

Speculative Analysis

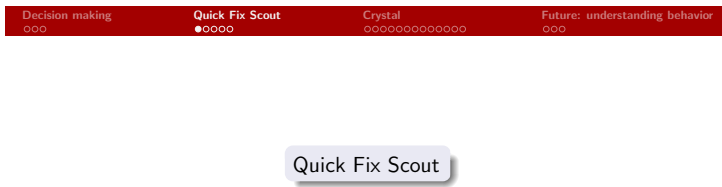
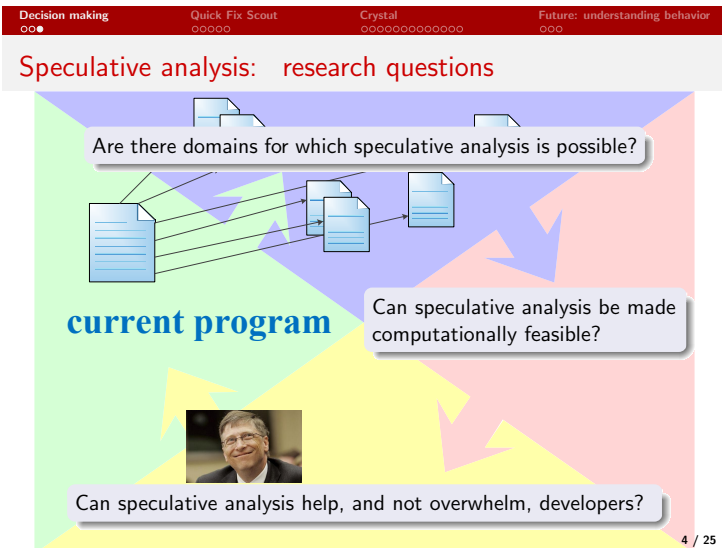
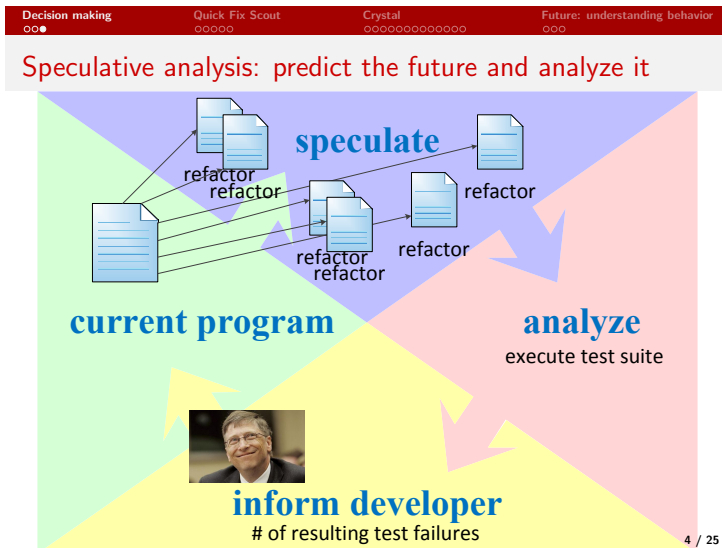
