the future and analyze it



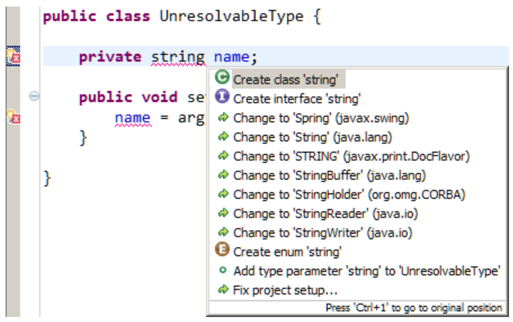
Quick Fix Scout

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



```
public class UnresolvableType {

    private string name;

    public void setName(String arg) {
        name = arg;
    }

}
```

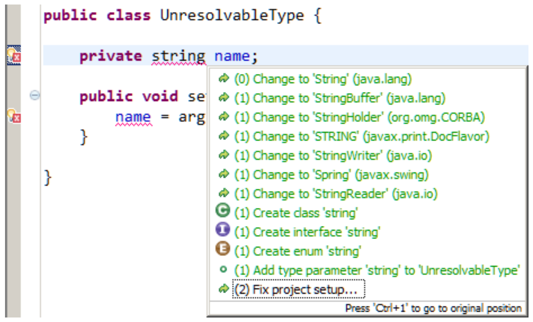
Eclipse provides Quick Fixes to resolve compilation errors.



```
public class UnresolvableType {

    private string name;

    public void se
        name = arg
    }

}
```

Create class 'string'
Create interface 'string'
Change to 'Spring' (javax.swing)
Change to 'String' (java.lang)
Change to 'STRING' (javax.print.DocFlavor)
Change to 'StringBuffer' (java.lang)
Change to 'StringHolder' (org.omg.CORBA)
Change to 'StringReader' (java.io)
Change to 'StringWriter' (java.io)
Create enum 'string'
Add type parameter 'string' to 'UnresolvableType'
Fix project setup...
Press 'Ctrl+1' to go to original position

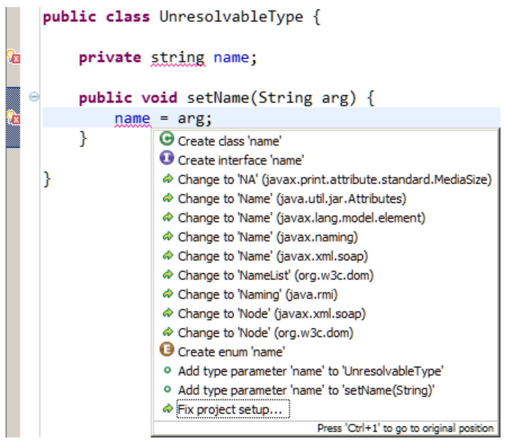
But Eclipse can't tell which fix is best.



```
public class UnresolvableType {

    private string name;

    public void se
        name = arg
    }

}
```

(0) Change to 'String' (java.lang)
(1) Change to 'StringBuffer' (java.lang)
(1) Change to 'StringHolder' (org.omg.CORBA)
(1) Change to 'STRING' (javax.print.DocFlavor)
(1) Change to 'StringWriter' (java.io)
(1) Change to 'Spring' (javax.swing)
(1) Change to 'StringReader' (java.io)
(1) Create class 'string'
(1) Create interface 'string'
(1) Create enum 'string'
(1) Add type parameter 'string' to 'UnresolvableType'
(2) Fix project setup...
Press 'Ctrl+1' to go to original position

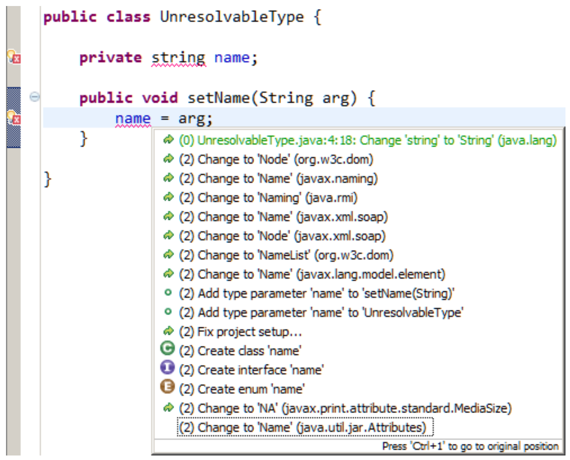
We can speculatively apply each fix to find out how many errors remain.



```
public class UnresolvableType {

    private string name;

    public void setName(String arg) {
        name = arg;
    }

}
```

