Fixr: Mining and Understanding Bug Fixes for App-Framework Protocol Defects (TA2)

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

Shawn Meier Mazin Hakeem Krishna Chaitanya Sripada Sanghee Kim

University of Colorado Boulder

MUSE PI Meeting
July 22, 2015

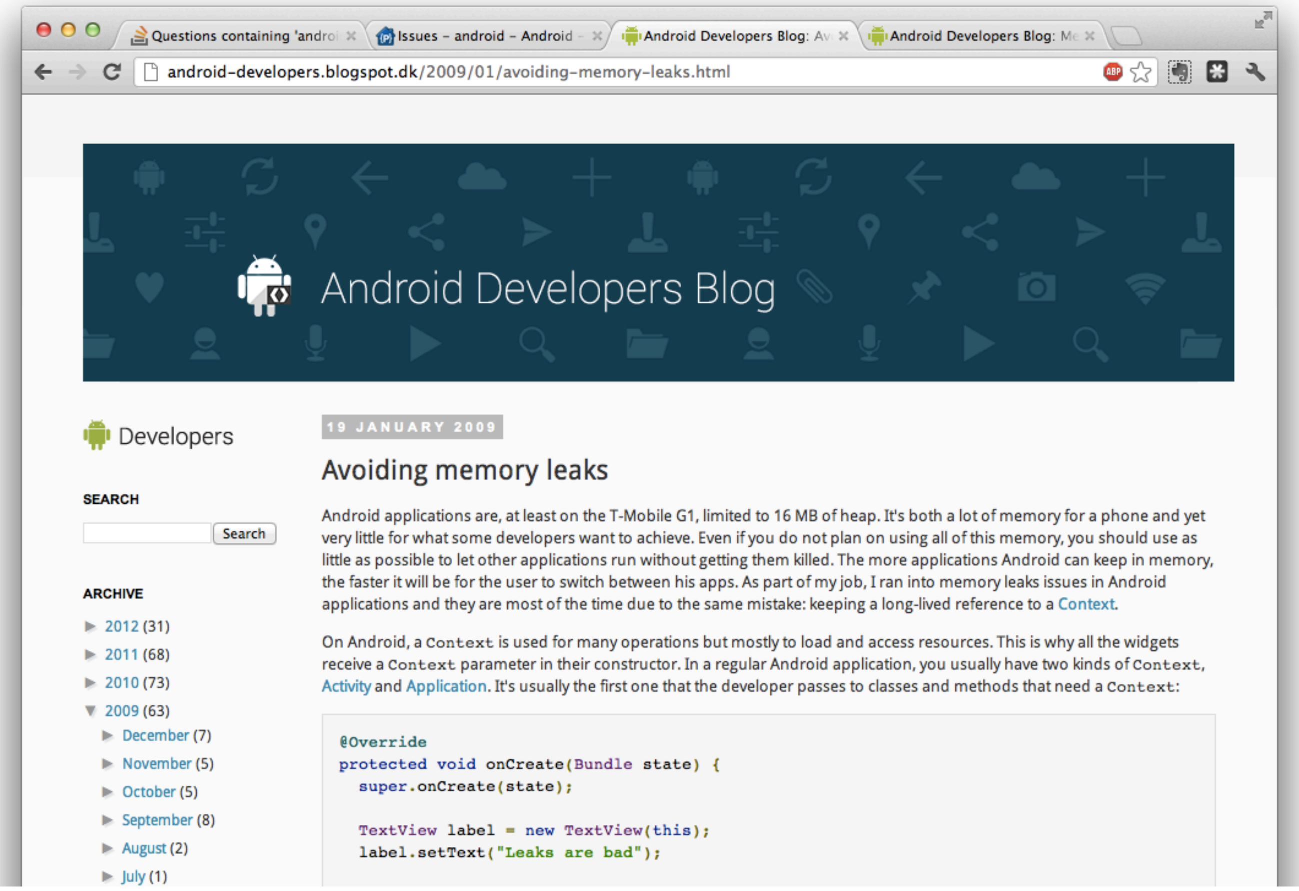
PLV

Imagining a post-MUSE scenario …

for

App developer asks framework devs …

App developer asks framework devs …


Questions containing 'andro
Issues - android - Android
Android Developers Blog: Av
Android Developers Blog: Me
android-developers.blogspot.dk/2009/01/avoiding-memory-leaks.html
Android Developers Blog
Developers
SEARCH
Search
ARCHIVE
2012 (31)
2011 (68)
2010 (73)
2009 (63)
December (7)
November (5)
October (5)
September (8)
August (2)
July (1)

19 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 most of the time due to the same mistake: keeping a long-lived reference to a Context.

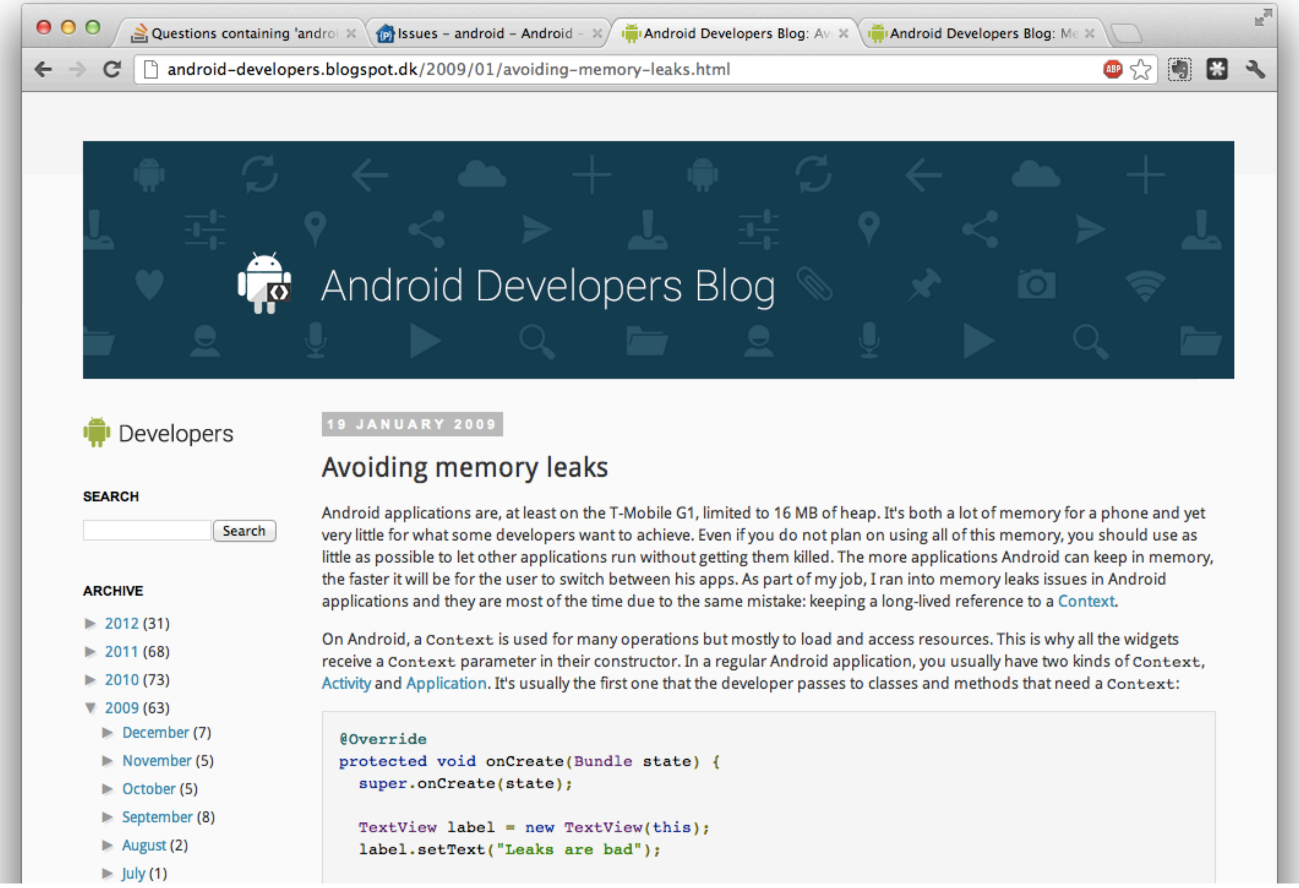
On Android, a `Context` is used for many operations but mostly to load and access resources. This is why all the widgets receive a `Context` parameter in their constructor. In a regular Android application, you usually have two kinds of `Context`, Activity and Application. It's usually the first one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

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

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

App developer asks framework devs …

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



Questions containing 'andro
Issues - android - Android
Android Developers Blog: Av
Android Developers Blog: Me
android-developers.blogspot.dk/2009/01/avoiding-memory-leaks.html
Android Developers Blog
Developers
SEARCH
Search
ARCHIVE
2012 (31)
2011 (68)
2010 (73)
2009 (63)
December (7)
November (5)
October (5)
September (8)
August (2)
July (1)

19 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 most of the time due to the same mistake: keeping a long-lived reference to a Context.

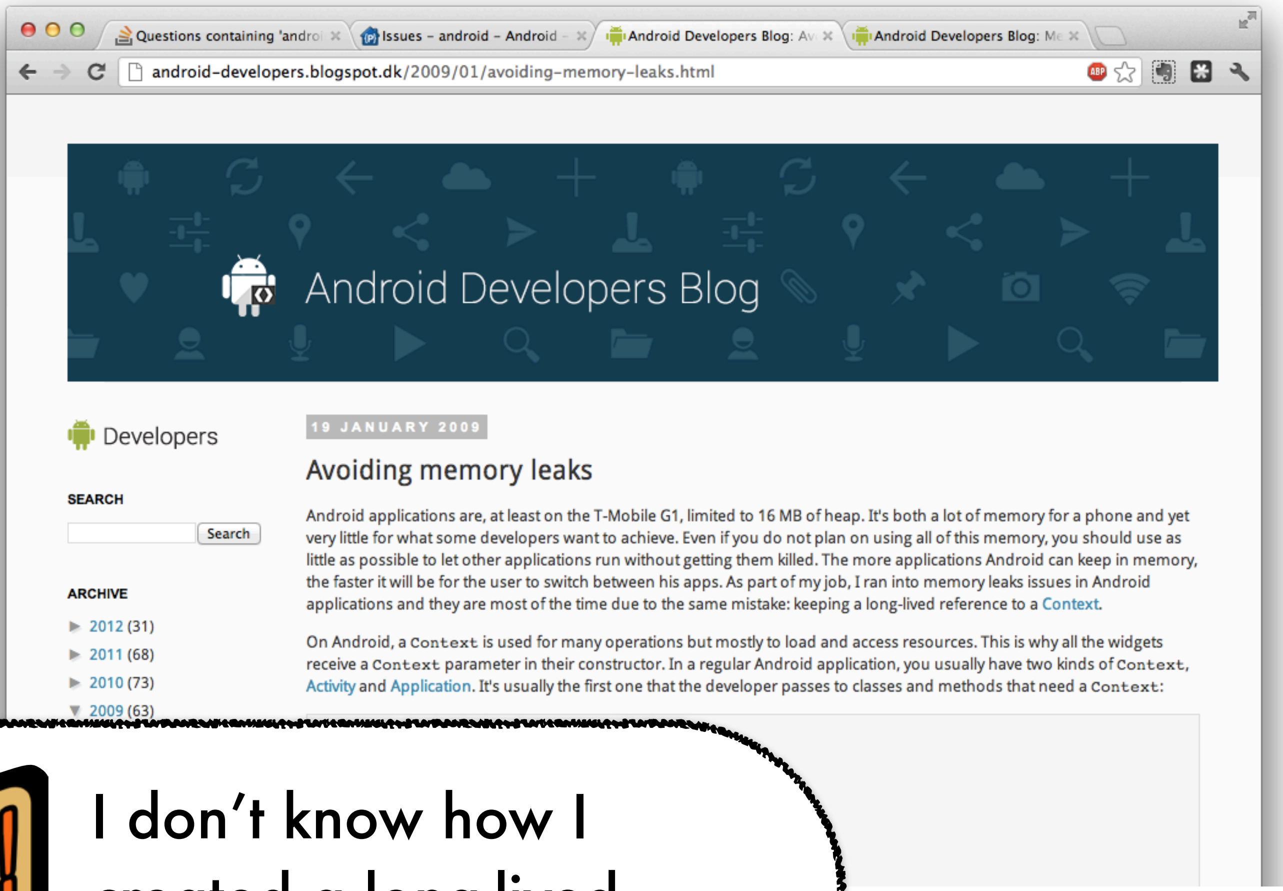
On Android, a `Context` is used for many operations but mostly to load and access resources. This is why all the widgets receive a `Context` parameter in their constructor. In a regular Android application, you usually have two kinds of `Context`, Activity and Application. It's usually the first one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

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

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

App developer asks framework devs ...

"Do not keep long-lived references to a context-activity"



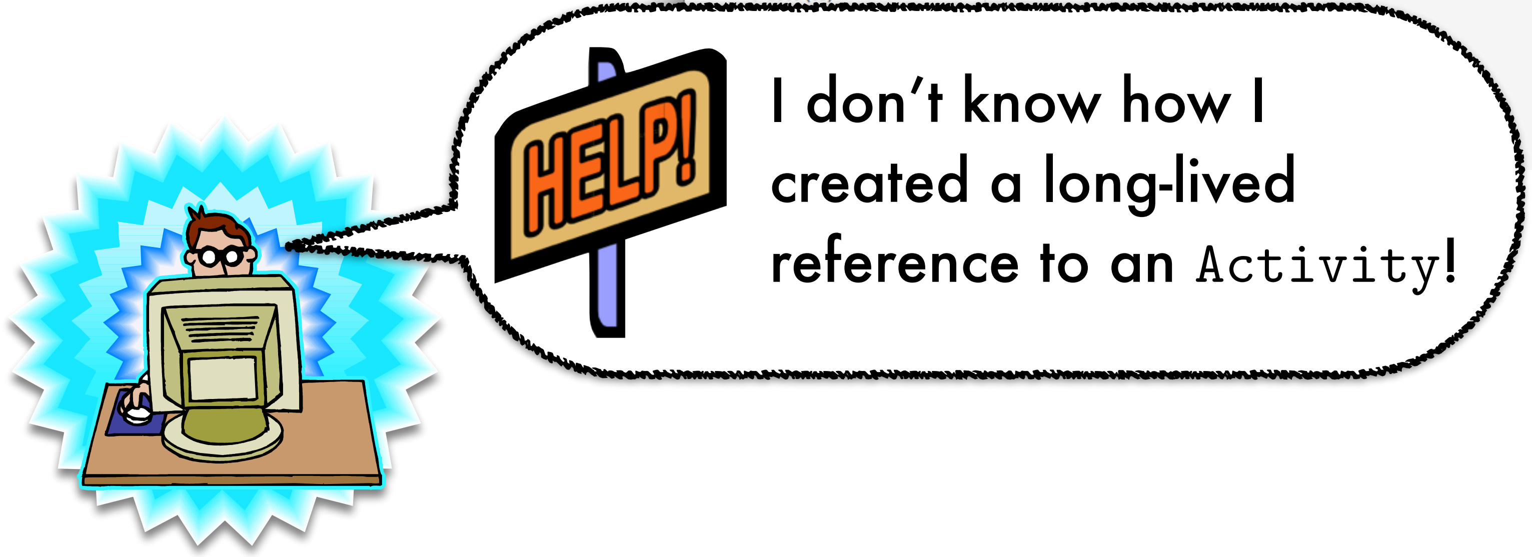
Questions containing 'andro
Issues – android – Android
Android Developers Blog: Av
Android Developers Blog: Me
android-developers.blogspot.dk/2009/01/avoiding-memory-leaks.html
Android Developers Blog
Developers
SEARCH
Search
ARCHIVE
▶ 2012 (31)
▶ 2011 (68)
▶ 2010 (73)
▼ 2009 (63)

19 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 most of the time due to the same mistake: keeping a long-lived reference to a Context.

