Welcome DARPA and AFRL Visitors!

**MUSE Meeting in DLC 170**

1:00pm-1:10pm  Welcome
1:10pm-1:45pm  Overview of the **Fixr** Project, Evan Chang
1:45pm-3:00pm  Demo and Discussion: Analysis and Synthesis of App Commits, Shawn Meier/Vaibhav Singh
3:00pm-3:20pm  Break
3:20pm-4:00pm  Demo and Discussion: Harvesting and Storing App Commits, Mazin Hakeem/Sanghee Kim

**Additional Research Meetings in ECCS 1B11**

4:00pm-5:15pm  Graduate Students
Fixr: Mining and Understanding Bug Fixes for App-Framework Protocol Defects

Bor-Yuh Evan Chang  Ken Anderson  Pavol Cerny  Sriram Sankaranarayanan  Tom Yeh

University of Colorado Boulder

MUSE Site Visit
February 25, 2015
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...

caused by an app-created memory leak
Ask framework devs ...
Ask framework devs ...

Android Developers Blog

18 JANUARY 2009

Avoiding memory leaks

Android applications are, at least on the T-Mobile G1, limited to 16 MB of heap. It's both a lot of memory for a phone and yet very little for what some developers want to achieve. Even if you do not plan on using all of this memory, you should use as little as possible to let other applications run without getting them killed. The more applications Android can keep in memory, the faster it will be for the user to switch between his apps. As part of my job, I ran into memory leaks issues in Android applications and they are mostly of the time due to the same mistake: keeping a long-lived reference to a Context.

On Android, a Context is used for many operations but mostly to load and access resources. This is why all the widgets receive a Context parameter in their constructor. In a regular Android application, you usually have two kinds of Context: Activity and Application. It's usually the first one that the developer passes to classes and methods that need a Context:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TextView label = new TextView(this);
    label.setText("Leaks are bad");

    //...
```
Ask framework devs ...

“Do not keep long-lived references to a context-activity”
Ask framework devs ...

“Do not keep long-lived references to a context-activity”

I don’t know how I created a long-lived reference to an Activity!
Ask framework devs ...

“Do not keep long-lived references to a context-activity”

I don’t know how I created a long-lived reference to an Activity!

Often: a misunderstanding of a library causes the library to keep the Activity
Ask framework devs ...

“Do not keep long-lived references to a context-activity”

Bug from violating (implicit) framework protocol rules
Imagining a post-MUSE scenario ...

for

I don’t know how I created a long-lived reference to an Activity!
Elsewhere, following the state of practice for debugging leaks ...
Elsewhere, following the state of practice for debugging leaks ...

1. Run the app
Elsewhere, following the state of practice for debugging leaks ...

1. Run the app
2. Watch the heap usage
Elsewhere, following the state of practice for debugging leaks ...

1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
Elsewhere, following the state of practice for debugging leaks ...

1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
Elsewhere, following the state of practice for debugging leaks ...

1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
4. Commit a bugfix
Elsewhere, following the state of practice for debugging leaks …

1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
4. Commit a bugfix
5. Bugfix is picked up by Fixr
A **Fixr**-enabled IDE responds ...

I don’t know how I created a long-lived reference to an *Activity*!
A Fixr-enabled IDE responds …

I don’t know how I created a long-lived reference to an Activity!
A Fixr-enabled IDE responds ...

It looks like you’ve created a memory leak like and 100,000 others. Would you like to apply?

I don’t know how I created a long-lived reference to an Activity!
A Fixr-enabled IDE responds ...

It looks like you’ve created a memory leak like and 100,000 others. Would you like to apply ?

the bugfix is “transferred”

I don’t know how I created a long-lived reference to an Activity!
Summary: Mine framework specifications with bugfixes

Prior Hypothesis of a Framework Invariant

Observe a Bugfix

Bayesian Update

Posterior Hypothesis
Summary: **Mine** framework specifications with **bugfixes**

Prior Hypothesis of a Framework Invariant

*Observe a Bugfix*

**Bayesian Update**

Posterior Hypothesis

---

The **Fixr** Loop: Create as many observations as possible
Simple motivating example: A well-understood Android bug
Simple motivating example:
A well-understood Android bug

a common misuse of the framework
aView.setTag(..., anObject)
aView.setTag(..., anObject)

if anObject can reach aView
Bug (on Android <4)

Framework Invariant

class View {
    static WeakHashMap<View, SparseArray<Object>> sTags;
    Object mTag;
}

aView.setTag(..., anObject)

if anObject can reach aView
class View {
    static WeakHashMap<View, SparseArray<Object>> sTags;
    Object mTag;
}

because of an unspecified class invariant: sTags' values (Object) must not reach their keys (View)

Bug (on Android <4) because of an unspecified class invariant: sTags' values (Object) must not reach their keys (View)
Bug (on Android <4)

aView.setTag(..., anObject)

if anObject can reach aView

class View {
    static WeakHashMap<View, SparseArray<Object>> sTags;
    Object mTag;
}

Framework Invariant

because of an unspecified class invariant: sTags' values (:Object) must not reach their keys (:View)

A Fix
```java
class View {
    static WeakHashMap<View, SparseArray<Object>> sTags;
    Object mTag;
}