Create class 'name'
Create interface 'name'
Change to 'NA' (javax.print.attribute.standard.MediaSize)
Change to 'Name' (java.util.jar.Attributes)
Change to 'Name' (javax.lang.model.element)
Change to 'Name' (javax.naming)
Change to 'Name' (javax.xml.soap)
Change to 'NameList' (org.w3c.dom)
Change to 'Naming' (java.rmi)
Change to 'Node' (javax.xml.soap)
Change to 'Node' (org.w3c.dom)
Create enum 'name'
Add type parameter 'name' to 'UnresolvableType'
Add type parameter 'name' to 'setName(String)'
Fix project setup...
Press 'Ctrl+1' to go to original position

Sometimes, local fixes cannot resolve an error.

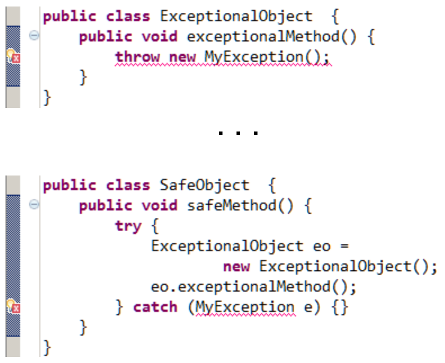


```
public class UnresolvableType {

    private string name;

    public void setName(String arg) {
        name = arg;
    }

}
```

(0) UnresolvableType.java:4:18: Change 'string' to 'String' (java.lang)
(2) Change to 'Node' (org.w3c.dom)
(2) Change to 'Name' (javax.naming)
(2) Change to 'Naming' (java.rmi)
(2) Change to 'Name' (javax.xml.soap)
(2) Change to 'Node' (javax.xml.soap)
(2) Change to 'NameList' (org.w3c.dom)
(2) Change to 'Name' (javax.lang.model.element)
(2) Add type parameter 'name' to 'setName(String)'
(2) Add type parameter 'name' to 'UnresolvableType'
(2) Fix project setup...
(2) Create class 'name'
(2) Create interface 'name'
(2) Create enum 'name'
(2) Change to 'NA' (javax.print.attribute.standard.MediaSize)
(2) Change to 'Name' (java.util.jar.Attributes)
Press 'Ctrl+1' to go to original position

Speculation can discover remote fixes that resolve errors.

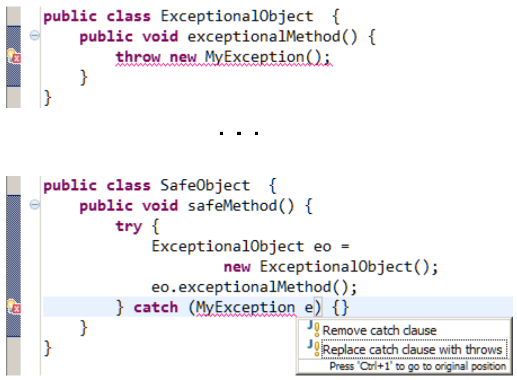
Complex error dependencies



```
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

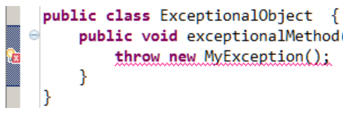


```
public class ExceptionalObject  {
    public void exceptionalMethod() {
        throw new MyException();
    }
}

                ...