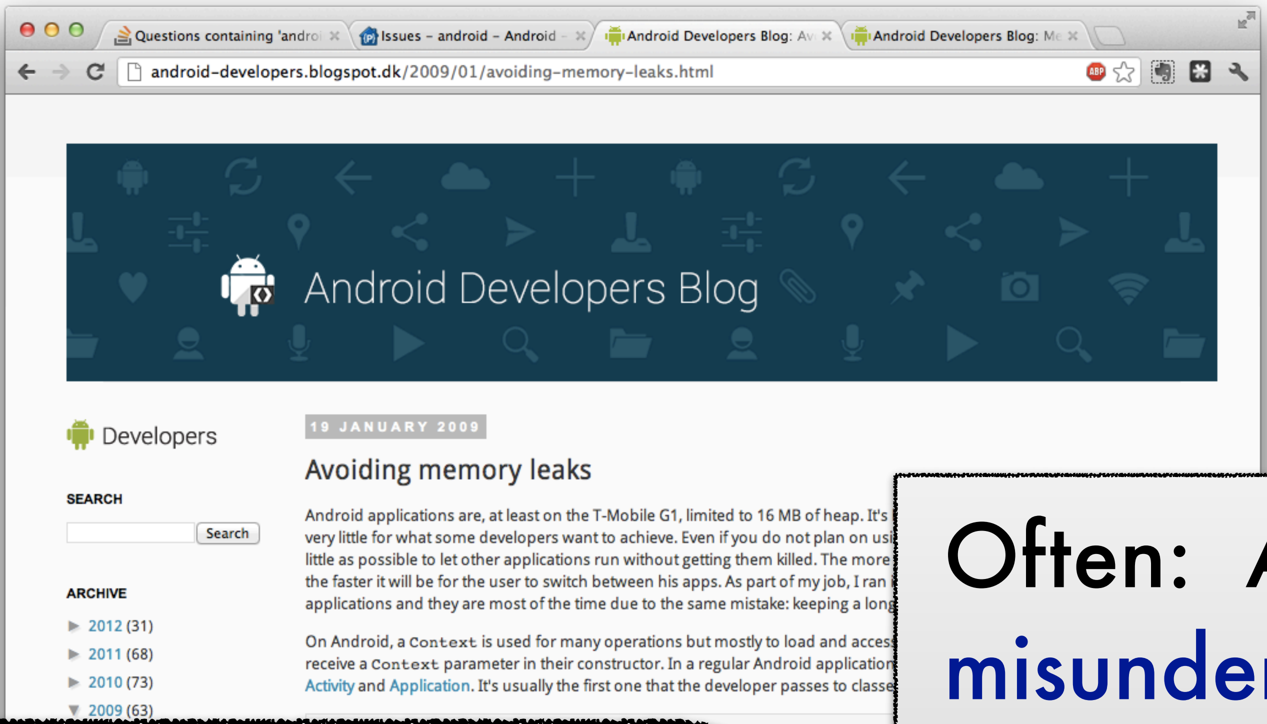
On Android, a `Context` is used for many operations but mostly to load and access resources. This is why all the widgets receive a `Context` parameter in their constructor. In a regular Android application, you usually have two kinds of `Context`, Activity and Application. It's usually the first one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



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

App developer asks framework devs ...

"Do not keep long-lived references to a context-activity"



android-developers.blogspot.dk/2009/01/avoiding-memory-leaks.html

Android Developers Blog

Developers

SEARCH

Search

ARCHIVE

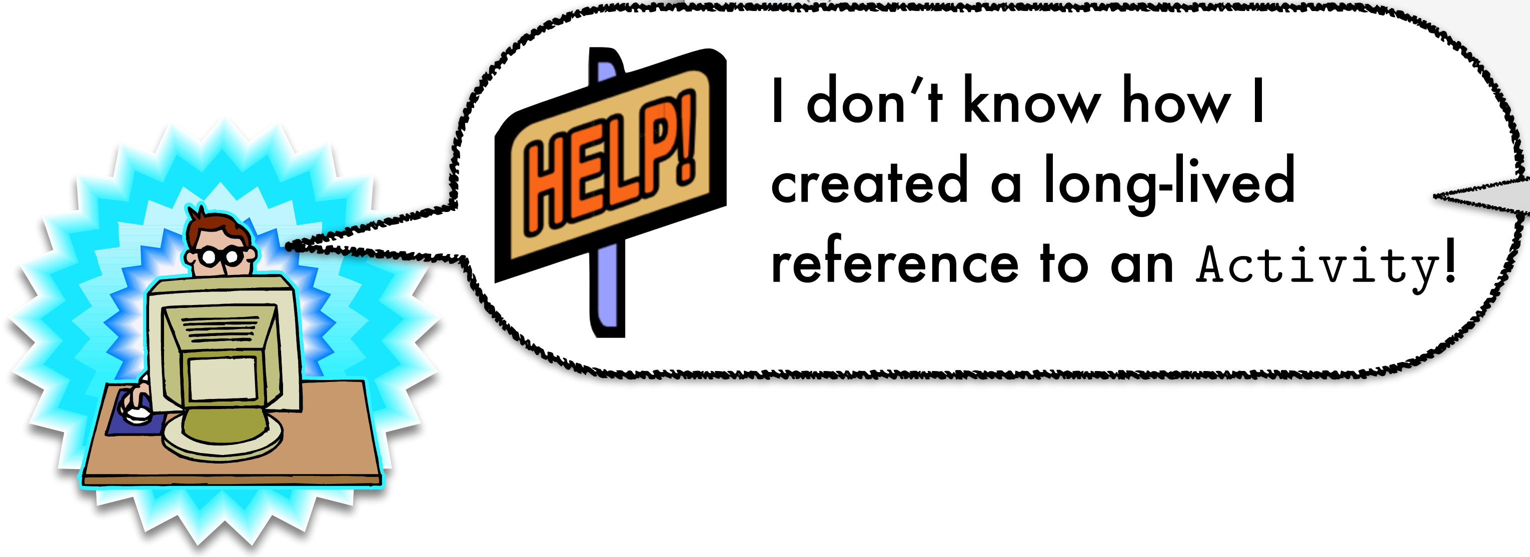
▶ 2012 (31)
▶ 2011 (68)
▶ 2010 (73)
▼ 2009 (63)

19 JANUARY 2009

Avoiding memory leaks

Android applications are, at least on the T-Mobile G1, limited to 16 MB of heap. It's
very little for what some developers want to achieve. Even if you do not plan on usi
little as possible to let other applications run without getting them killed. The more
the faster it will be for the user to switch between his apps. As part of my job, I ran i
applications and they are most of the time due to the same mistake: keeping a long

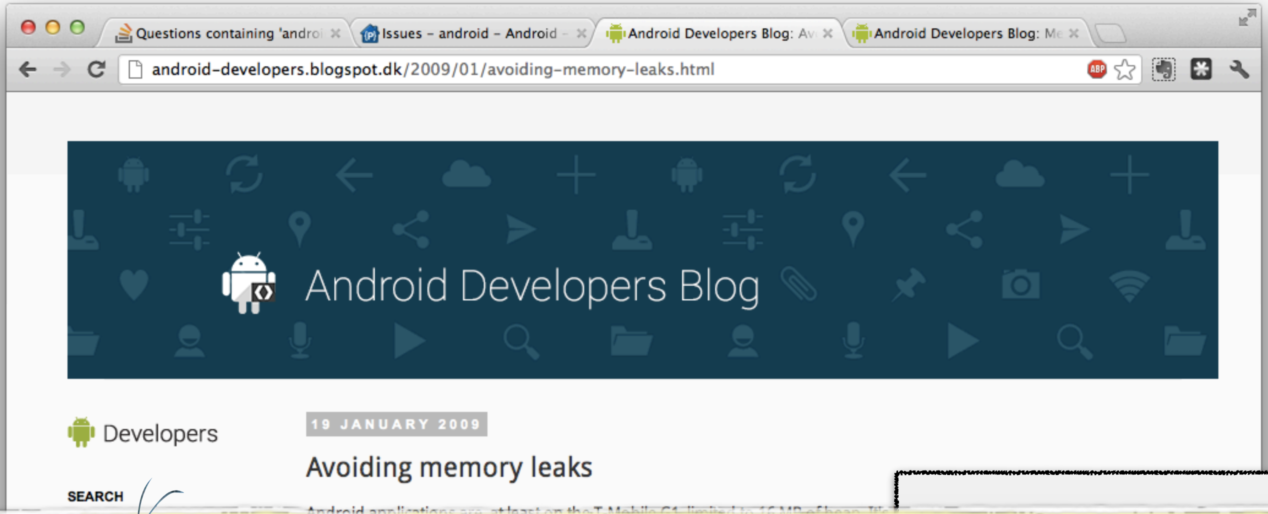
On Android, a `Context` is used for many operations but mostly to load and access
receive a `Context` parameter in their constructor. In a regular Android application
Activity and Application. It's usually the first one that the developer passes to classe


HELP!
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`

App developer asks framework devs …

"Do not keep long-lived references to a context-activity"



Questions containing 'andro Issues – android – Android Android Developers Blog: Av Android Developers Blog: Me

android-developers.blogspot.dk/2009/01/avoiding-memory-leaks.html

Android Developers Blog

Developers

SEARCH

19 JANUARY 2009

Avoiding memory leaks

Bug from violating
(implicit) framework protocol rules

Somewhere ...

Somewhere ...

Somewhere ...

Questions containing 'andro
Issues - android - Android
Android Developers Blog: Av
Android Developers Blog: Me
android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html

Android Developers Blog

Developers

SEARCH

Search

ARCHIVE

- 2012 (31)
- 2011 (68)
  - December (7)
  - November (7)
  - October (5)
  - September (5)
  - August (3)
  - July (7)
  - June (3)
  - May (5)

24 MARCH 2011

Memory Analysis for Android Applications

[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

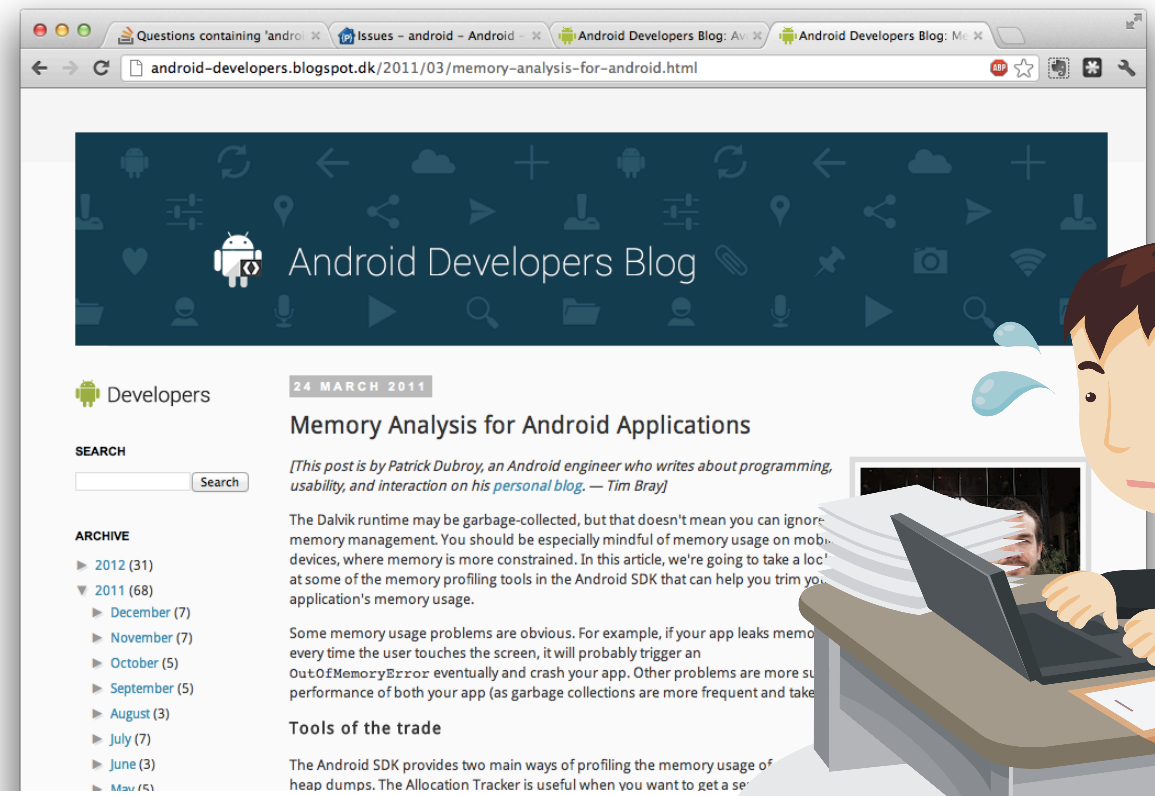
The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignor
memory management. You should be especially mindful of memory usage on mob
devices, where memory is more constrained. In this article, we're going to take a loc
at some of the memory profiling tools in the Android SDK that can help you trim yo
application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo
every time the user touches the screen, it will probably trigger an
`OutOfMemoryError` eventually and crash your app. Other problems are more su
performance of both your app (as garbage collections are more frequent and take

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of
heap dumps. The Allocation Tracker is useful when you want to get a se

Somewhere ...


Questions containing 'andro
Issues - android - Android
Android Developers Blog: An
Android Developers Blog: Me
android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html

Android Developers Blog

Developers

SEARCH

Search

ARCHIVE

- 2012 (31)
- 2011 (68)
  - December (7)
  - November (7)
  - October (5)
  - September (5)
  - August (3)
  - July (7)
  - June (3)
  - May (5)

24 MARCH 2011

Memory Analysis for Android Applications

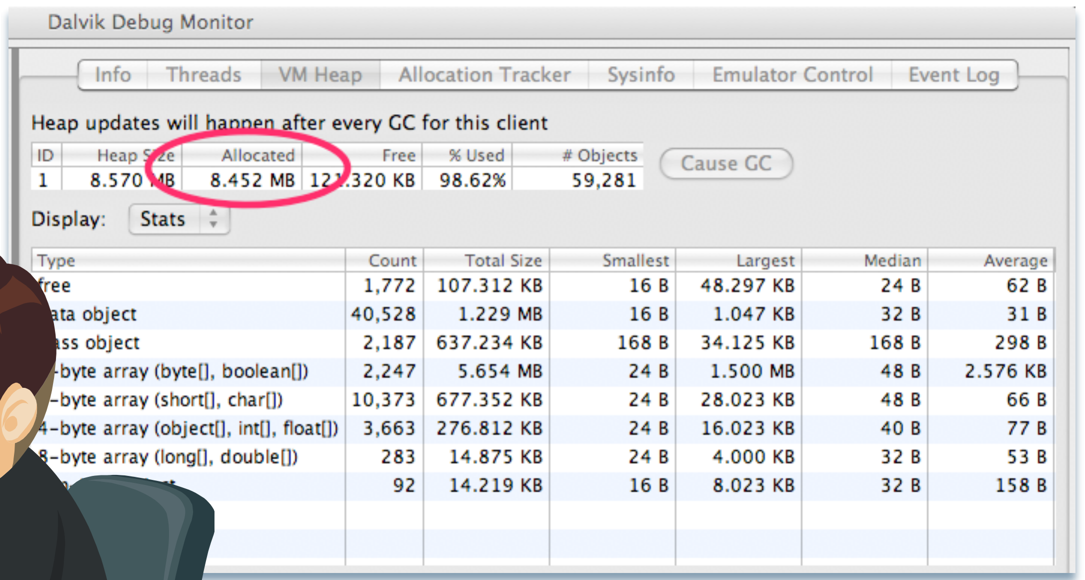
[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignore memory management. You should be especially mindful of memory usage on mobi devices, where memory is more constrained. In this article, we're going to take a loc at some of the memory profiling tools in the Android SDK that can help you trim yo application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo every time the user touches the screen, it will probably trigger an `OutOfMemoryError` eventually and crash your app. Other problems are more su performance of both your app (as garbage collections are more frequent and take

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of heap dumps. The Allocation Tracker is useful when you want to get a se



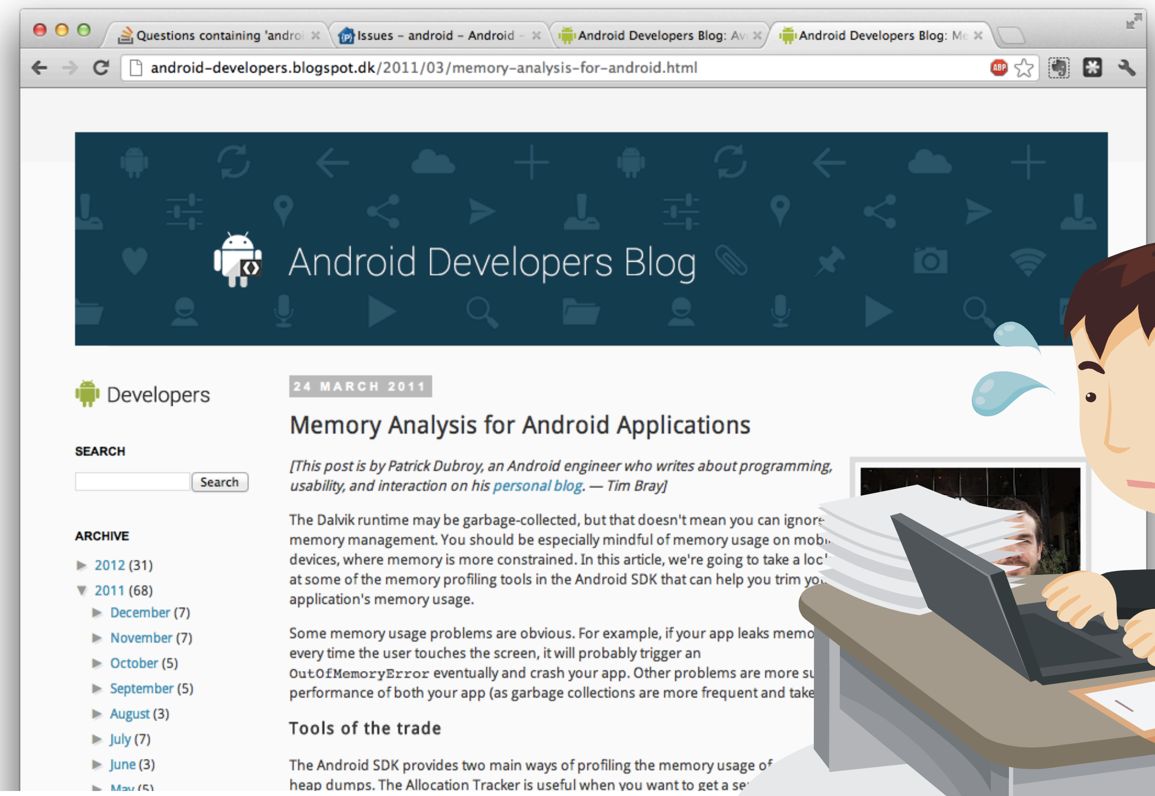
Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	1,772	107.312 KB	16 B	48.297 KB	24 B	62 B
ata object	40,528	1.229 MB	16 B	1.047 KB	32 B	31 B
ass object	2,187	637.234 KB	168 B	34.125 KB	168 B	298 B
-byte array (byte[], boolean[])	2,247	5.654 MB	24 B	1.500 MB	48 B	2.576 KB
-byte array (short[], char[])	10,373	677.352 KB	24 B	28.023 KB	48 B	66 B
4-byte array (object[], int[], float[])	3,663	276.812 KB	24 B	16.023 KB	40 B	77 B
8-byte array (long[], double[])	283	14.875 KB	24 B	4.000 KB	32 B	53 B
[illegible]	92	14.219 KB	16 B	8.023 KB	32 B	158 B

Somewhere ...