Bug (on Android <4)
because of an unspecified class invariant: sTags' values (:Object) must not reach their keys (:View)

A Fix
aView.setTag(..., anObject)
aView.setTag(..., anObject...
uses mTag instead
```
**Bug** (on Android <4)

```
aView.setTag(..., anObject)
```

**if anObject can reach aView**

```
class View {
    static WeakHashMap<View, SparseArray<Object>> sTags;
    Object mTag;
}
```

**Framework Invariant**

because of an unspecified class invariant: `sTags`' values (:Object) must not reach their keys (:View)

**Goal:** Produce this repair specification: bug pre, framework invariant, fix suggestion
Challenges

Given a bugfix commit, how do we summarize and generalize the fix (to be able “transfer”)?

How do we find bugfix commits?
Challenges

Given a bugfix commit, how do we **summarize** and **generalize** the fix (to be able “transfer”)?

A specification of the View.setTag repair

How do we find **bugfix commits**?
Challenges

Given a bugfix commit, how do we **summarize** and **generalize** the fix (to be able “transfer”)?

- a specification of the View.setTag repair

How do we find **bugfix commits**?

- an instance of a View.setTag fix
Fixr
Components and Workflows
Workflow 1a: Harvesting bugfix commits
Workflow 1a: Harvesting bugfix commits
Workflow 1a: Harvesting bugfix commits

FixrDB

Harvestr: Social Validation and Mining of Fixes
Workflow 1a: Harvesting bugfix commits
Workflow 1a: Harvesting bugfix commits

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Harvestr: Social Validation and Mining of Fixes

Github
Workflow 1a: Harvesting bugfix commits

Deltar: Inferring Semantic Deltas and Repair Specifications

E.g., a commit removing a call to setTag

FixrDB

Harvestr: Social Validation and Mining of Fixes

Github
Workflow 1a: Harvesting bugfix commits

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Harvestr: Social Validation and Mining of Fixes

Github

E.g., a commit removing a call to setTag
Workflow 1a: Harvesting bugfix commits

Deltar: Inferring Semantic Deltas and Repair Specifications

E.g., a specific bug pre for calling setTag

E.g., a commit removing a call to setTag

Harvestr: Social Validation and Mining of Fixes

FixrDB

Github

commit

repair specification
Workflow 1a: Harvesting bugfix commits

Component: **Deltar** maps bugfixes to candidate repair specifications (including bug pre)
Workflow 1a: Harvesting bugfix commits

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

E.g., a specific bug pre for calling setTag
repair specification

E.g., a commit removing a call to setTag
fix

Component: **Deltar maps bugfixes to candidate repair specifications (including bug pre)**
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

repair specification

FixrDB
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

FixrDB

repair specification
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

FixrDB

Triggers query to FixrDB for buggy apps (satisfying bug pre)
Workflow 1b: Aggregating repairs

**Deltar:** Inferring Semantic Deltas and Repair Specifications

**Prepair:** Deriving Probabilistic Repair Specifications

*Triggers query to FixrDB for buggy apps (satisfying bug pre)*
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

FixrDB

- Triggers query to FixrDB for buggy apps (satisfying bug pre)
- E.g., repair conditions for setTag
- probabilistic repair specification

repair specification
Workflow 1b: Aggregating repairs

Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

FixrDB

Component: Prepair reduces candidate repair specifications to generalized probabilistic repair specifications
Workflow 1c: Synthesizing patches
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

probabilistic repair specification
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

Probabilistic repair specification

Patchr: Detecting Potential Bugs and Synthesizing Patches
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

Patchr: Detecting Potential Bugs and Synthesizing Patches

Triggers query to FixrDB for buggy apps (satisfying bug pre)

Probabilistic repair specification
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

Probabilistic repair specification

Patchr: Detecting Potential Bugs and Synthesizing Patches

Triggers query to FixrDB for buggy apps (satisfying bug pre)
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

probabilistic repair specification

Patchr: Detecting Potential Bugs and Synthesizing Patches

Triggers query to FixrDB for buggy apps (satisfying bug pre)

Github

patch
Workflow 1c: Synthesizing patches

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

Probabilistic repair specification

Patchr: Detecting Potential Bugs and Synthesizing Patches

Triggers query to FixrDB for buggy apps (satisfying bug pre)

E.g., replace call to setTag with alternative

Github
Workflow 1c: Synthesizing patches