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

Remove catch clause
Replace catch clause with throws
Press 'Ctrl+1' to go to original position

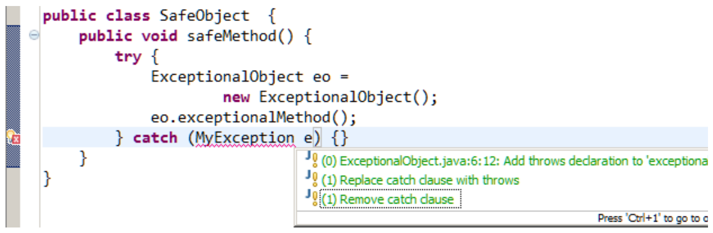
http://quick-fix-scout.googlecode.com

Complex error dependencies



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

. . .

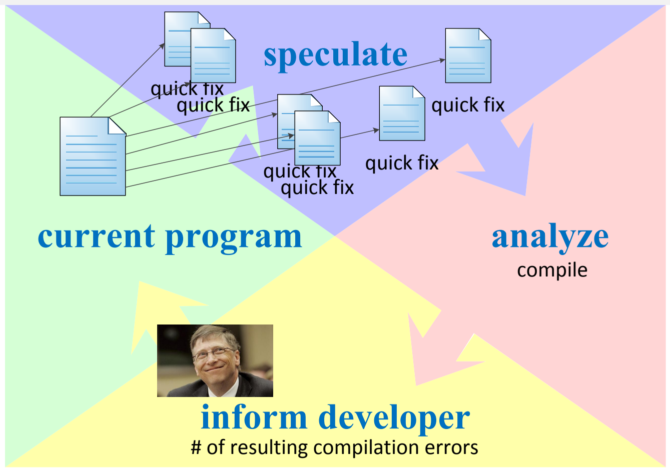


```
public class SafeObject  {
    public void safeMethod() {
        try {
            ExceptionalObject eo =
                    new ExceptionalObject();
            eo.exceptionalMethod();
        } catch (MyException e) {}
    }
}
```

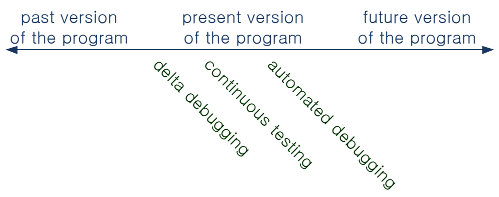
(0) ExceptionalObject.java:6:12: Add throws declaration to 'exceptiona
(1) Replace catch clause with throws
(1) Remove catch clause
Press 'Ctrl+1' to go to o

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

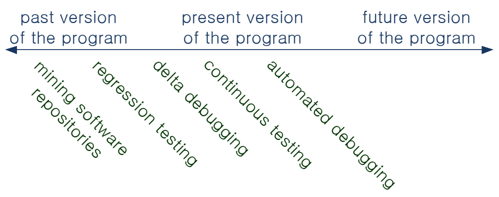
Speculative analysis for Quick Fix



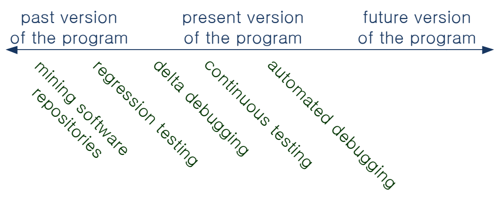
Exploring the future


past version of the program
present version of the program
future version of the program
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 software repositories
regression testing
delta debugging
continuous testing
automated debugging

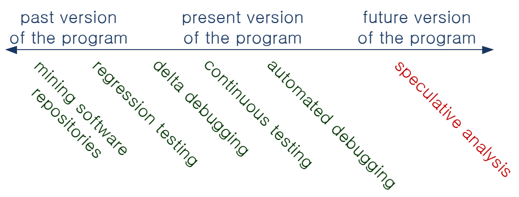
Exploring the future


past version of the program
present version of the program
future version of the program
mining software repositories
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


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
speculative analysis

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

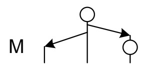
Collaborators: Reid Holmes, Michael D. Ernst, and David Notkin

Version-control terminology

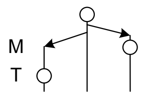
Proactive conflict detection applies to both centralized and distributed version control.