Questions containing 'andro
Issues - android - Android
Android Developers Blog: An
Android Developers Blog: M

android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html

Android Developers Blog

Developers

SEARCH

Search

ARCHIVE

- 2012 (31)
- 2011 (68)
  - December (7)
  - November (7)
  - October (5)
  - September (5)
  - August (3)
  - July (7)
  - June (3)
  - May (5)

24 MARCH 2011

Memory Analysis for Android Applications

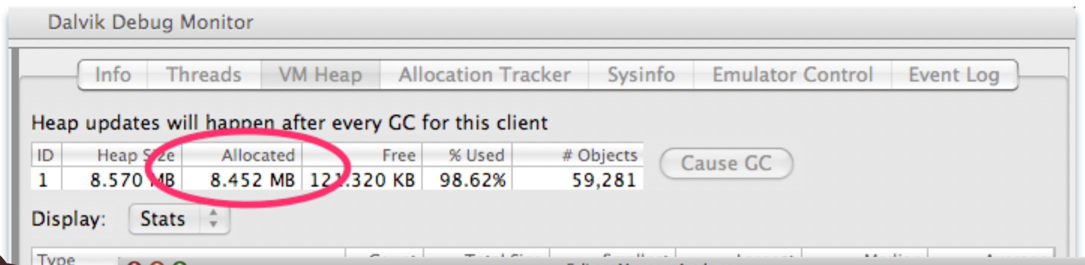
[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignore memory management. You should be especially mindful of memory usage on mobi devices, where memory is more constrained. In this article, we're going to take a loo at some of the memory profiling tools in the Android SDK that can help you trim yo application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo every time the user touches the screen, it will probably trigger an `OutOfMemoryError` eventually and crash your app. Other problems are more su performance of both your app (as garbage collections are more frequent and take

Tools of the trade

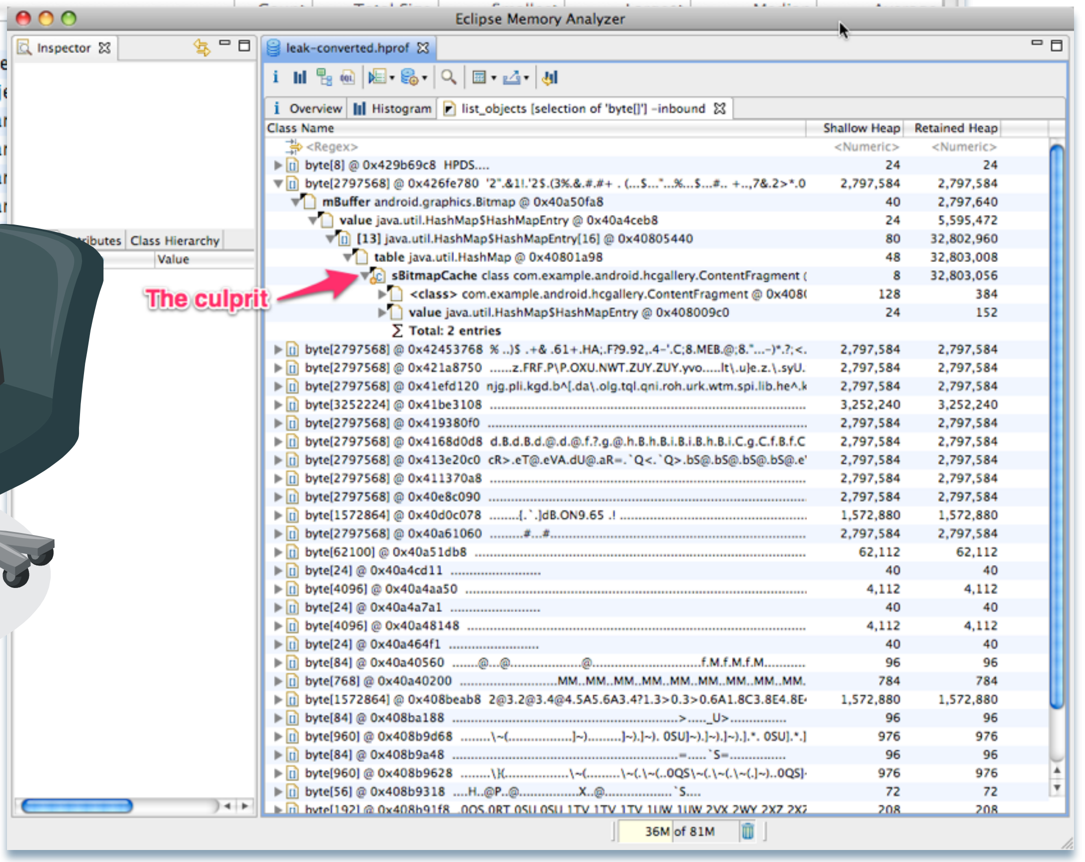
The Android SDK provides two main ways of profiling the memory usage of heap dumps. The Allocation Tracker is useful when you want to get a se



Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

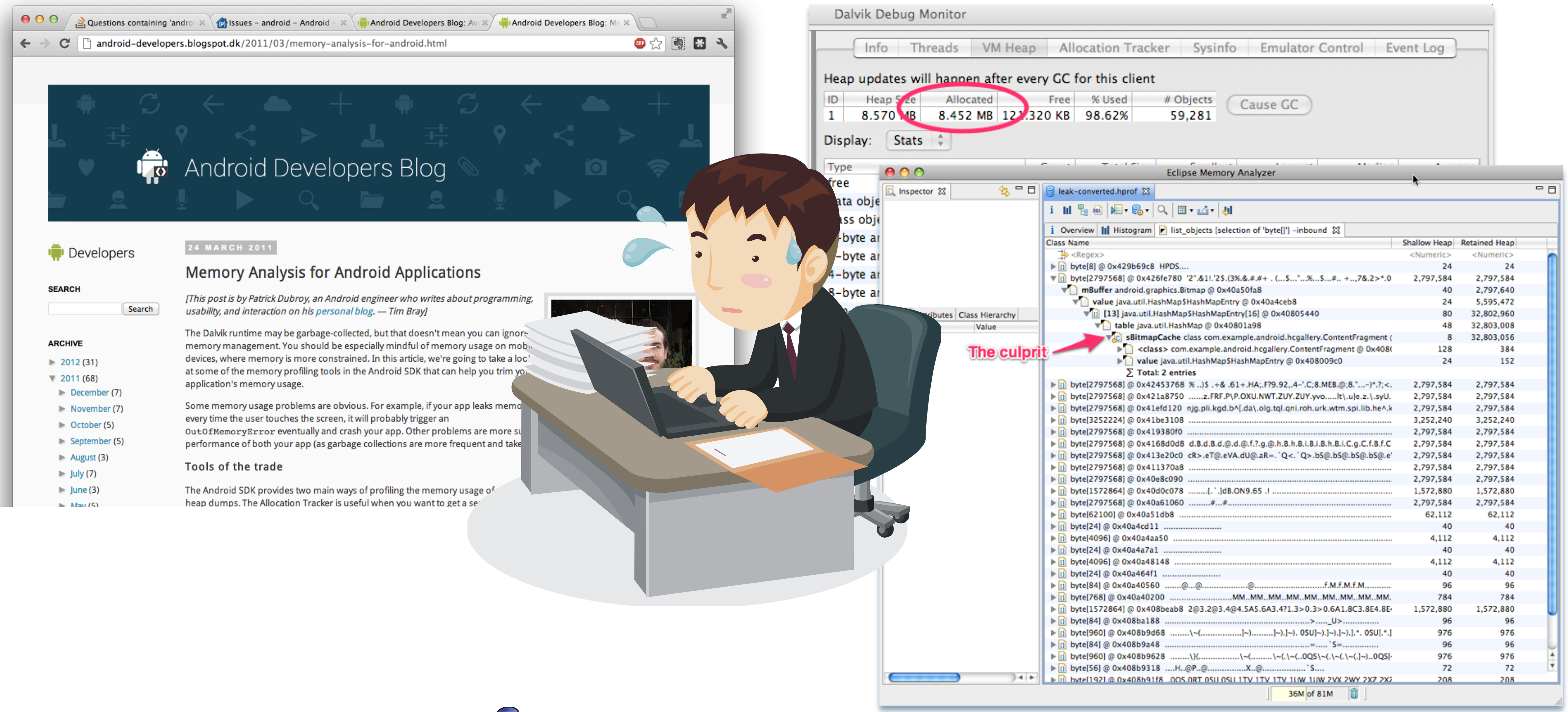
Display: Stats



```
Class Name                                                                                  Shallow Heap  Retained Heap
<Regex>                                                                                       <Numeric>      <Numeric>
▶ byte[8] @ 0x429b69c8  HPDS....                                                                     24             24
▼ byte[2797568] @ 0x426fe780  '2".&1!.'2$.(3%.&.#.#+ . (...$..."...%...$...#.. +..,7&.2>*.0  2,797,584      2,797,584
  ▼ mBuffer android.graphics.Bitmap @ 0x40a50fa8                                                     40      2,797,640
    ▼ value java.util.HashMap$HashMapEntry @ 0x40a4ceb8                                              24      5,595,472
      ▼ [13] java.util.HashMap$HashMapEntry[16] @ 0x40805440                                         80     32,802,960
        ▼ table java.util.HashMap @ 0x40801a98                                                       48     32,803,008
          ▼ sBitmapCache class com.example.android.hcgallery.ContentFragment (                        8     32,803,056
            ▶ <class> com.example.android.hcgallery.ContentFragment @ 0x408(                        128            384
            ▶ value java.util.HashMap$HashMapEntry @ 0x408009c0                                      24            152
            Σ Total: 2 entries
▶ byte[2797568] @ 0x42453768  % ..)$ .+& .61+.HA;.F?9.92,.4-'.C;8.MEB.@;8."...-)*.?;<.        2,797,584      2,797,584
▶ byte[2797568] @ 0x421a8750  ......z.FRF.P\P.OXU.NWT.ZUY.ZUY.yvo.....lt\.u]e.z.\.syU.        2,797,584      2,797,584
▶ byte[2797568] @ 0x41efd120  njg.pli.kgd.b^[.da\.olg.tql.qni.roh.urk.wtm.spi.lib.he^.k         2,797,584      2,797,584
▶ byte[3252224] @ 0x41be3108  ..........................................................        3,252,240      3,252,240
▶ byte[2797568] @ 0x419380f0  ..........................................................        2,797,584      2,797,584
▶ byte[2797568] @ 0x4168d0d8  d.B.d.B.d.@.d.@.f.?.g.@.h.B.h.B.i.B.i.B.h.B.i.C.g.C.f.B.f.C       2,797,584      2,797,584
▶ byte[2797568] @ 0x413e20c0  cR>.eT@.eVA.dU@.aR=.`Q<.`Q>.bS@.bS@.bS@.bS@.e'                   2,797,584      2,797,584
▶ byte[2797568] @ 0x411370a8  ..........................................................        2,797,584      2,797,584
▶ byte[2797568] @ 0x40e8c090  ..........................................................        2,797,584      2,797,584
▶ byte[1572864] @ 0x40d0c078  ........[.`.]dB.ON9.65 .! ....................................    1,572,880      1,572,880
▶ byte[2797568] @ 0x40a61060  .........#...#............................................       2,797,584      2,797,584
▶ byte[62100] @ 0x40a51db8  ............................................................          62,112         62,112
▶ byte[24] @ 0x40a4cd11  ........................                                                    40             40
▶ byte[4096] @ 0x40a4aa50  .............................................................          4,112          4,112
▶ byte[24] @ 0x40a4a7a1  ........................                                                    40             40
▶ byte[4096] @ 0x40a48148  .............................................................          4,112          4,112
▶ byte[24] @ 0x40a464f1  ........................                                                    40             40
▶ byte[84] @ 0x40a40560  .......@...@..................@...........................f.M.f.M.f.M...........  96             96
▶ byte[768] @ 0x40a40200  ........................MM..MM..MM..MM..MM..MM..MM..MM..MM.         784            784
▶ byte[1572864] @ 0x408beab8  2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3.8E4.8E·            1,572,880      1,572,880
▶ byte[84] @ 0x408ba188  ..........................................................>......_U>...............  96             96
▶ byte[960] @ 0x408b9d68  ........\~(.................]~)..........]~).]~). 0SU]~).]~).]~).].*. 0SU].*.]  976            976
▶ byte[84] @ 0x408b9a48  ..........................................................=...... `S=...............  96             96
▶ byte[960] @ 0x408b9628  ........\}(.................\~(..........\~(.\~(..0QS\~(.\~(.\~(.]~)..0QS]·     976            976
▶ byte[56] @ 0x408b9318  ....H..@P..@...............X..@................. `S....                      72             72
▶ byte[192] @ 0x408b91f8  .0QS.0RT.0SU.0SU.1TV.1TV.1TV.1UW.1UW.2VX.2WY.2XZ.2XZ                    208            208
```

36M of 81M

Somewhere ...



android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html

Android Developers Blog

Developers

SEARCH

Search

ARCHIVE

- 2012 (31)
- 2011 (68)
  - December (7)
  - November (7)
  - October (5)
  - September (5)
  - August (3)
  - July (7)
  - June (3)

24 MARCH 2011

Memory Analysis for Android Applications

[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignore memory management. You should be especially mindful of memory usage on mobi devices, where memory is more constrained. In this article, we're going to take a loo at some of the memory profiling tools in the Android SDK that can help you trim yo application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo every time the user touches the screen, it will probably trigger an `OutOfMemoryError` eventually and crash your app. Other problems are more su performance of both your app (as garbage collections are more frequent and take

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of heap dumps. The Allocation Tracker is useful when you want to get a se

Dalvik Debug Monitor

Info | Threads | VM Heap | Allocation Tracker | Sysinfo | Emulator Control | Event Log

Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

Cause GC

Display: Stats

Eclipse Memory Analyzer

leak-converted.hprof

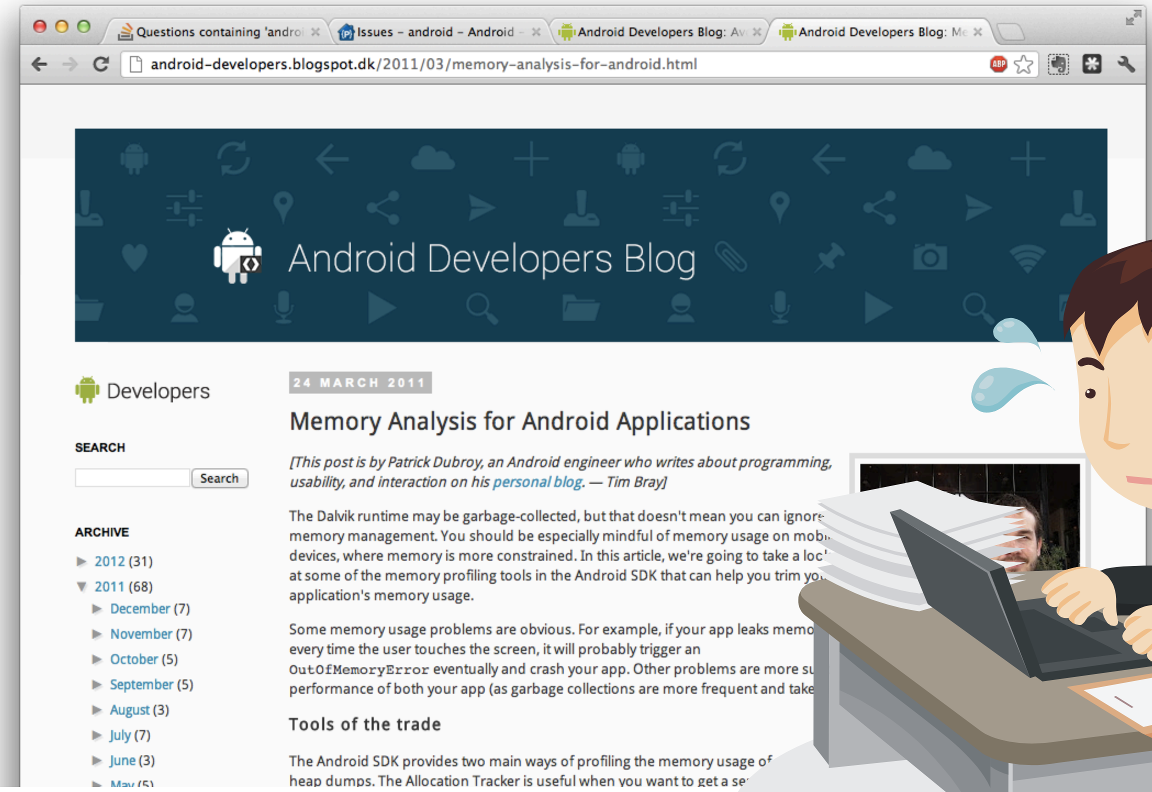
Overview | Histogram | list_objects [selection of 'byte[]'] -inbound

Class Name	Shallow Heap	Retained Heap
<Regex>	<Numeric>	<Numeric>
byte[8] @ 0x429b69c8 HPDS....	24	24
byte[2797568] @ 0x426fe780 '2".&1!.'2$.(3%.&.#.#+ . (...$..."...%...$...#.. +...,7&.2>*.0	2,797,584	2,797,584
mBuffer android.graphics.Bitmap @ 0x40a50fa8	40	2,797,640
value java.util.HashMap$HashMapEntry @ 0x40a4ceb8	24	5,595,472
[13] java.util.HashMap$HashMapEntry[16] @ 0x40805440	80	32,802,960
table java.util.HashMap @ 0x40801a98	48	32,803,008
sBitmapCache class com.example.android.hcgallery.ContentFragment (	8	32,803,056
<class> com.example.android.hcgallery.ContentFragment @ 0x408(	128	384
value java.util.HashMap$HashMapEntry @ 0x408009c0	24	152
Σ Total: 2 entries		
byte[2797568] @ 0x42453768 % ..)$.+& .61+.HA;.F?9.92,.4-'.C;8.MEB.@;8."...-)*.?;<.	2,797,584	2,797,584
byte[2797568] @ 0x421a8750 ......z.FRF.P\P.OXU.NWT.ZUY.ZUY.yvo.....lt\.u]e.z.\.syU.	2,797,584	2,797,584
byte[2797568] @ 0x41efd120 njg.pli.kgd.b^[.da\.olg.tql.qni.roh.urk.wtm.spi.lib.he^.k	2,797,584	2,797,584
byte[3252224] @ 0x41be3108	3,252,240	3,252,240
byte[2797568] @ 0x419380f0	2,797,584	2,797,584
byte[2797568] @ 0x4168d0d8 d.B.d.B.d.@.d.@.f.?.g.@.h.B.h.B.i.B.i.B.h.B.i.C.g.C.f.B.f.C	2,797,584	2,797,584
byte[2797568] @ 0x413e20c0 cR>.eT@.eVA.dU@.aR=.\`Q<.\`Q>.bS@.bS@.bS@.bS@.e'	2,797,584	2,797,584
byte[2797568] @ 0x411370a8	2,797,584	2,797,584
byte[2797568] @ 0x40e8c090	2,797,584	2,797,584
byte[1572864] @ 0x40d0c078 ....[.\`.]dB.ON9.65 .!	1,572,880	1,572,880
byte[2797568] @ 0x40a61060 ....#...#..............................	2,797,584	2,797,584
byte[62100] @ 0x40a51db8	62,112	62,112
byte[24] @ 0x40a4cd11	40	40
byte[4096] @ 0x40a4aa50	4,112	4,112
byte[24] @ 0x40a4a7a1	40	40
byte[4096] @ 0x40a48148	4,112	4,112
byte[24] @ 0x40a464f1	40	40
byte[84] @ 0x40a40560 ....@...@..................@..........................f.M.f.M.f.M...........	96	96
byte[768] @ 0x40a40200 .........................MM..MM..MM..MM..MM..MM..MM..MM..MM.	784	784
byte[1572864] @ 0x408beab8 2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3.8E4.8E·	1,572,880	1,572,880
byte[84] @ 0x408ba188 ....>....._U>...............	96	96
byte[960] @ 0x408b9d68 ....\~(.................]~)..........]~).]~). 0SU]~).]~).]~).].*. 0SU].*.]	976	976
byte[84] @ 0x408b9a48 ....=.....\`S=...............	96	96
byte[960] @ 0x408b9628 ....\}(.................\~(..........\~(.\~(..0QS\~(.\~(.\~(.]~)..0QS]·	976	976
byte[56] @ 0x408b9318 ....H..@P..@...............X..@................. \`S....	72	72
byte[192] @ 0x408b91f8 ..0QS.0RT.0SU.0SU.1TV.1TV.1TV.1UW.1UW.2VX.2WY.2XZ.2XZ	208	208

36M of 81M

Find a bugfix

Somewhere ...



android-developers.blogspot.dk/2011/03/memory-analysis-for-android.html

Android Developers Blog

24 MARCH 2011

Memory Analysis for Android Applications

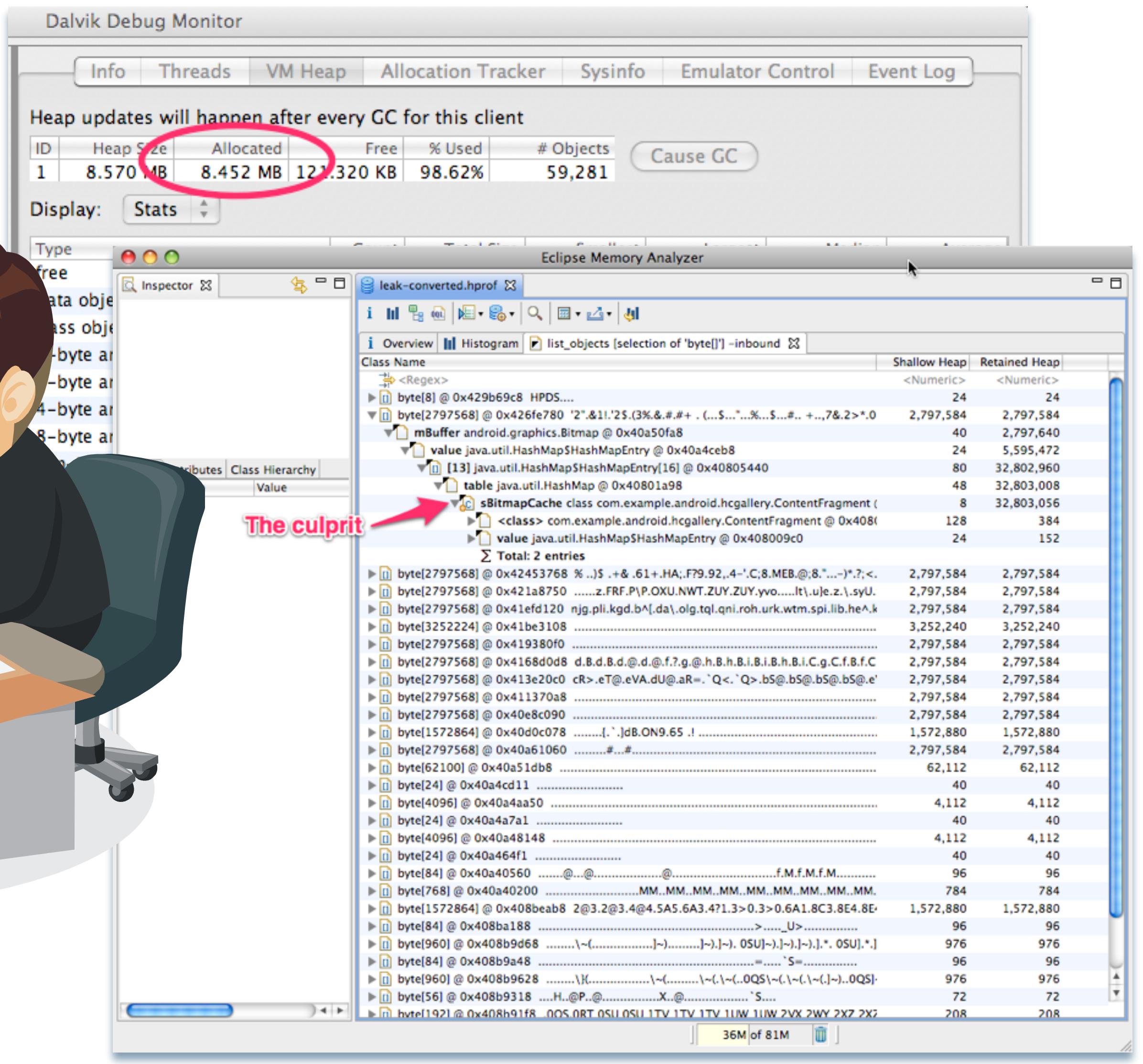
[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignor
memory management. You should be especially mindful of memory usage on mob
devices, where memory is more constrained. In this article, we're going to take a loo
at some of the memory profiling tools in the Android SDK that can help you trim yo
application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo
every time the user touches the screen, it will probably trigger an
OutOfMemoryError eventually and crash your app. Other problems are more su
performance of both your app (as garbage collections are more frequent and take

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of
heap dumps. The Allocation Tracker is useful when you want to get a se



Dalvik Debug Monitor

Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

Eclipse Memory Analyzer

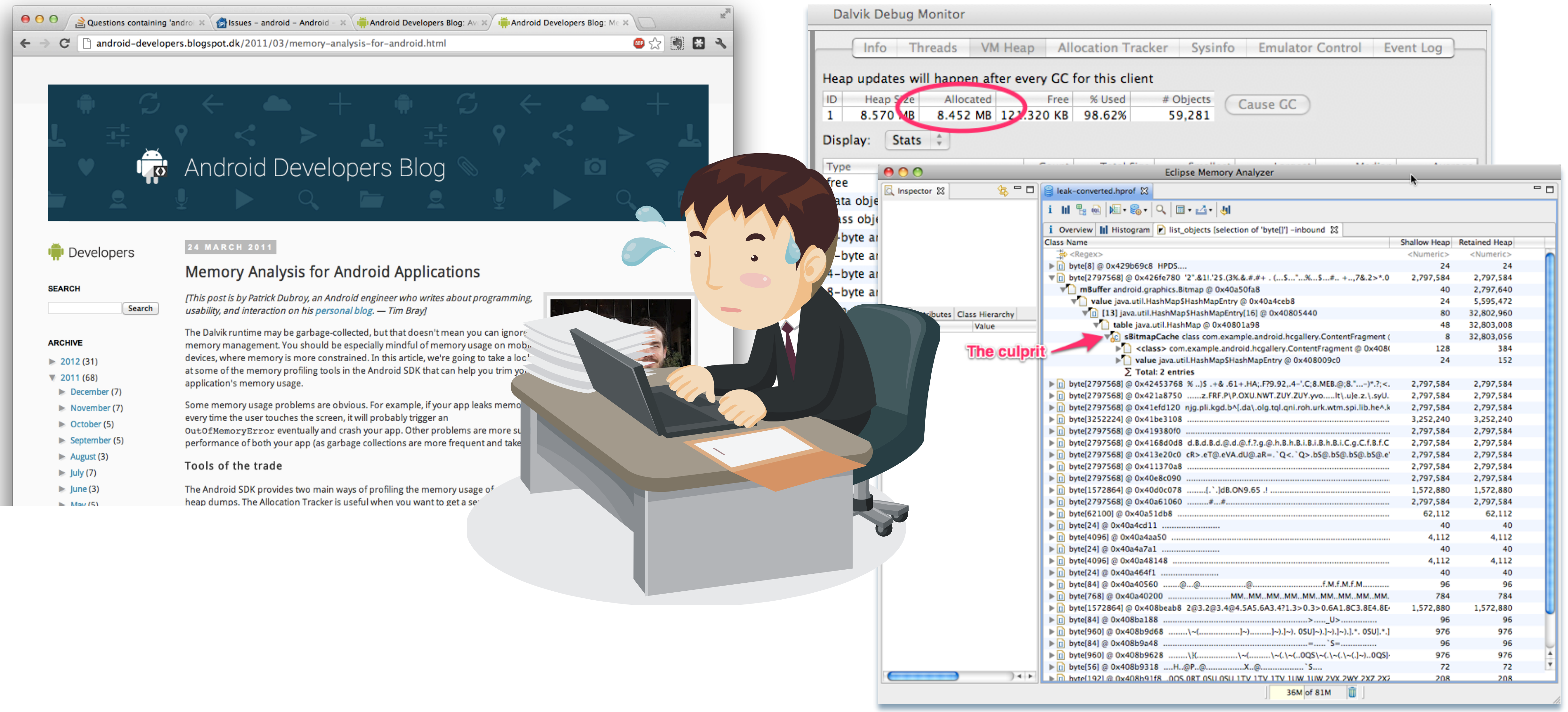
leak-converted.hprof

Overview | Histogram | list_objects [selection of 'byte[]'] -inbound

```
Class Name                                                                          Shallow Heap  Retained Heap
<Regex>                                                                              <Numeric>     <Numeric>
byte[8] @ 0x429b69c8  HPDS....                                                              24            24
byte[2797568] @ 0x426fe780  '2".&1!.'2$.(3%.&.#.#+ . (...$..."...%...$...#.. +..,7&.2>*.0  2,797,584  2,797,584
  mBuffer android.graphics.Bitmap @ 0x40a50fa8                                              40     2,797,640
    value java.util.HashMap$HashMapEntry @ 0x40a4ceb8                                       24     5,595,472
      [13] java.util.HashMap$HashMapEntry[16] @ 0x40805440                                  80    32,802,960
        table java.util.HashMap @ 0x40801a98                                                48    32,803,008
          sBitmapCache class com.example.android.hcgallery.ContentFragment (                 8    32,803,056
            <class> com.example.android.hcgallery.ContentFragment @ 0x408(                128           384
            value java.util.HashMap$HashMapEntry @ 0x408009c0                               24           152
            Σ Total: 2 entries
byte[2797568] @ 0x42453768  % ..)$ .+& .61+.HA;.F?9.92,.4-'.C;8.MEB.@;8."...-)*.?;<.  2,797,584  2,797,584
byte[2797568] @ 0x421a8750  ......z.FRF.P\P.OXU.NWT.ZUY.ZUY.yvo.....It\.u]e.z.\.syU.   2,797,584  2,797,584
byte[2797568] @ 0x41efd120  njg.pli.kgd.b^[.da\.olg.tql.qni.roh.urk.wtm.spi.lib.he^.k  2,797,584  2,797,584
byte[3252224] @ 0x41be3108  ................................................................  3,252,240  3,252,240
byte[2797568] @ 0x419380f0  ................................................................  2,797,584  2,797,584
byte[2797568] @ 0x4168d0d8  d.B.d.B.d.@.d.@.f.?.g.@.h.B.h.B.i.B.i.B.h.B.i.C.g.C.f.B.f.C  2,797,584  2,797,584
byte[2797568] @ 0x413e20c0  cR>.eT@.eVA.dU@.aR=.`Q<.`Q>.bS@.bS@.bS@.bS@.e'              2,797,584  2,797,584
byte[2797568] @ 0x411370a8  ................................................................  2,797,584  2,797,584
byte[2797568] @ 0x40e8c090  ................................................................  2,797,584  2,797,584
byte[1572864] @ 0x40d0c078  ........[.`.]dB.ON9.65 .! ...........................................  1,572,880  1,572,880
byte[2797568] @ 0x40a61060  .........#...#....................................................  2,797,584  2,797,584
byte[62100] @ 0x40a51db8  ................................................................     62,112     62,112
byte[24] @ 0x40a4cd11  ........................                                                   40         40
byte[4096] @ 0x40a4aa50  ................................................................      4,112      4,112
byte[24] @ 0x40a4a7a1  ........................                                                   40         40
byte[4096] @ 0x40a48148  ................................................................      4,112      4,112
byte[24] @ 0x40a464f1  ........................                                                   40         40
byte[84] @ 0x40a40560  .......@...@..................@........................f.M.f.M.f.M...........  96  96
byte[768] @ 0x40a40200  ........................MM..MM..MM..MM..MM..MM..MM..MM..MM.         784        784
byte[1572864] @ 0x408beab8  2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3.8E4.8E·          1,572,880  1,572,880
byte[84] @ 0x408ba188  ....................................................>....._U>...............  96  96
byte[960] @ 0x408b9d68  ........\~(.................]~).........]~).]~). 0SU]~).]~).]~).].*. 0SU].*.]  976  976
byte[84] @ 0x408b9a48  ...................................................=..... `S=...............  96  96
byte[960] @ 0x408b9628  ........\}(.................\~(..........\~(.\~(..0QS\~(.\~(.\~(.]~)..0QS]·  976  976
byte[56] @ 0x408b9318  ....H..@P..@................X..@.................. `S....                  72         72
byte[192] @ 0x408b91f8  .0QS.0RT.0SU.0SU.1TV.1TV.1TV.1UW.1UW.2VX.2WY.2XZ.2XZ            208        208
36M of 81M
```