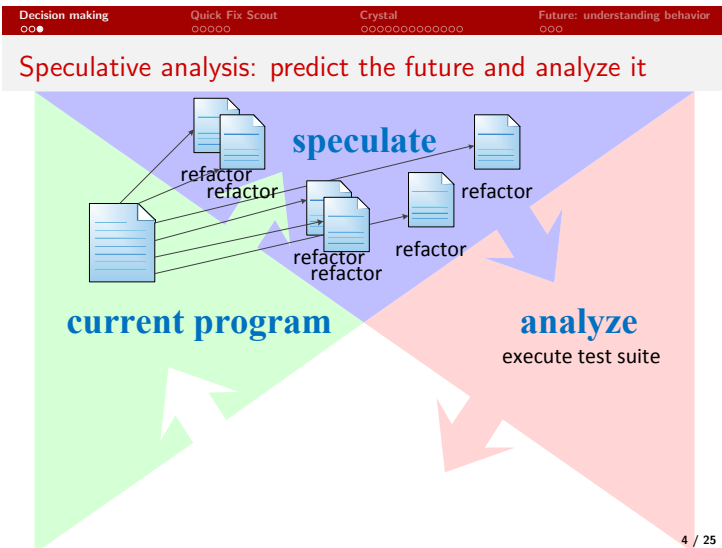
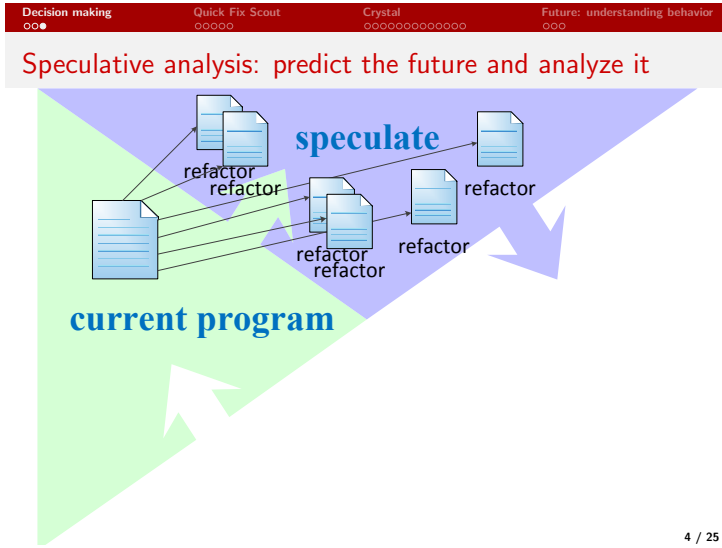
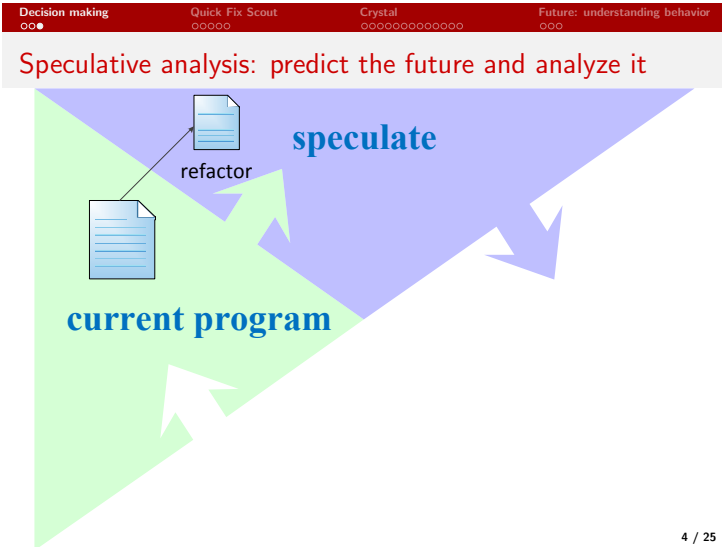
Homework 3