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

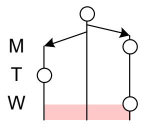
The Gates conflict



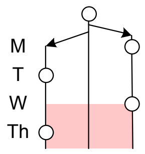
The Gates conflict



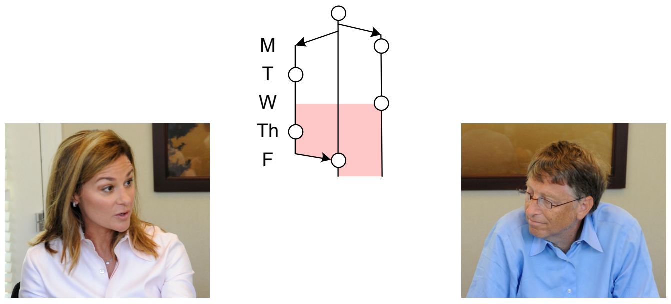
The Gates conflict



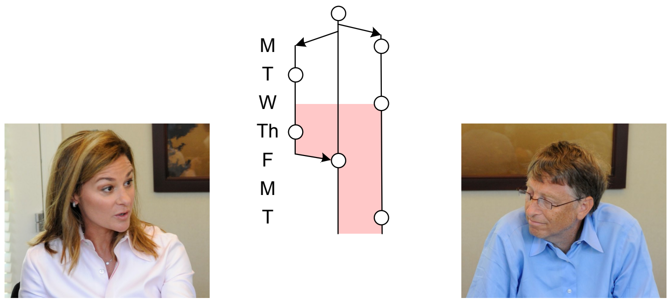
The Gates conflict



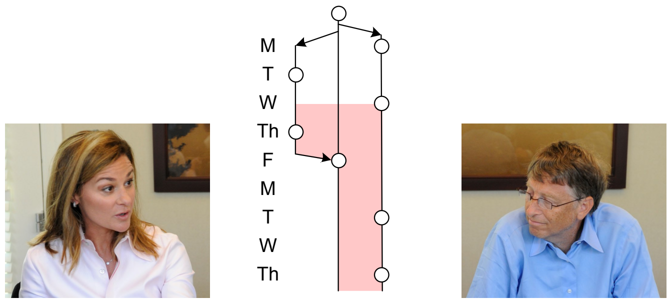
The Gates conflict



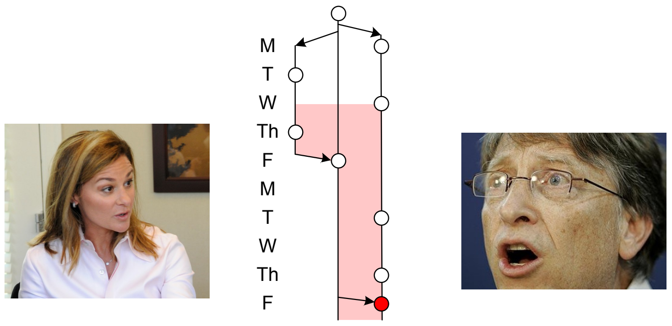
The Gates conflict



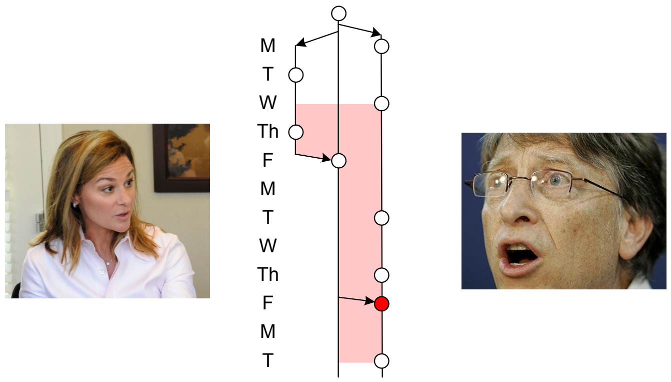
The Gates conflict



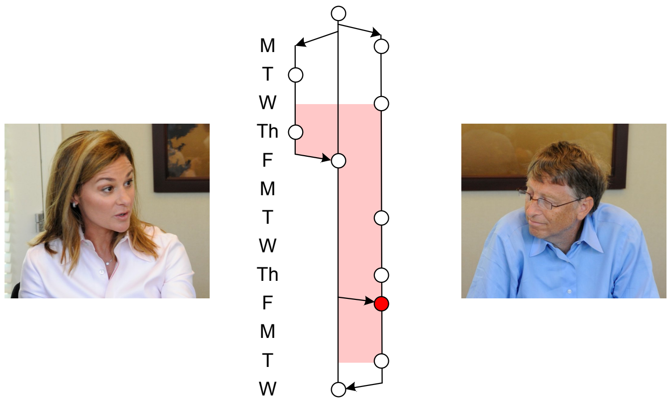
The Gates conflict



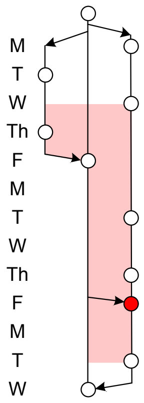
The Gates conflict



The Gates conflict

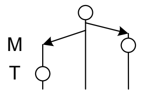


The Gates conflict



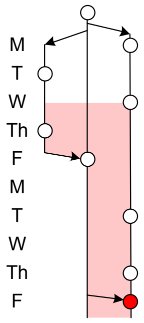
The information was all there, but the developers didn't know it.

What could well-informed developers do?



- avoid conflicts

What could well-informed developers do?



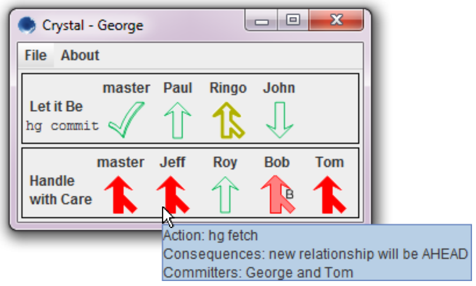