Find a bugfix
Commit a bugfix

Somewhere …



24 MARCH 2011

Memory Analysis for Android Applications

[This post is by Patrick Dubroy, an Android engineer who writes about programming, usability, and interaction on his personal blog. — Tim Bray]

The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignor memory management. You should be especially mindful of memory usage on mob devices, where memory is more constrained. In this article, we're going to take a loo at some of the memory profiling tools in the Android SDK that can help you trim yo application's memory usage.

Some memory usage problems are obvious. For example, if your app leaks memo every time the user touches the screen, it will probably trigger an `OutOfMemoryError` eventually and crash your app. Other problems are more su performance of both your app (as garbage collections are more frequent and take

Tools of the trade

The Android SDK provides two main ways of profiling the memory usage of heap dumps. The Allocation Tracker is useful when you want to get a se

Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	8.570 MB	8.452 MB	121.320 KB	98.62%	59,281

```
Class Name                                                                                  Shallow Heap   Retained Heap
<Regex>                                                                                        <Numeric>       <Numeric>
byte[8] @ 0x429b69c8  HPDS....                                                                        24              24
byte[2797568] @ 0x426fe780  '2".&1!.'2$.(3%.&.#.#+ . (...$..."...%...$...#.. +..,7&.2>*.0      2,797,584       2,797,584
  mBuffer android.graphics.Bitmap @ 0x40a50fa8                                                        40       2,797,640
    value java.util.HashMap$HashMapEntry @ 0x40a4ceb8                                                 24       5,595,472
      [13] java.util.HashMap$HashMapEntry[16] @ 0x40805440                                            80      32,802,960
        table java.util.HashMap @ 0x40801a98                                                          48      32,803,008
          sBitmapCache class com.example.android.hcgallery.ContentFragment (                           8      32,803,056
            <class> com.example.android.hcgallery.ContentFragment @ 0x408(                           128             384
            value java.util.HashMap$HashMapEntry @ 0x408009c0                                         24             152
            Σ Total: 2 entries
byte[2797568] @ 0x42453768  % ..)$ .+& .61+.HA;.F?9.92,.4-'.C;8.MEB.@;8."...-)*.?;<.          2,797,584       2,797,584
byte[2797568] @ 0x421a8750  ......z.FRF.P\P.OXU.NWT.ZUY.ZUY.yvo.....lt\.u]e.z.\.syU.           2,797,584       2,797,584
byte[2797568] @ 0x41efd120  njg.pli.kgd.b^[.da\.olg.tql.qni.roh.urk.wtm.spi.lib.he^.k          2,797,584       2,797,584
byte[3252224] @ 0x41be3108  ..............................................................   3,252,240       3,252,240
byte[2797568] @ 0x419380f0  ..............................................................   2,797,584       2,797,584
byte[2797568] @ 0x4168d0d8  d.B.d.B.d.@.d.@.f.?.g.@.h.B.h.B.i.B.i.B.h.B.i.C.g.C.f.B.f.C        2,797,584       2,797,584
byte[2797568] @ 0x413e20c0  cR>.eT@.eVA.dU@.aR=.`Q<.`Q>.bS@.bS@.bS@.bS@.e'                    2,797,584       2,797,584
byte[2797568] @ 0x411370a8  ..............................................................   2,797,584       2,797,584
byte[2797568] @ 0x40e8c090  ..............................................................   2,797,584       2,797,584
byte[1572864] @ 0x40d0c078  ........[.`.]dB.ON9.65 .! ......................................   1,572,880       1,572,880
byte[2797568] @ 0x40a61060  ........#...#...................................................   2,797,584       2,797,584
byte[62100] @ 0x40a51db8  ..................................................................      62,112          62,112
byte[24] @ 0x40a4cd11  ........................                                                       40              40
byte[4096] @ 0x40a4aa50  ...................................................................       4,112           4,112
byte[24] @ 0x40a4a7a1  ........................                                                       40              40
byte[4096] @ 0x40a48148  ...................................................................       4,112           4,112
byte[24] @ 0x40a464f1  ........................                                                       40              40
byte[84] @ 0x40a40560  .......@...@..................@..........................f.M.f.M.f.M...........       96              96
byte[768] @ 0x40a40200  ..........................MM..MM..MM..MM..MM..MM..MM..MM..MM.             784             784
byte[1572864] @ 0x408beab8  2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3.8E4.8E·                  1,572,880       1,572,880
byte[84] @ 0x408ba188  .........................................................>....._U>...............       96              96
byte[960] @ 0x408b9d68  ........\~(.................]~).........]~).]~). 0SU]~).]~).]~).].*. 0SU].*.]      976             976
byte[84] @ 0x408b9a48  .........................................................=.....`S=...............       96              96
byte[960] @ 0x408b9628  ........\}(.................\~(.........\~(.\~(..0QS\~(.\~(.\~(.]~)..0QS]·       976             976
byte[56] @ 0x408b9318  ....H..@P..@................X..@.................`S....                       72              72
byte[192] @ 0x408b91f8  .0QS.0RT.0SU.0SU.1TV.1TV.1TV.1UW.1UW.2VX.2WY.2XZ.2XZ                       208             208
```

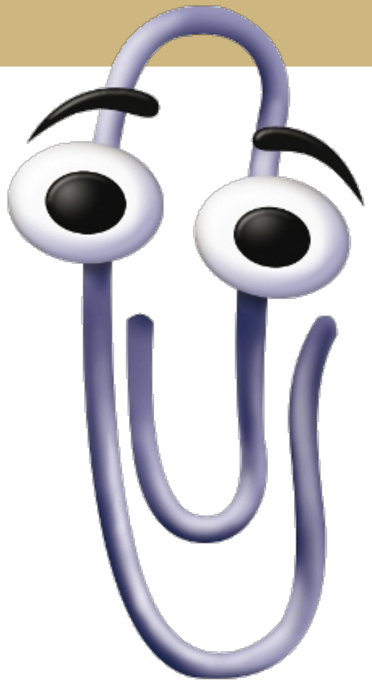
Find a bugfix
Commit a bugfix
Bugfix is picked up by **Fixr**

GitHub

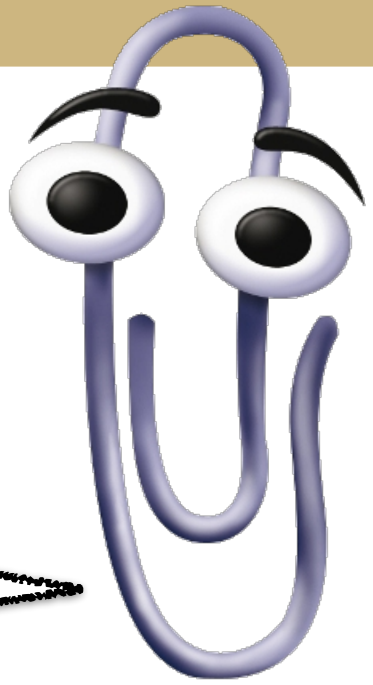
BUGFIX
Fixr

A **Fixr**-enabled IDE responds …

A **Fixr**-enabled IDE responds …



A **Fixr**-enabled IDE responds …

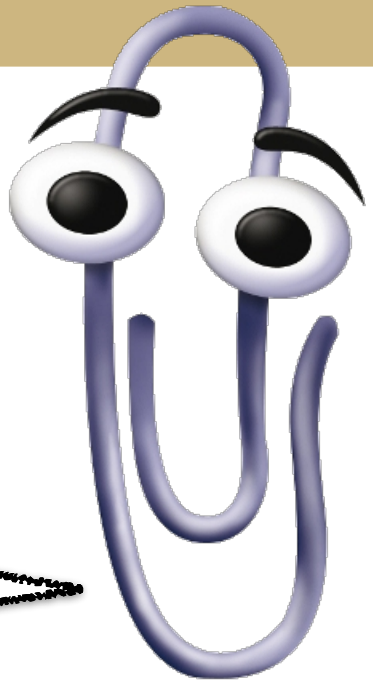


It looks like you've created a memory leak like and

100,000 others. Would you like to apply ?

I don't know how I created a memory leak!

A Fixr-enabled IDE responds ...

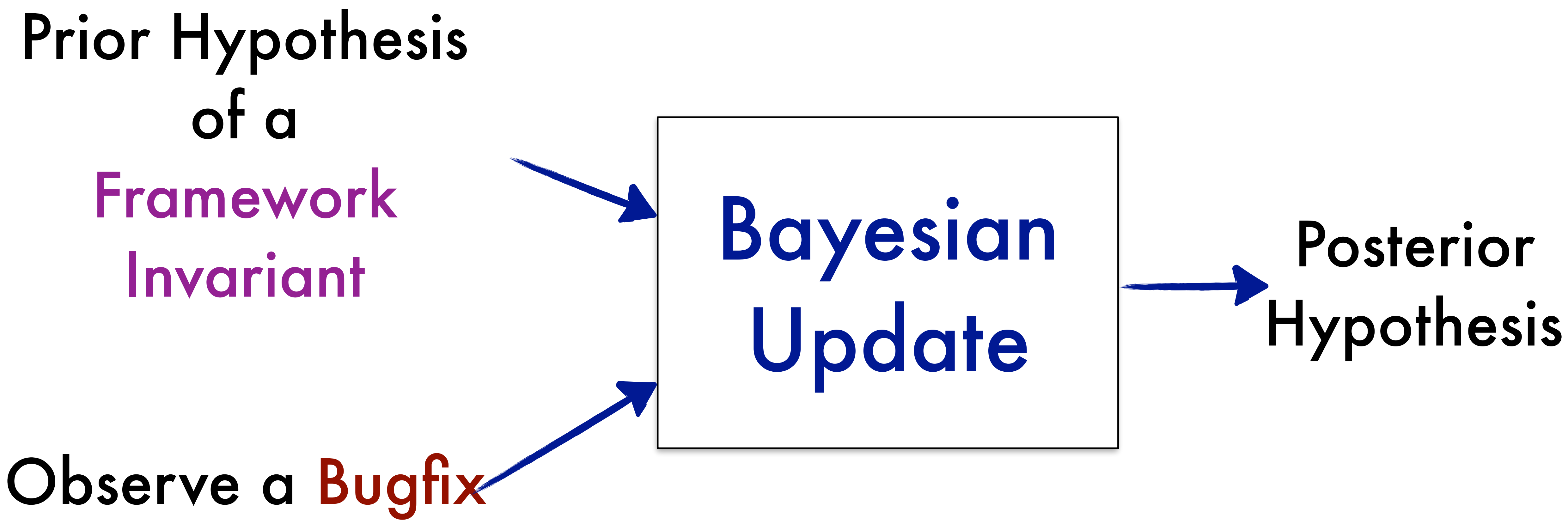


It looks like you've created a memory leak like and 100,000 others. Would you like to apply ?

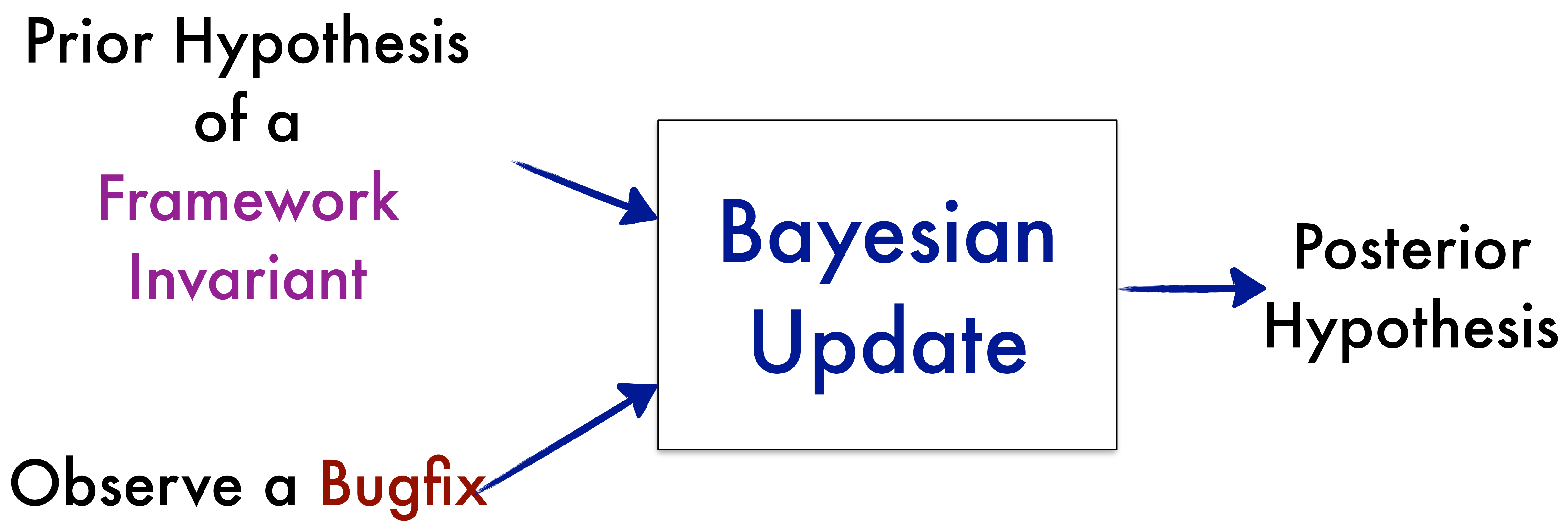
the bugfix is "transferred"

I don't know how I created a memory leak!