- Will cover model inference
- Wednesday, October 28

BRING A LAPTOP TO CLASS!

Literature review

- for those doing research projects
- due November 2nd



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

```

Decision making Quick Fix Scout Crystal Future: understanding behavior
000 0000 0000000000000000 000
public class UnresolvableType {

    private String name;

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

```

```

Decision making Quick Fix Scout Crystal Future: understanding behavior
000 0000 0000000000000000 000
public class UnresolvableType {

    private String name;

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

```

- Create class 'string'
- Create interface 'string'
- Change to 'String' (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

Eclipse provides Quick Fixes to resolve compilation errors.

But Eclipse can't tell which fix is best.

```

Decision making Quick Fix Scout Crystal Future: understanding behavior
000 0000 0000000000000000 000
public class UnresolvableType {

    private String name;

    public void setName(String arg) {
        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 'String' (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

```

Decision making Quick Fix Scout Crystal Future: understanding behavior
000 0000 0000000000000000 000
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

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

Sometimes, local fixes cannot resolve an error.

```

Decision making Quick Fix Scout Crystal Future: understanding behavior
000 0000 0000000000000000 000
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

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) {}
    }
}

```

Speculation can discover remote fixes that resolve errors.

<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) {}
    }
}
    
```

J Remove catch clause
 J 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();
    }
}

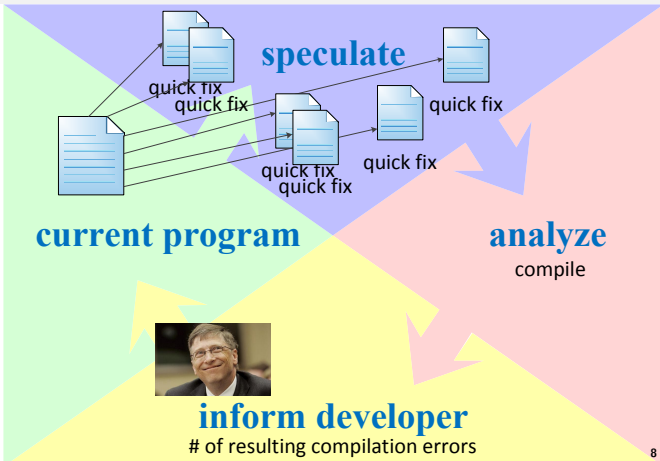
...