- avoid conflicts
- become aware of conflicts earlier

Introducing Crystal: a proactive conflict detector

DEMO

Introducing Crystal: a proactive conflict detector

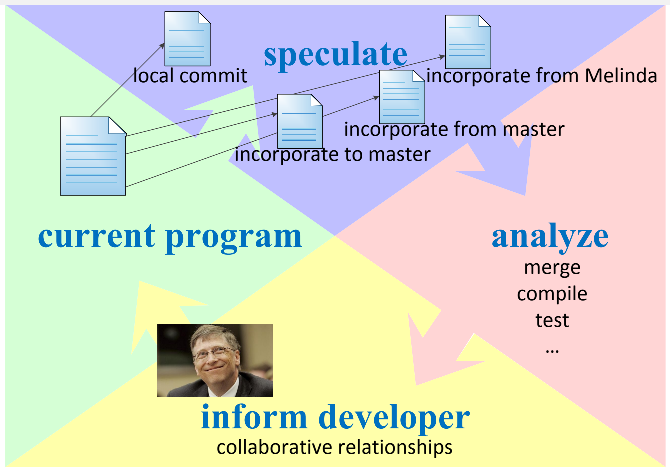
DEMO



Action: hg fetch
Consequences: new relationship will be AHEAD
Committers: George and Tom

http://crystalvc.googlecode.com

Speculative analysis in collaborative development



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?

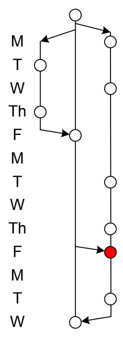
Are textual collaborative conflicts a real problem?

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?

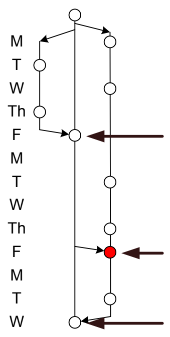


histories of 9 open-source projects:

size:	26K–1.4MSLoC
developers:	298
versions:	140,000

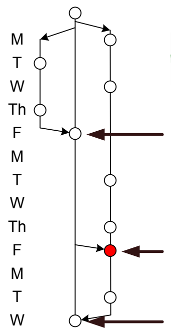
Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

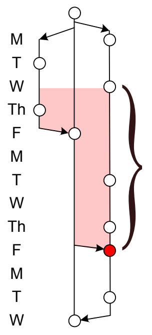
Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

16% of the merges have textual conflicts.