Component: **Patchr** maps buggy apps to patched apps

**Prepair:** Deriving Probabilistic Repair Specifications

**Patchr:** Detecting Potential Bugs and Synthesizing Patches

E.g., replace call to `setTag` with alternative

Triggers query to FixrDB for buggy apps (satisfying bug pre)

FixrDB

probabilistic repair specification
Workflow 1c: Synthesizing patches

Component: Patchr

maps buggy apps to patched apps

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

probabilistic repair specification

Triggers query to FixrDB for buggy apps (satisfying bug pre)

Patchr: Detecting Potential Bugs and Synthesizing Patches

E.g., replace call to setTag with alternative

patch

Github

Vaibhav Singh
Workflow 1c: Synthesizing patches

Component: **Patchr** maps buggy apps to patched apps

**Prepair**: Deriving Probabilistic Repair Specifications

FixrDB

Mazin Hakeem

probabilistic repair specification

Triggers query to FixrDB for buggy apps (satisfying bug pre)

Patchr: Detecting Potential Bugs and Synthesizing Patches

E.g., replace call to setTag with alternative

Github

Vaibhav Singh

Component: **Patchr** maps buggy apps to patched apps
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

Github

commit
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

Github

FixrDB

commit
commit
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Harvestr: Social Validation and Mining of Fixes

Github
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

- **Harvestr**: Social Validation and Mining of Fixes
- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **FixrDB**: Inferring Semantic Facts
- **Github**: Commit
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

- **Harvestr**: Social Validation and Mining of Fixes
- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **FixrDB**: Storing semantic facts
- **Patchr**: Detecting Potential Bugs and Synthesizing Patches

Diagram:

1. Github → Harvestr
2. Harvestr → FixrDB
3. FixrDB → Deltar
4. Deltar → Patchr

Flow of information:
- commit from Github to Harvestr
- semantic facts from FixrDB to Deltar
- commit from Deltar to Patchr
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Patchr: Detecting Potential Bugs and Synthesizing Patches

Github

commit

semantic facts

patch
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Patchr: Detecting Potential Bugs and Synthesizing Patches

Github
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

**Harvestr:** Social Validation and Mining of Fixes

**Deltar:** Inferring Semantic Deltas and Repair Specifications

**FixrDB**

**Patchr:** Detecting Potential Bugs and Synthesizing Patches

E.g., pull request accepted

interaction

commit

semantic facts

commit

patch
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

Deltar: Inferring Semantic Deltas and Repair Specifications

FixrDB

Prepair: Deriving Probabilistic Repair Specifications

Patchr: Detecting Potential Bugs and Synthesizing Patches

E.g., pull request accepted
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

- **Harvestr**: Social Validation and Mining of Fixes
- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **Prepair**: Deriving Probabilistic Repair Specifications
- **Patchr**: Detecting Potential Bugs and Synthesizing Patches

- Github

- FixrDB

- E.g., pull request accepted

- Interaction

- Commit

- Patch

- Updated probabilities
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Component: Harvestr maps external events to semantic facts, updated probabilities, and patches

- **Harvestr**: Social Validation and Mining of Fixes
- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **Prepair**: Deriving Probabilistic Repair Specifications
- **Patchr**: Detecting Potential Bugs and Synthesizing Patches

E.g., pull request accepted

Interaction: Harvestr maps external events to semantic facts, updated probabilities, and patches

Commit: Github
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Component: **Harvestr** maps external events to semantic facts, updated probabilities, and patches

- **Harvestr**: Social Validation and Mining of Fixes
- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **Prepair**: Deriving Probabilistic Repair Specifications
- **Patchr**: Detecting Potential Bugs and Synthesizing Patches

E.g., pull request accepted

**Github**

Sanghee Kim
Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

Harvestr: Social Validation and Mining of Fixes

Patchr: Detecting Potential Bugs and Synthesizing Patches

FixrDB

social program analysis

semantic
statistical-semantic
syntactic
social

software engineering
for big data

interaction

commit

github

Tom Yeh

Bor-Yuh Evan Chang

Sriram Sankaranarayanan

Pavol Cerny

Tom Yeh

Ken Anderson

Semantic facts

repair specification

probabilistic repair specification

program synthesis

user-centered big data analytics