Summary: Mine framework specifications with bugfixes



Summary: Mine framework specifications with bugfixes



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

Bug
(on Android <4)

aView.setTag(..., anObject)

Bug
(on Android <4)

aView.setTag(..., anObject)

$$\begin{tikzcd}
\text{aView} & \text{anObject} \arrow[l, bend left]
\end{tikzcd}$$

if anObject can reach aView

Bug
(on Android <4)

aView.setTag(..., anObject)

if anObject can reach aView

Framework Invariant

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

Bug
(on Android <4)

aView.setTag(..., anObject)

if anObject can reach aView

Framework Invariant

```
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)

aView.setTag(..., anObject)

if anObject can reach aView

Framework Invariant

```
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)

A Fix

Bug
(on Android <4)

aView.setTag(..., anObject)

if anObject can reach aView

Framework Invariant

```
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)

A Fix

~~aView.setTag(id, anObj)~~

aView.setTag(new Holder(id,anObj))

uses mTag instead

Bug
(on Android <4)

aView.setTag(..., anObject)

bug condition

if anObject can reach aView

Framework
Invariant

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

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

Challenges

How do we find bugfix commits?

Challenges

How do we find bugfix commits?

an instance of a View.setTag fix

Challenges

How do we find bugfix commits?

an instance of a View.setTag fix

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

Challenges

How do we find bugfix commits?

an instance of a View.setTag fix

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

a specification of the View.setTag repair

Challenges

How do we find bugfix commits?

an instance of a View.setTag fix

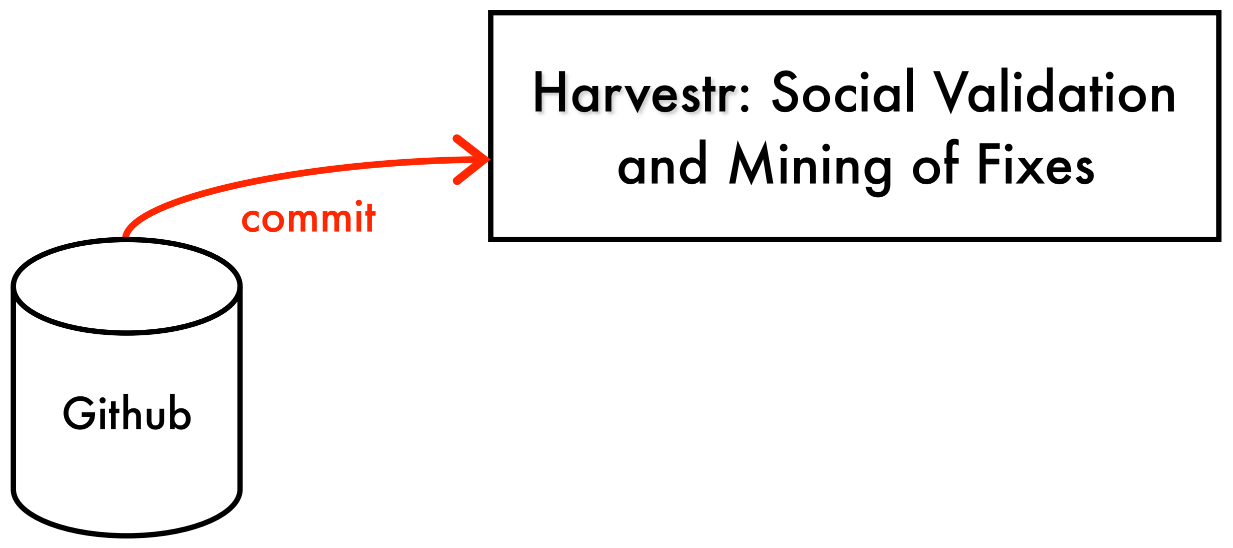
Given a bugfix commit, how do we
and

a specification of the View.setTag repair

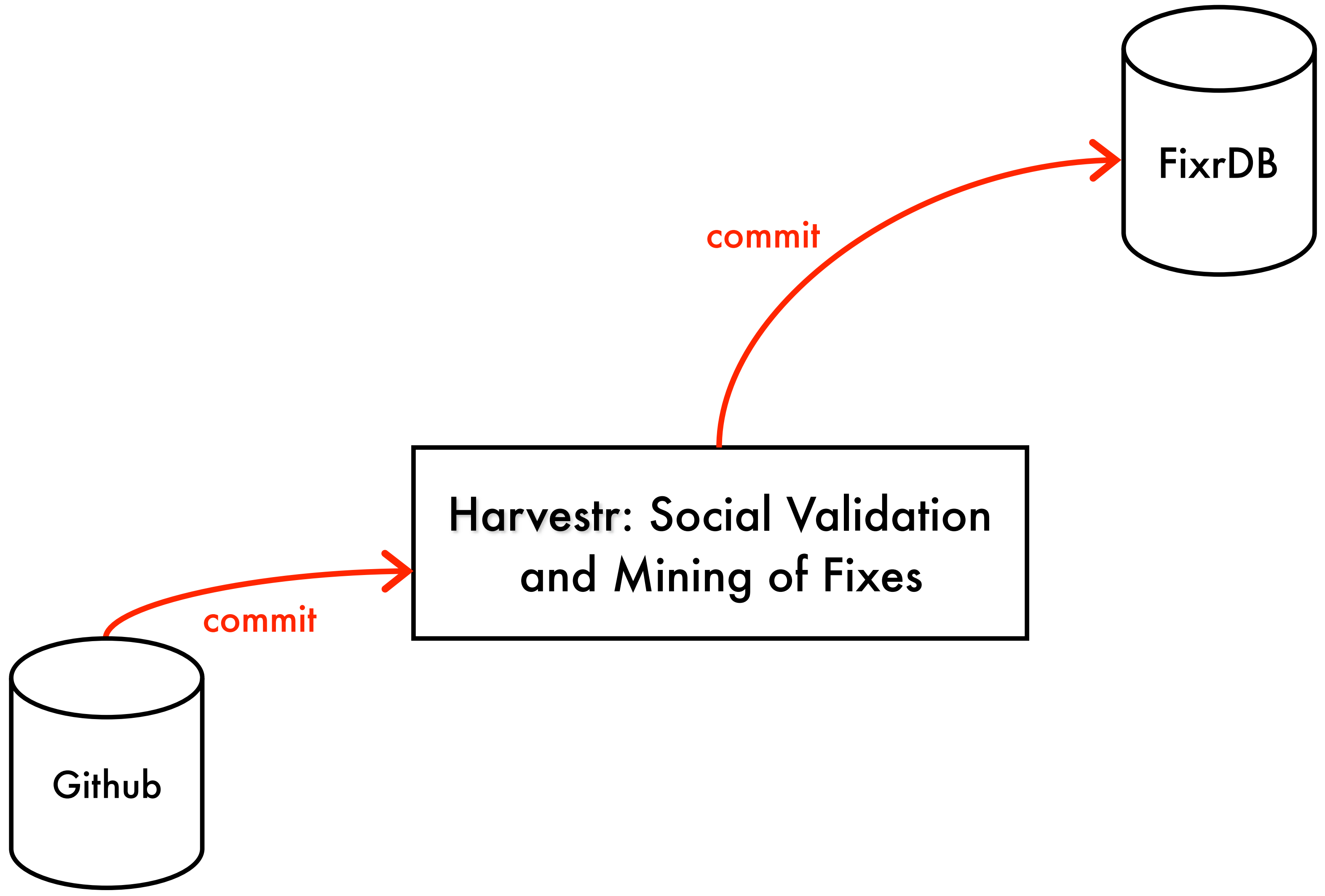
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