Are textual collaborative conflicts a real problem?

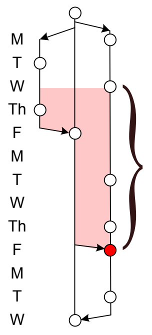


How frequent are textual conflicts?

16% of the merges have textual conflicts.

How long do textual conflicts persist?

Are textual collaborative conflicts a real problem?



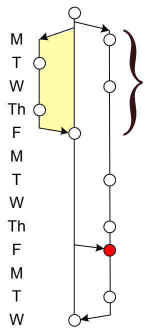
How frequent are textual conflicts?

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?

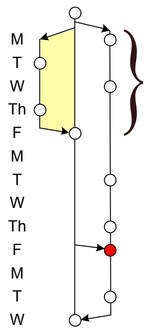
16% of the merges have textual conflicts.

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.

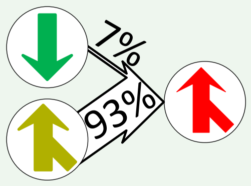
Can textual conflicts be prevented?

Where do textual conflicts come from?

Can textual conflicts be prevented?

Where do textual conflicts come from?

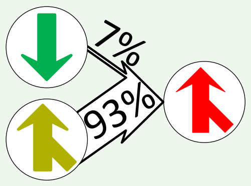
93% of textual conflicts developed from safe merges.



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?

program	conflicts			safe
	textual	build	test	merges
Git	17%	<1%	4%	79%
Perl5	8%	4%	28%	61%
Voldemort	17%	10%	3%	69%

Does merged code fail to build or fail tests?

One in three conflicts are build or test conflicts.

Microsoft Beacon

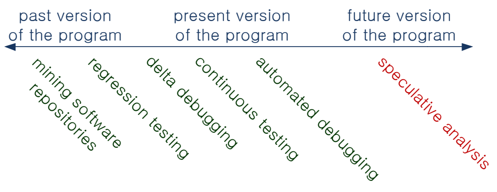
- A centralized version control-based tool.
- Microsoft product groups are using Beacon to help identify conflicts earlier in the development process.

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
present version of the program
future version of the program
mining software repositories
regression testing
delta debugging
continuous testing
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

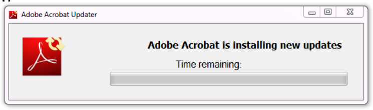
Next speculations:

- automated fault removal
- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining:

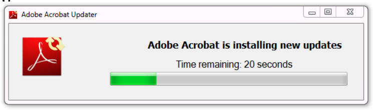
Next speculations:

- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining: 20 seconds

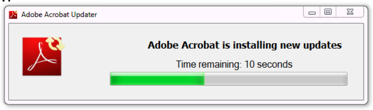
Next speculations:

- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining: 10 seconds

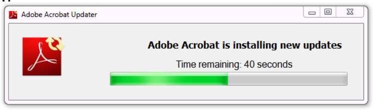
Next speculations:

- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining: 40 seconds

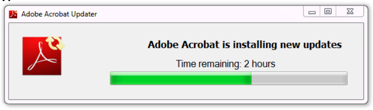
Next speculations:

- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining: 2 hours

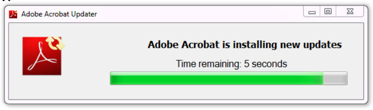
Next speculations:

- 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 inter


Adobe Acrobat Updater

Adobe Acrobat is installing new updates

Time remaining: 5 seconds

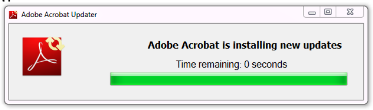
Next speculations:

- 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 inten



Adobe Acrobat is installing new updates

Time remaining: 0 seconds

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

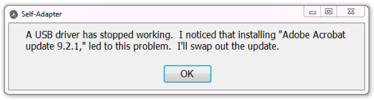
Next speculations:

- automated fault removal
- target platform deployment
- test generation and augmentation

Expanding the space of speculative analysis

Identify a domain with:

- likely, automatable developer actions
- informative, efficient analyses
- inferab



A USB driver has stopped working. I noticed that installing "Adobe Acrobat update 9.2.1," led to this problem. I'll swap out the update.