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

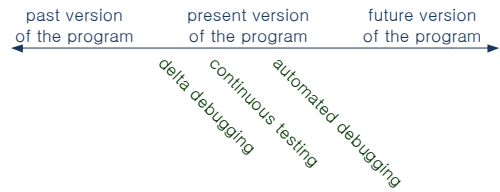
J (0) ExceptionalObject.java:6:12: Add throws declaration to 'exceptionalMethod()'
 J (1) Replace catch clause with throws
 J (4) Remove catch clause
 Press 'Ctrl+1' to go to original position

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

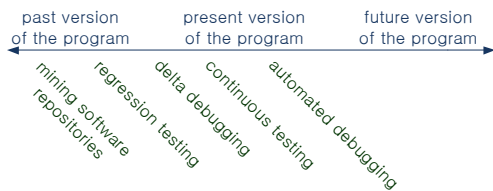
Speculative analysis for Quick Fix



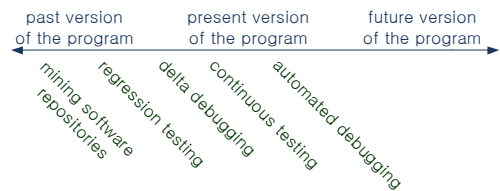
Exploring the future



Exploring the future



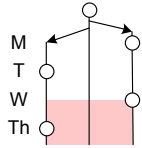
Exploring the future



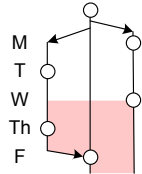
Continuous development

- 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]

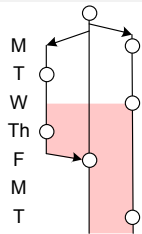
The Gates conflict



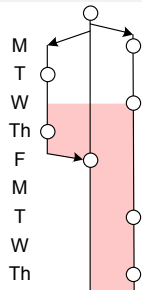
The Gates conflict



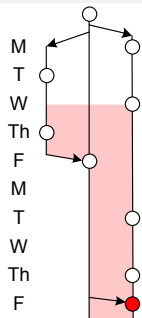
The Gates conflict



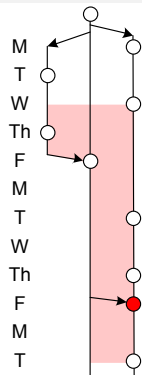
The Gates conflict



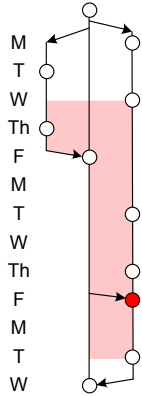
The Gates conflict



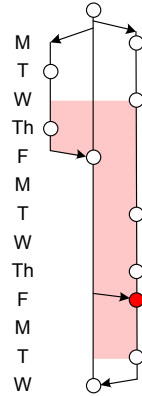
The Gates conflict



The Gates conflict

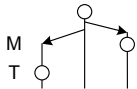


The Gates conflict



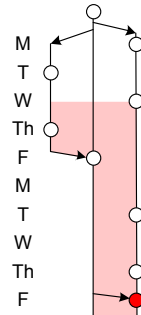
The information was all there, but the developers didn't know it.

What could well-informed developers do?



- avoid conflicts

What could well-informed developers do?



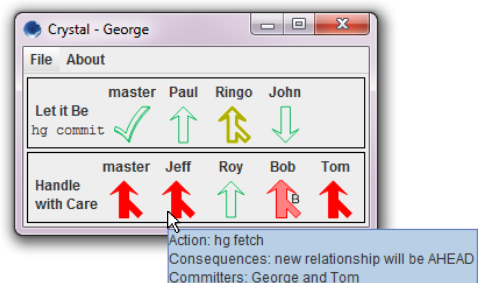
- avoid conflicts
- become aware of conflicts earlier

Introducing Crystal: a proactive conflict detector

DEMO

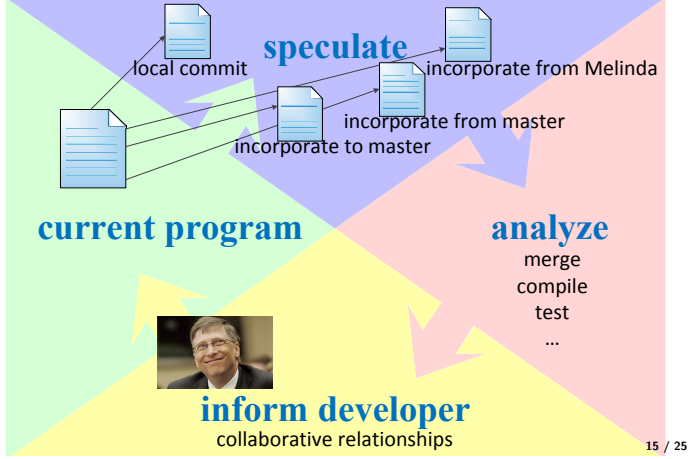
Introducing Crystal: a proactive conflict detector

DEMO



<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.

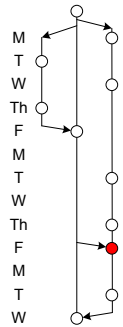
Utility of conflict detection

- 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?

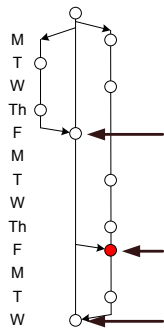
histories of 9 open-source projects:
 size: 26K–1.4MSLoC
 developers: 298
 versions: 140,000
 Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