Harvestr: Social Validation and Mining of Fixes

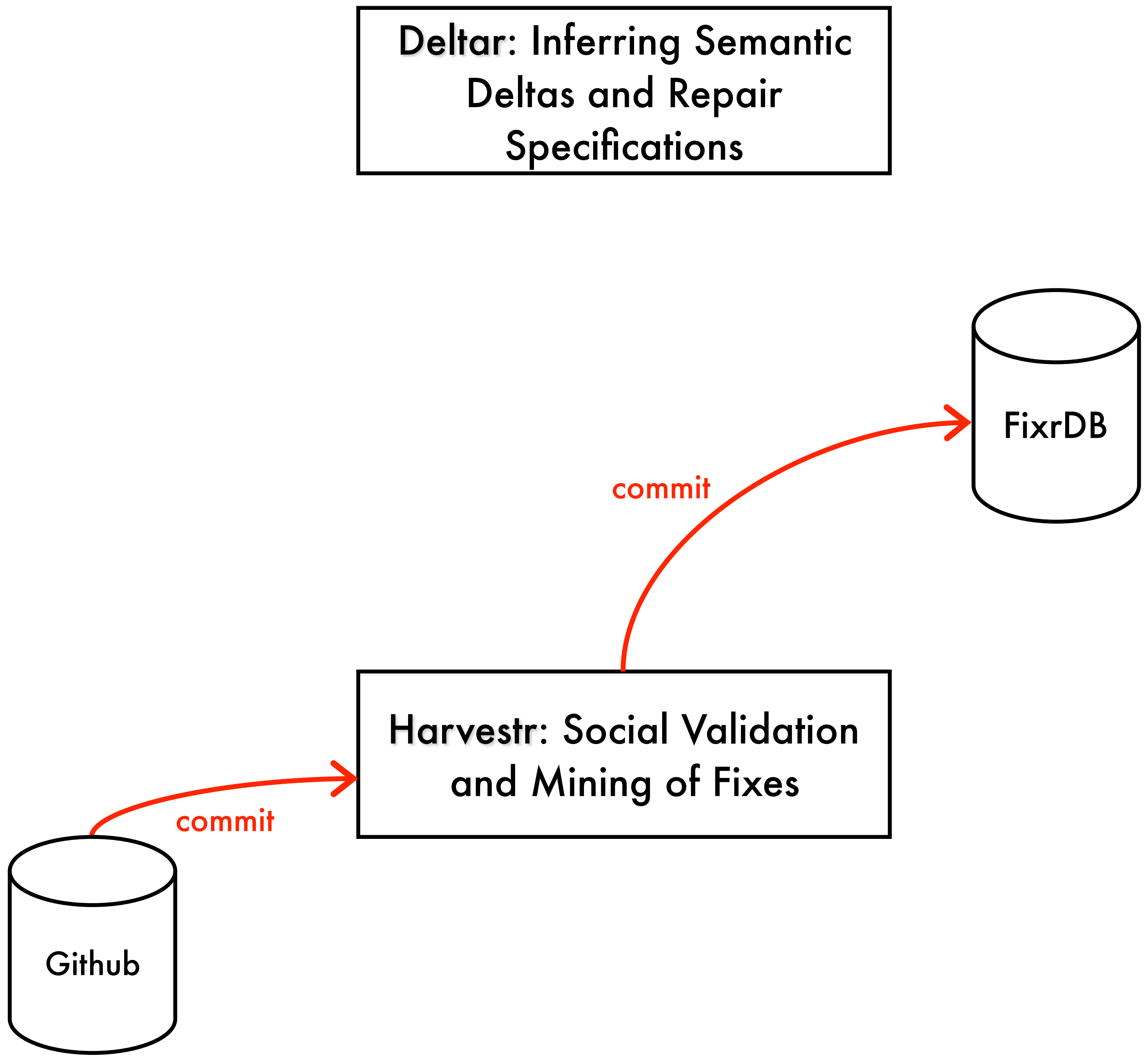
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation



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

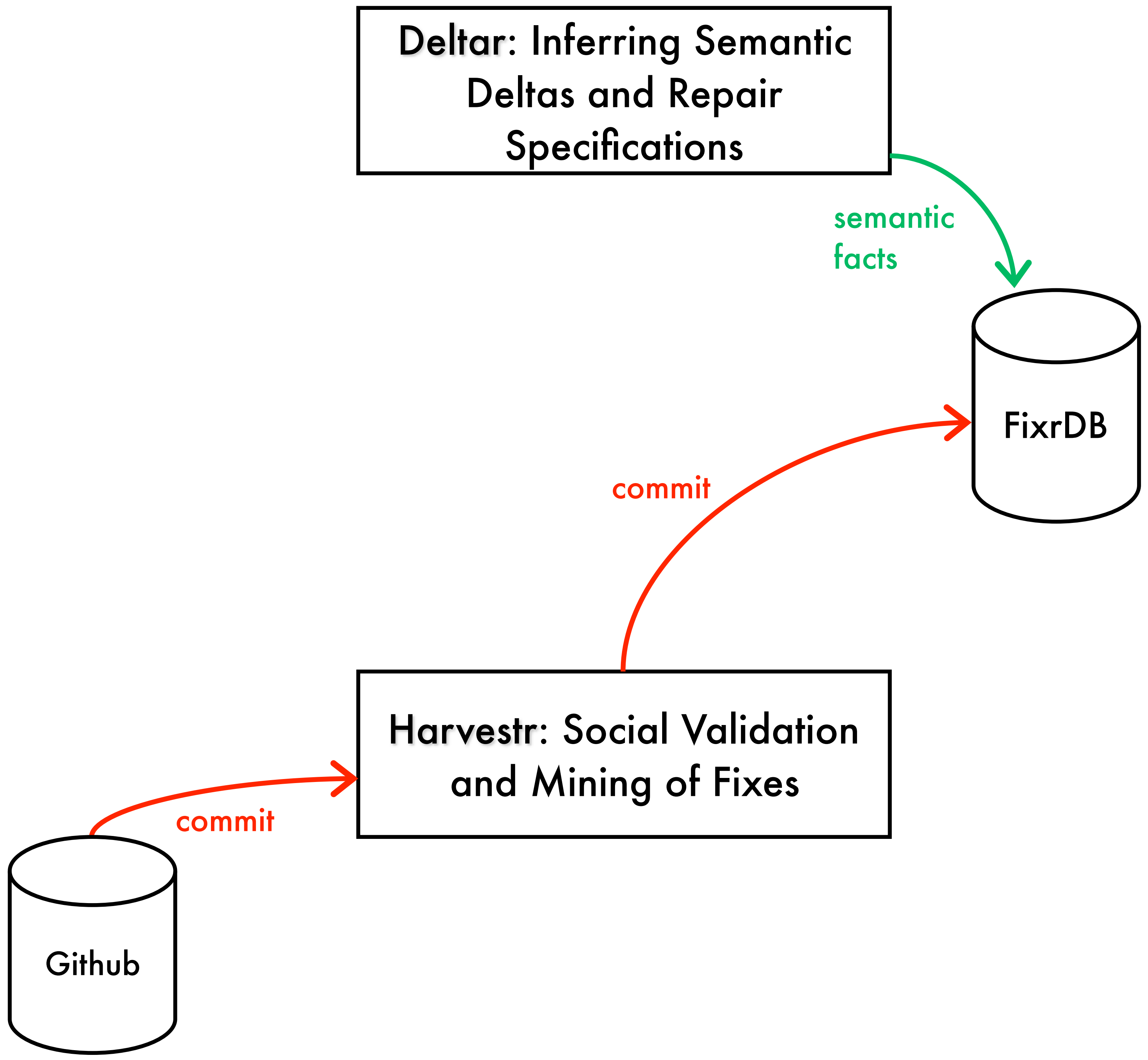


Workflow 0: Continuous commit harvesting, buggy app patching, and social validation

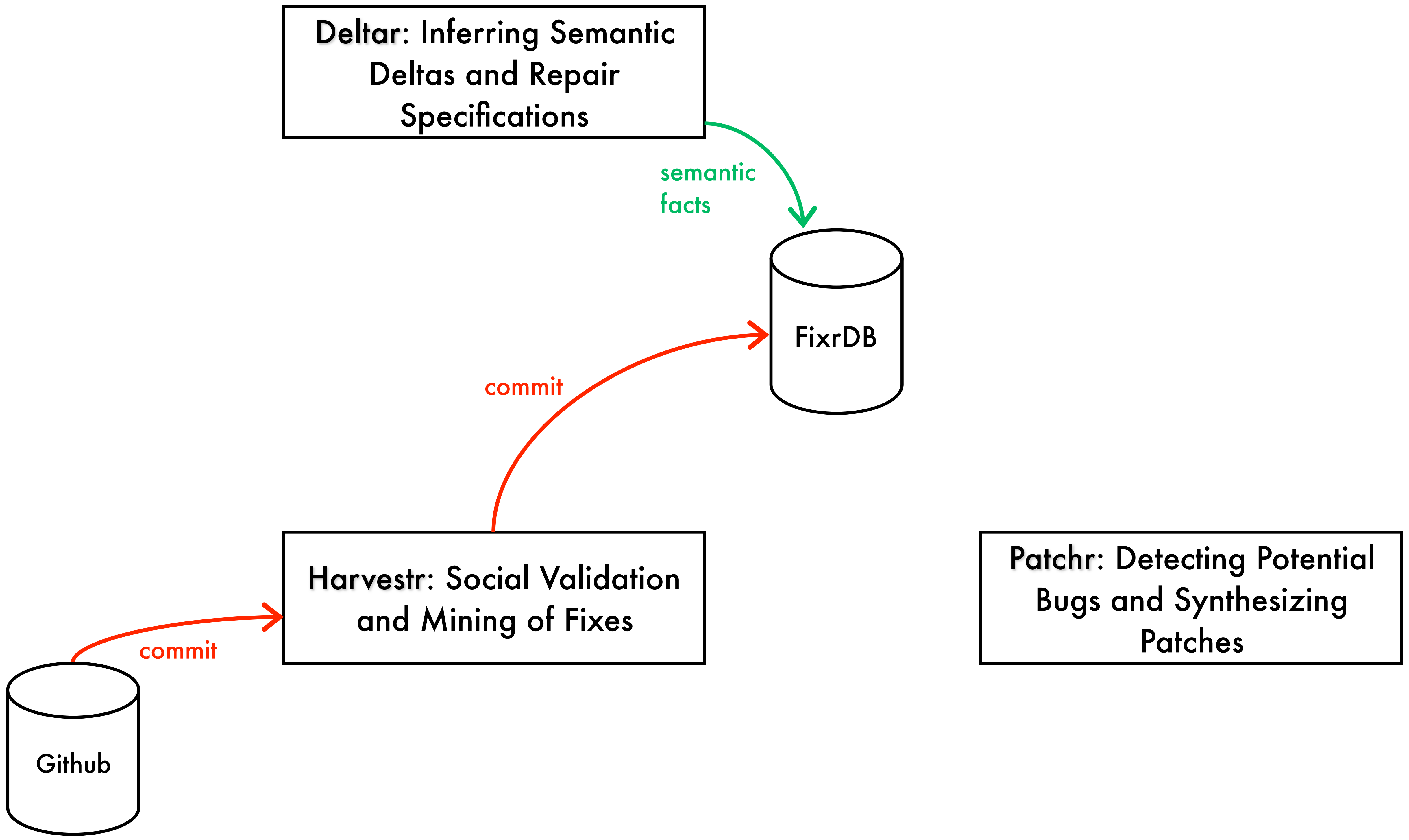


Deltar: Inferring Semantic Deltas and Repair Specifications
FixrDB
commit
Harvestr: Social Validation and Mining of Fixes
commit
Github

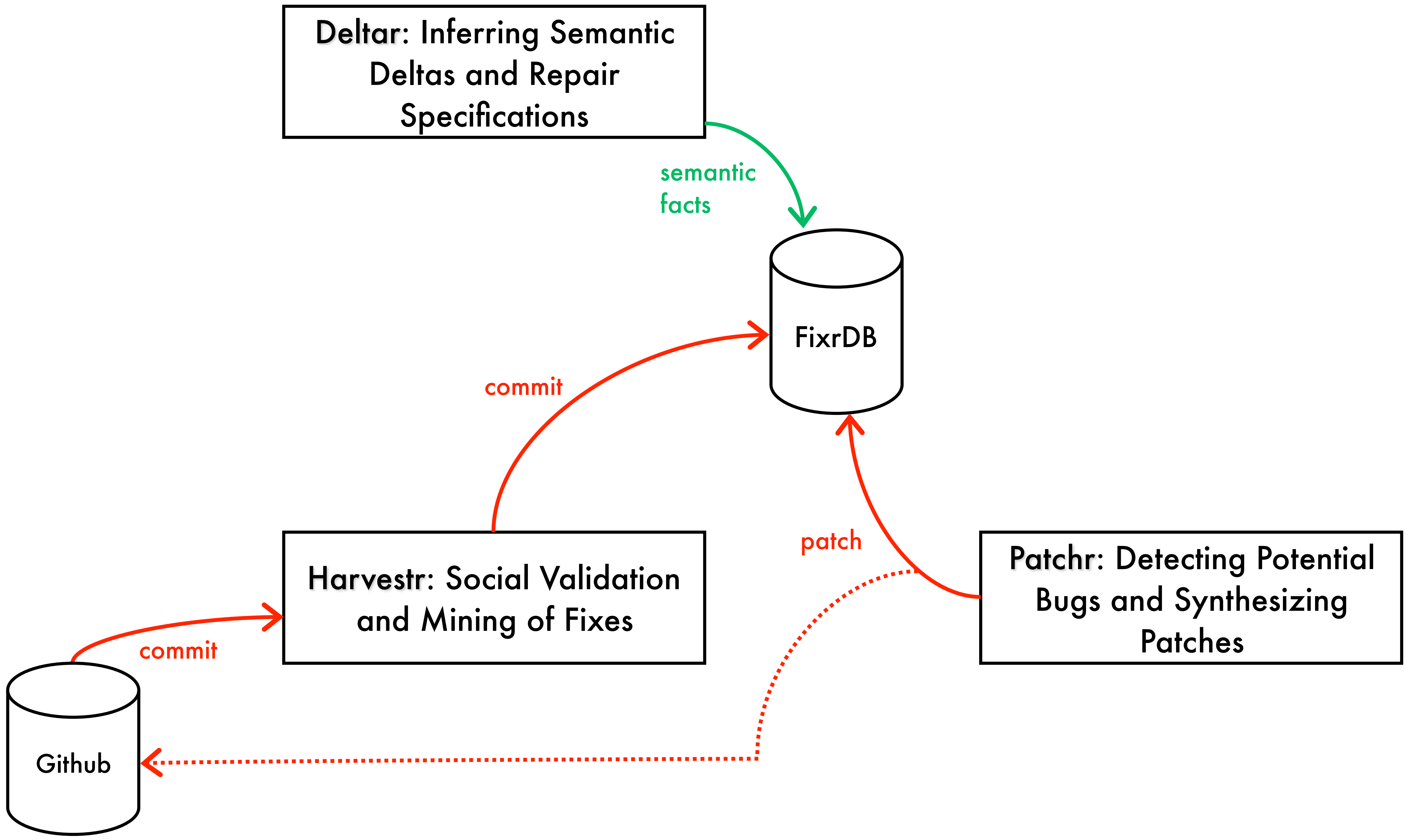
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


Deltar: Inferring Semantic Deltas and Repair Specifications
semantic facts
FixrDB
commit
Harvestr: Social Validation and Mining of Fixes
commit
Github

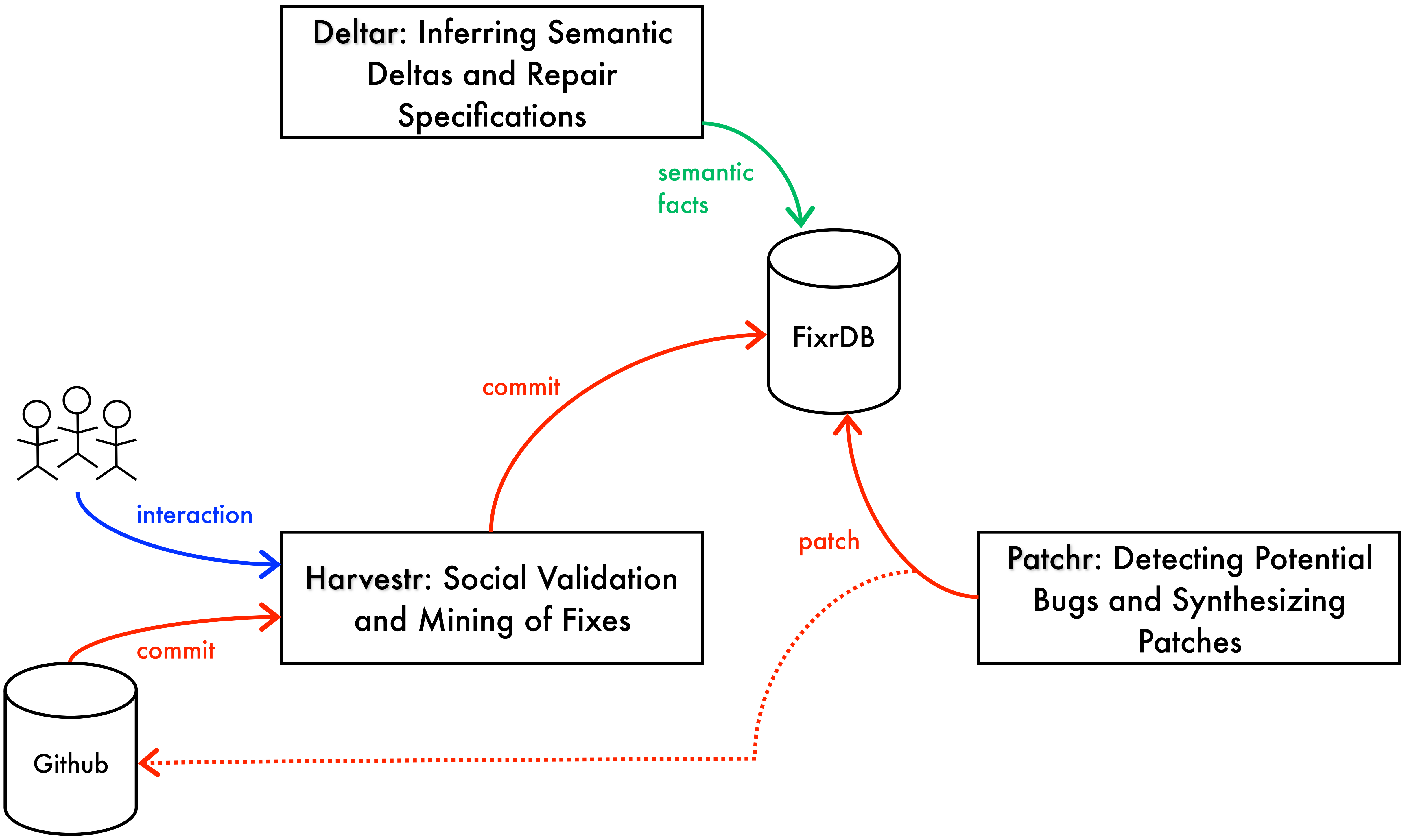
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


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

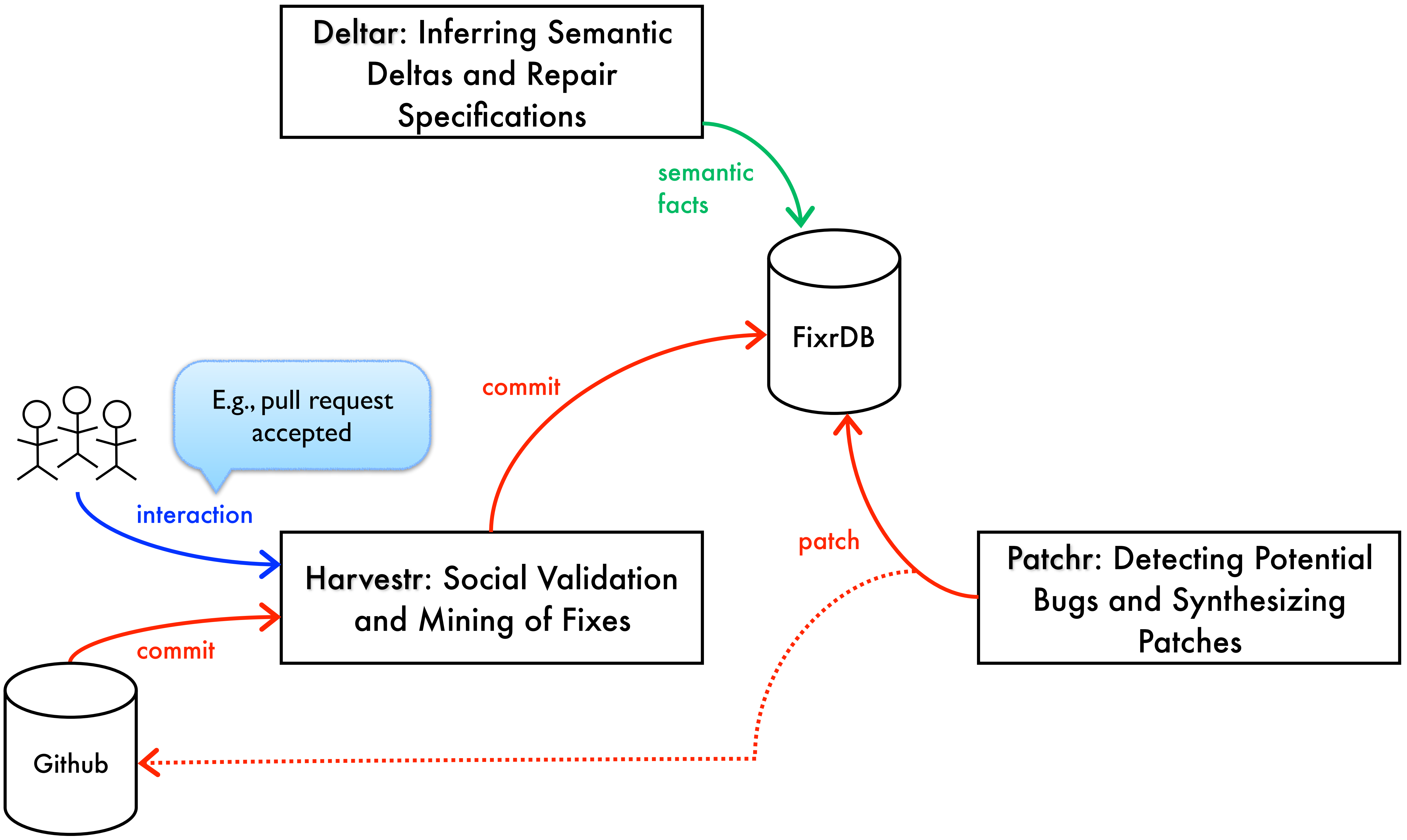
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


Deltar: Inferring Semantic Deltas and Repair Specifications
semantic facts
FixrDB
commit
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

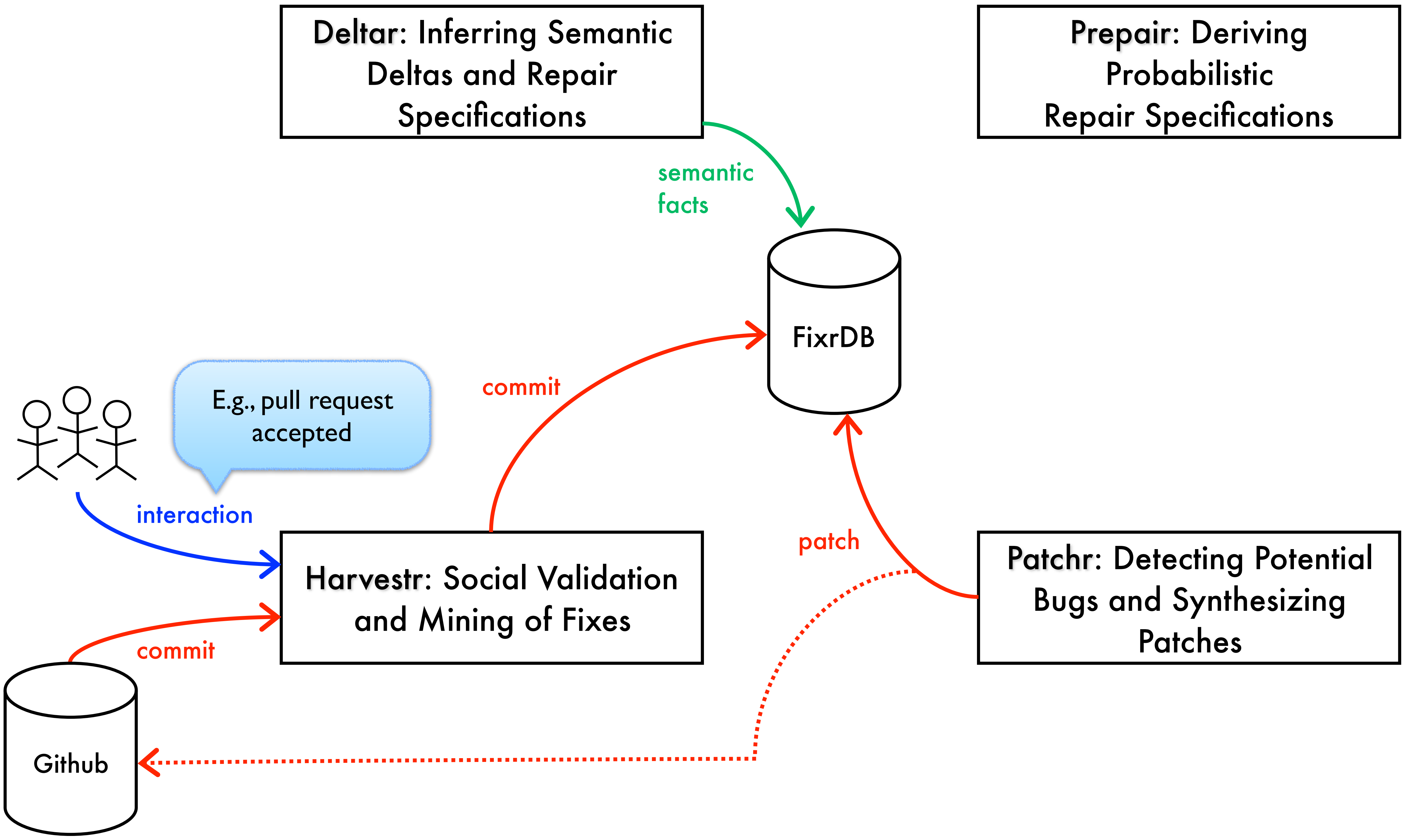
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


Deltar: Inferring Semantic Deltas and Repair Specifications
semantic facts
FixrDB
commit
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

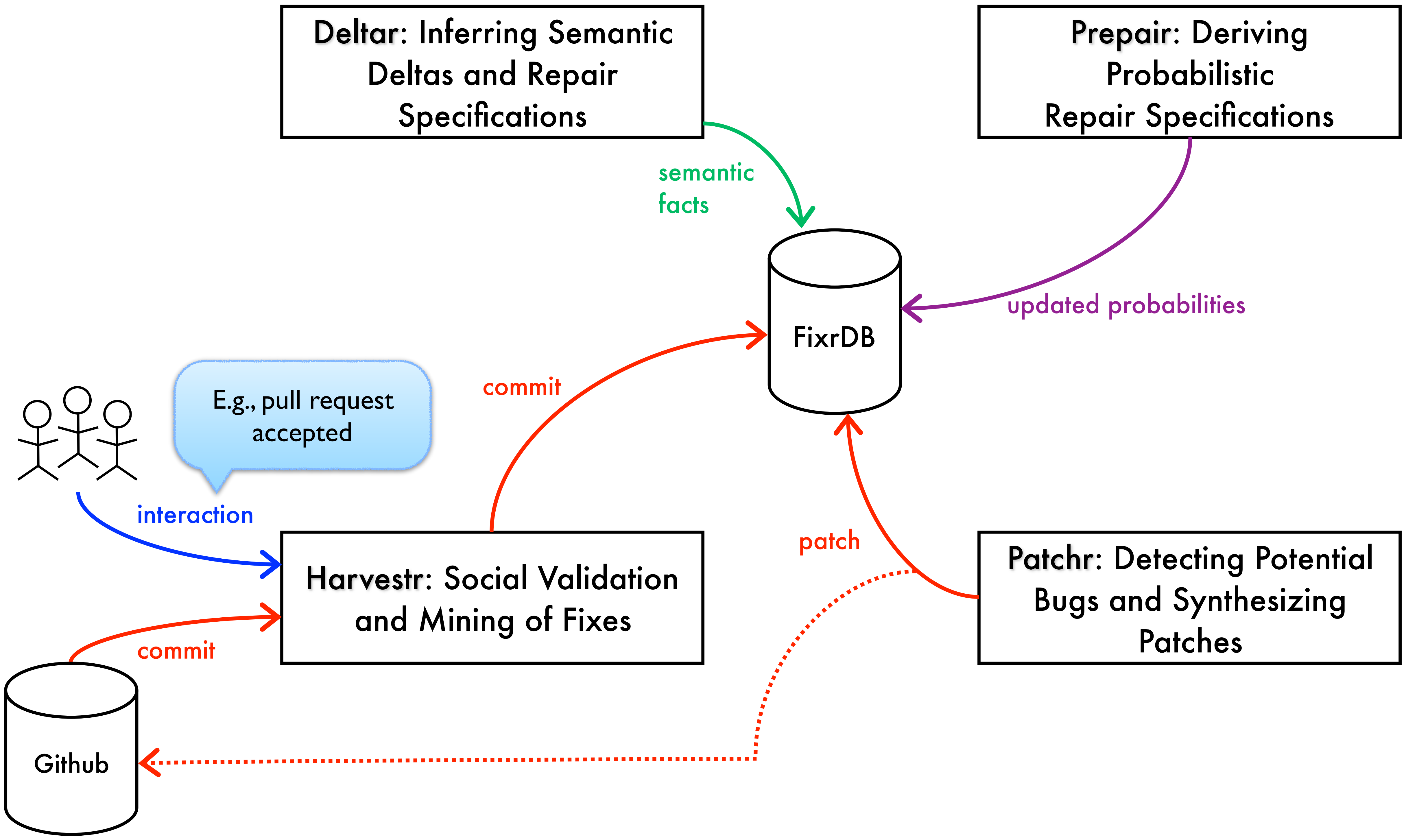
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


Deltar: Inferring Semantic Deltas and Repair Specifications
semantic facts
FixrDB
E.g., pull request accepted
commit
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

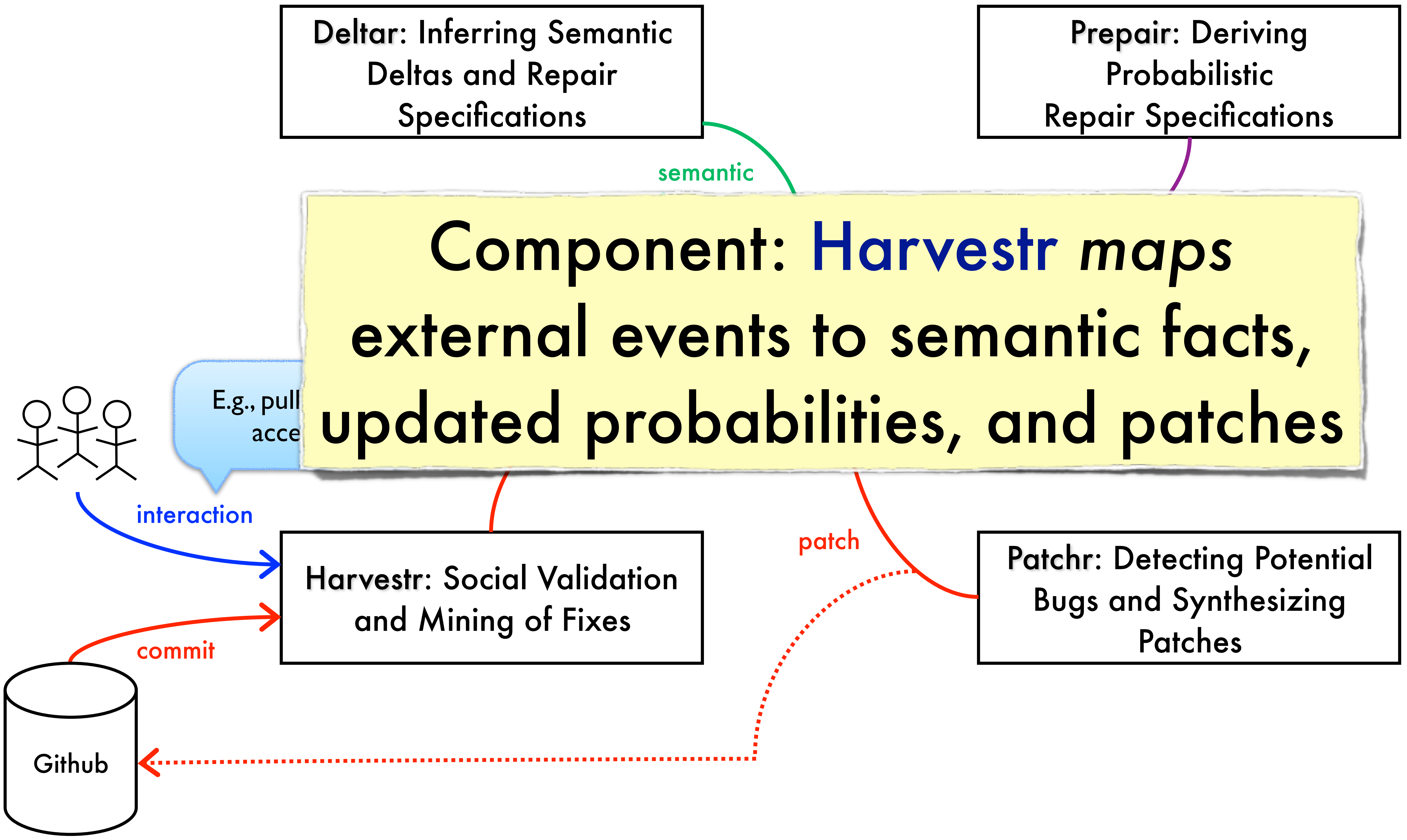
Workflow 0: Continuous commit harvesting, buggy app patching, and social validation


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
semantic facts
FixrDB
commit
E.g., pull request accepted
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

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