Next speculations:

- automated fault removal
- target platform deployment
- test generation and augmentation

Expanding the space of speculative analysis

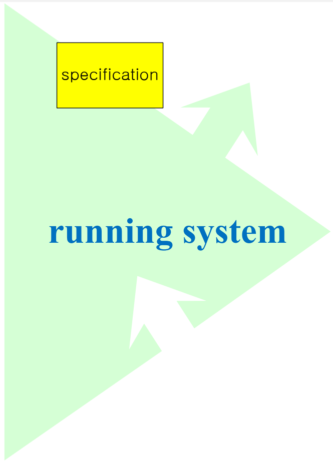
Identify a domain with:

- likely, automatable developer actions
- informative, efficient analyses
- inferable developer intent

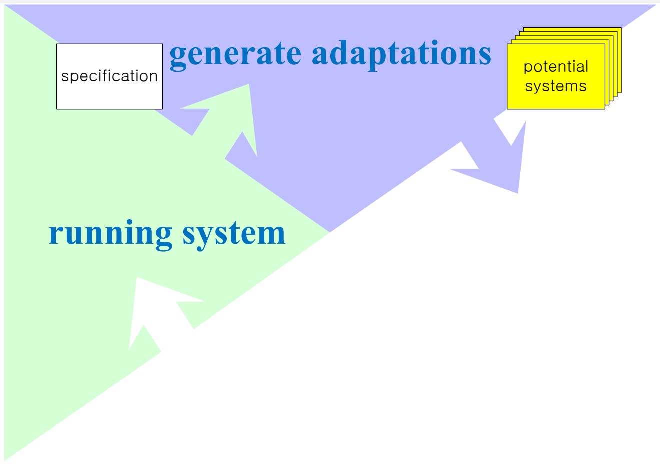
Next speculations:

- automated fault removal
- target platform deployment
- test generation and augmentation

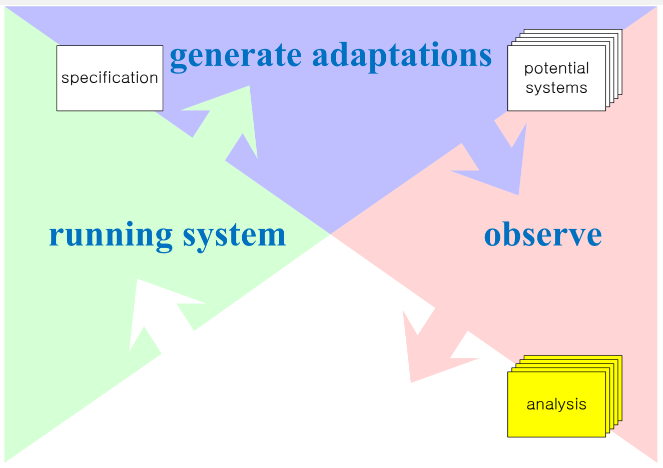
Automating decision making: self-adaptation



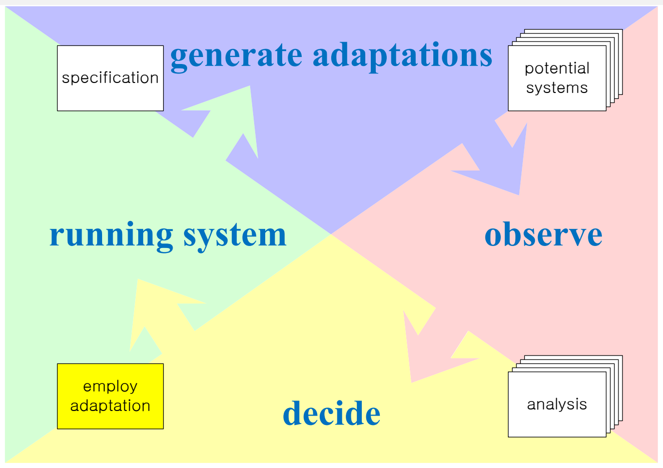
Automating decision making: self-adaptation



Automating decision making: self-adaptation



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
  - 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