Are textual collaborative conflicts a real problem?



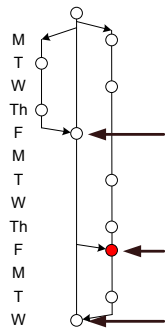
histories of 9 open-source projects:
 size: 26K–1.4MSLoC
 developers: 298
 versions: 140,000
 Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

Are textual collaborative conflicts a real problem?



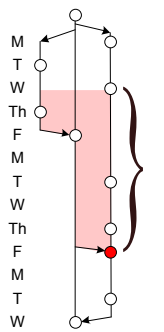
How frequent are textual conflicts?
[Empty box]

Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?
16% of the merges have textual conflicts.

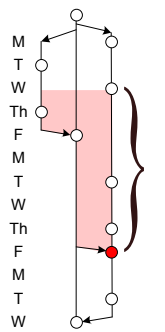
Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?
16% of the merges have textual conflicts.

How long do textual conflicts persist?
[Empty box]

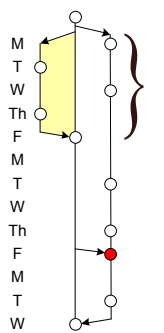
Are textual collaborative conflicts a real problem?



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?

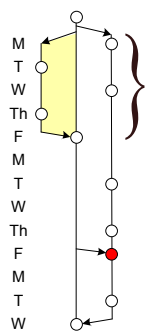


How frequent are textual conflicts?
16% of the merges have textual conflicts.

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?
[Empty box]

Are textual collaborative conflicts a real problem?



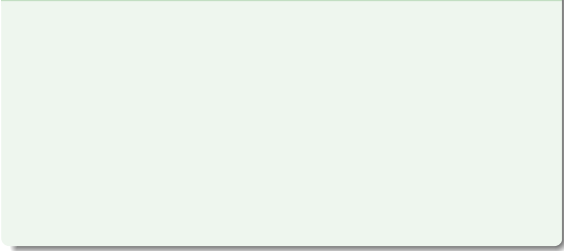
How frequent are textual conflicts?
16% of the merges have textual conflicts.

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?

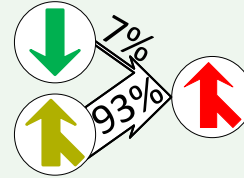


19 / 25

Can textual conflicts be prevented?

Where do textual conflicts come from?

93% of textual conflicts developed from safe merges.

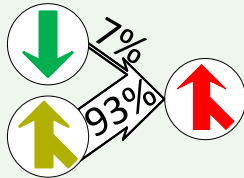


19 / 25

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.

19 / 25

Do build and test collaborative conflicts exist?

program	conflicts			safe merges
	textual	build	test	
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.

20 / 25

Microsoft Beacon

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

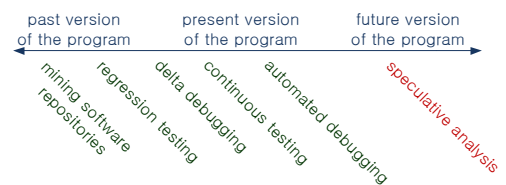
Next steps:

- 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

21 / 25

Contributions of speculative analysis



Improving developer awareness when making decisions

- compute precise, accurate information
- convert a pull mechanism to a push one

22 / 25

Expanding the space of speculative analysis

Identify a domain with:

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

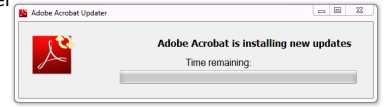
Next speculations:

- 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*



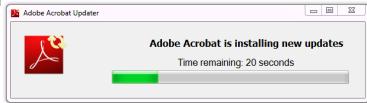
Next speculations:

- 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*



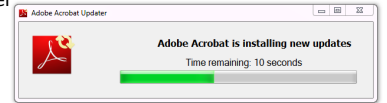
Next speculations:

- 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*



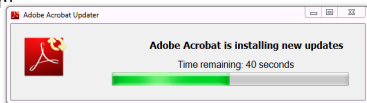
Next speculations:

- 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*



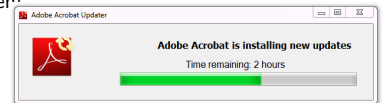
Next speculations:

- 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*



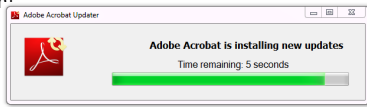
Next speculations:

- 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*



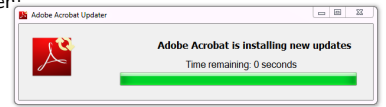
Next speculations:

- 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

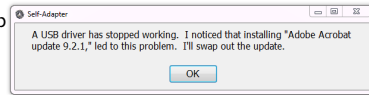
Next speculations:

- 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
- inferab



Next speculations:

- automated fault removal
- code parallelization
- test generation and augmentation

Expanding the space of speculative analysis

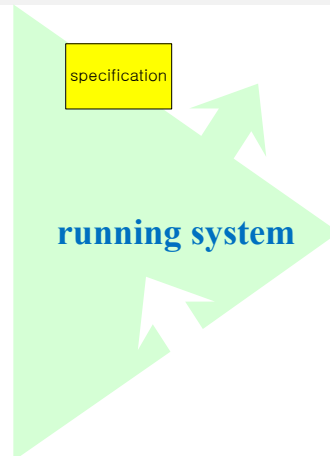
Identify a domain with:

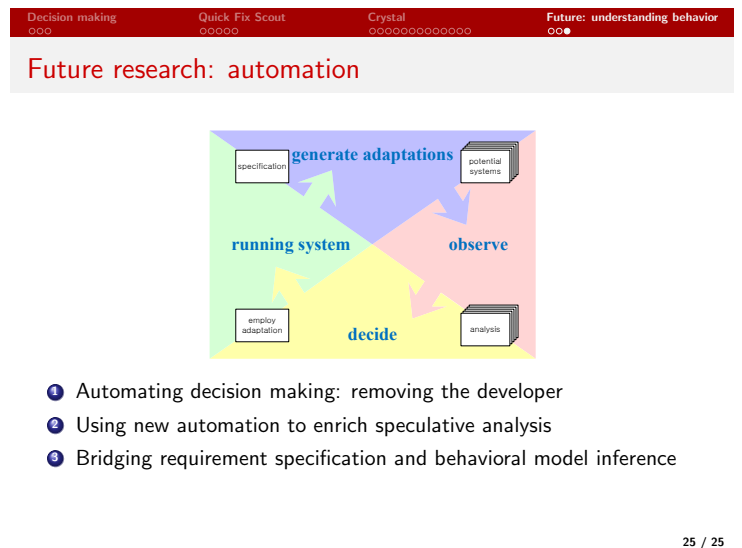
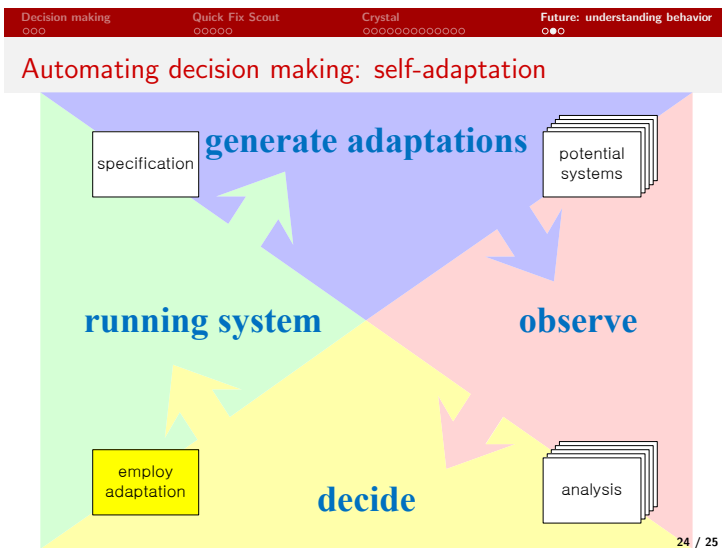
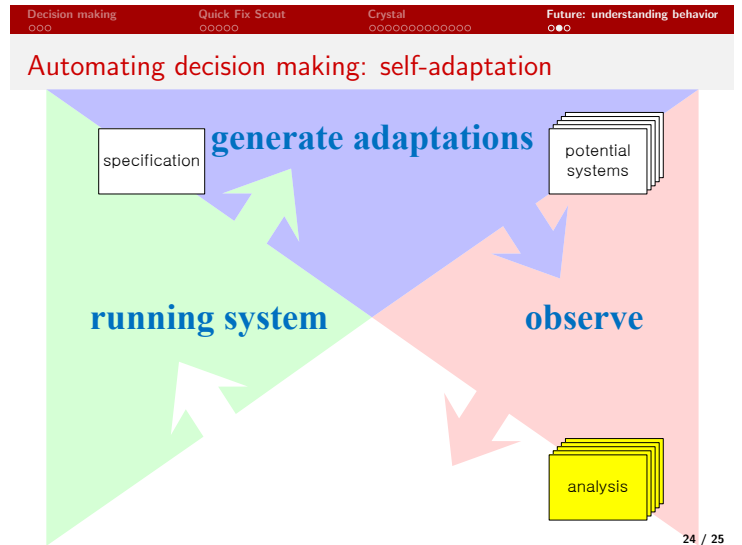
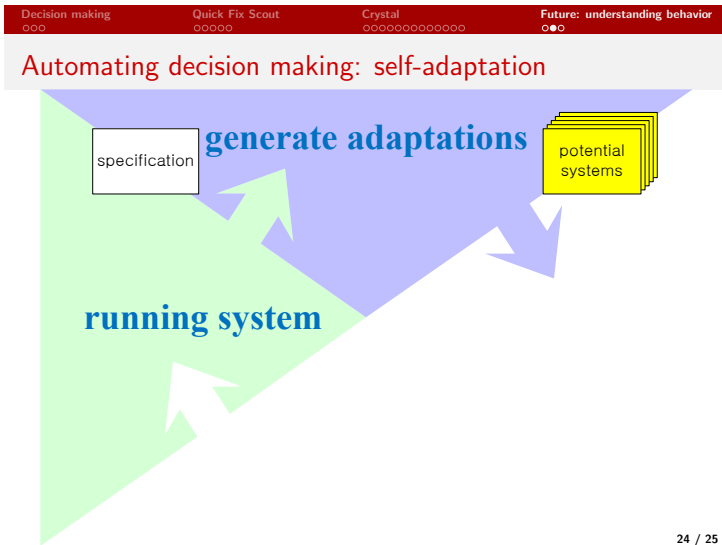
- likely, automatable developer actions
- informative, efficient analyses
- inferable developer intent

Next speculations:

- automated fault removal
- code parallelization
- test generation and augmentation

Automating decision making: self-adaptation





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

Jacob T. Biehl, Mary Czerwinski, Greg Smith, and George G. Robertson. FASTDash: A visual dashboard for fostering awareness in software teams. In *CHI*, pages 1313–1322, San Jose, CA, USA, Apr. 2007. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240823.

Bruce Childers, Jack W. Davidson, and Mary Lou Soffa. Continuous compilation: A new approach to aggressive and adaptive code transformation. In *IPDPS*, 2003.

Prasun Dewan and Rajesh Hegde. Semi-synchronous conflict detection and resolution in asynchronous software development. In *ECSCW*, pages 159–178, Limerick, Ireland, 2007.

Eclipse. The Eclipse foundation. <http://www.eclipse.org>, 2011.

Mário Luís Guimarães and António Rito-Silva. Towards real-time integration. In *CHASE*, pages 56–63, Cape Town, South Africa, May 2010.

Lile Hattori and Michele Lanza. Syde: A tool for collaborative software development. In *ICSE Tool Demo*, pages 235–238, Cape Town, South Africa, May 2010. ISBN 978-1-60558-719-6. doi: 10.1145/1810295.1810339.

Peter Henderson and Mark Weiser. Continuous execution: The VisiProg environment. In *ICSE*, pages 68–74, London, England, UK, Aug. 1985.

R. R. Karinithi and M. Weiser. Incremental re-execution of programs. In *SIIT*, pages 38–44, St. Paul, MN, USA, June 1987. ISBN 0-89791-235-7. doi: 10.1145/29650.29654.

David Saff and Michael D. Ernst. Reducing wasted development time via continuous testing. In *ISSRE*, pages 281–292, Denver, CO, USA, Nov. 2003. ISBN 0-7695-2007-3.

David Saff and Michael D. Ernst. An experimental evaluation of continuous testing during development. In *ISSTA*, pages 76–85, Boston, MA, USA, July 2004. doi: 10.1145/1007512.1007523.

Anita Sarma, Zahra Noroozi, and André van der Hoek. Palantir: Raising awareness among configuration management workspaces. In *ICSE*, pages 444–454, Portland, OR, May 2003. ISBN 0-7695-1877-X.

Steve Streeting. Sourcetre. <http://www.sourcetreapp.com>, 2010.

Jan Wloka, Barbara Ryder, Frank Tip, and Xiaoxia Ren. Safe-commit analysis to facilitate team software development. In *ICSE*, pages 507–517, Vancouver, BC, Canada, May 2009. ISBN 978-1-4244-3453-4. doi: 10.1109/ICSE.2009.5070549.