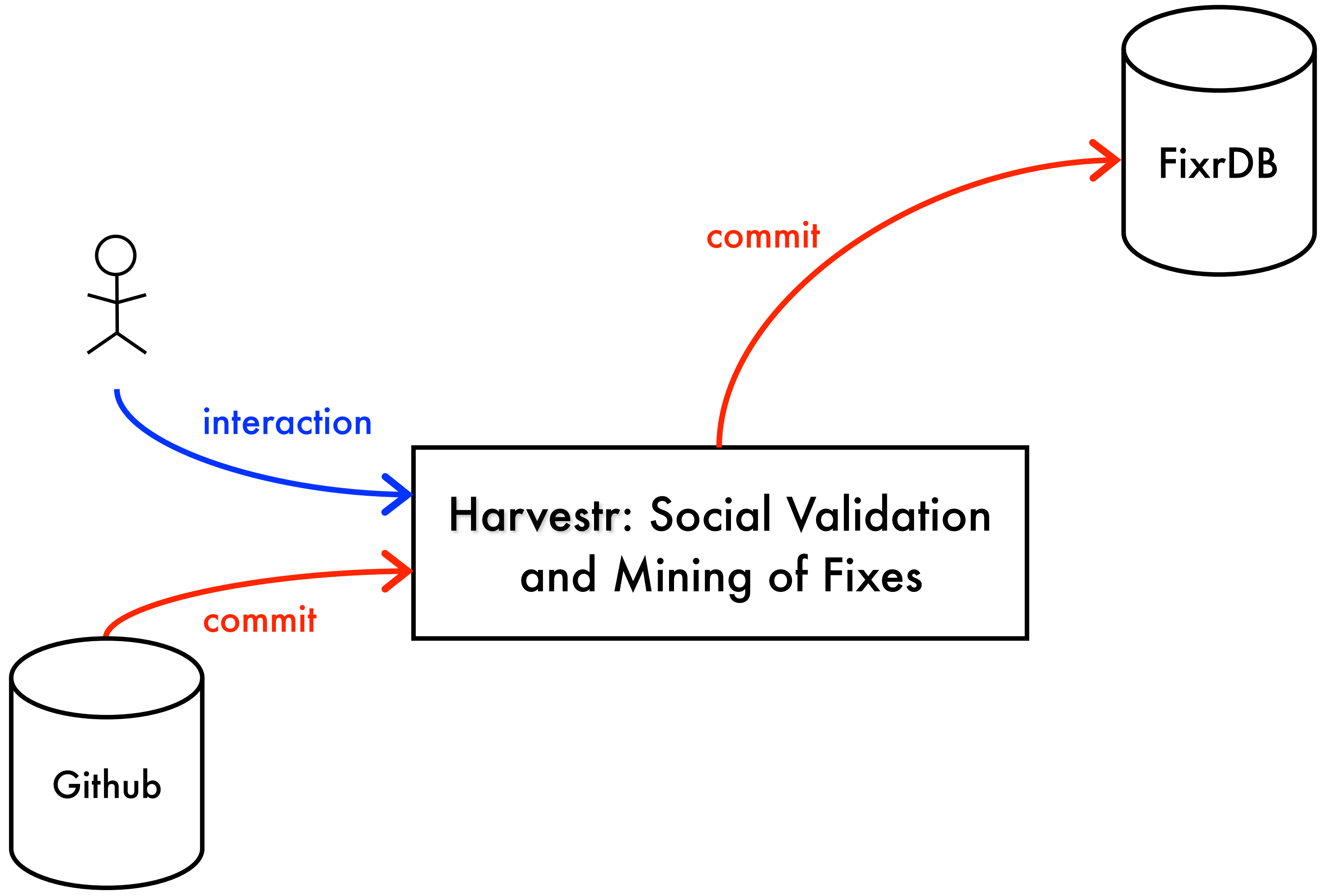

Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
semantic facts
FixrDB
updated probabilities
E.g., pull request accepted
commit
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

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


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
semantic
E.g., pull
acce
interaction
Harvestr: Social Validation and Mining of Fixes
commit
Github
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches

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

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



Identifying interesting commits with a trend analysis

Identifying interesting commits with a trend analysis

Idea: Enable an analyst (or a tool) to look for trends in commits over time.

Identifying interesting commits with a trend analysis

Idea: Enable an analyst (or a tool) to look for trends in commits over time.

Analyst's Hypothesis: A high number of commits involving an API method in a period might correspond to community-driven bug fixing.

Identifying interesting commits with a trend analysis

Idea: Enable an analyst (or a tool) to look for trends in commits over time.

Analyst's Hypothesis: A high number of commits involving an API method in a period might correspond to community-driven bug fixing.

Another Hypothesis: A sharp increase across periods in commits involving an API method might be bug fixing.

Identifying interesting commits with a trend analysis

Idea: Enable an analyst (or a tool) to look for trends in commits over time.

Analyst's Hypothesis: A high number of commits involving an API method in a period might correspond to community-driven bug fixing.

Another Hypothesis: A sharp increase across periods in commits involving an API method might be bug fixing.

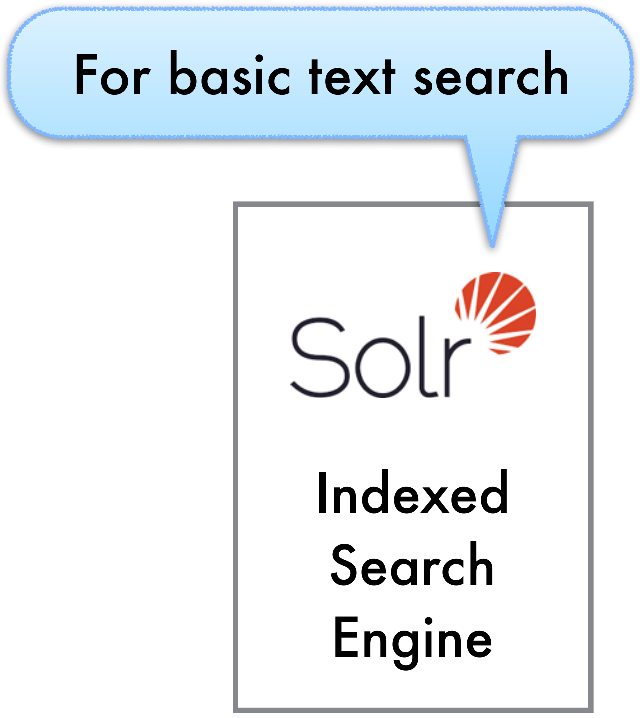
Prototype tools for trend analysis

A prototype trend analysis

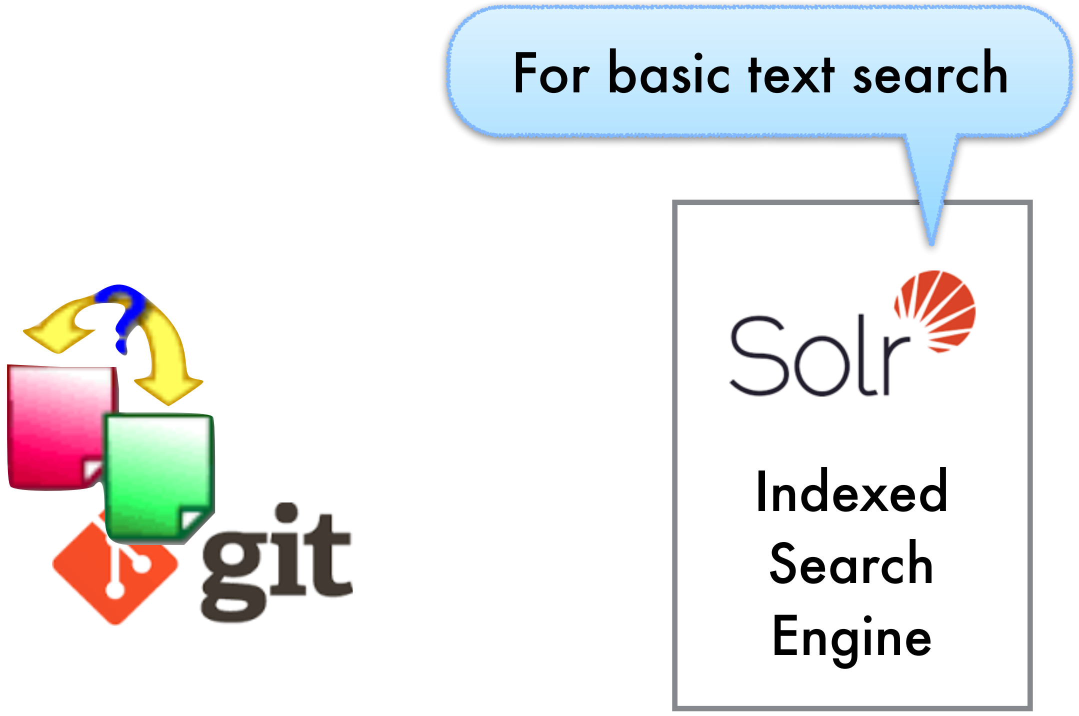
A prototype trend analysis

Indexed
Search
Engine

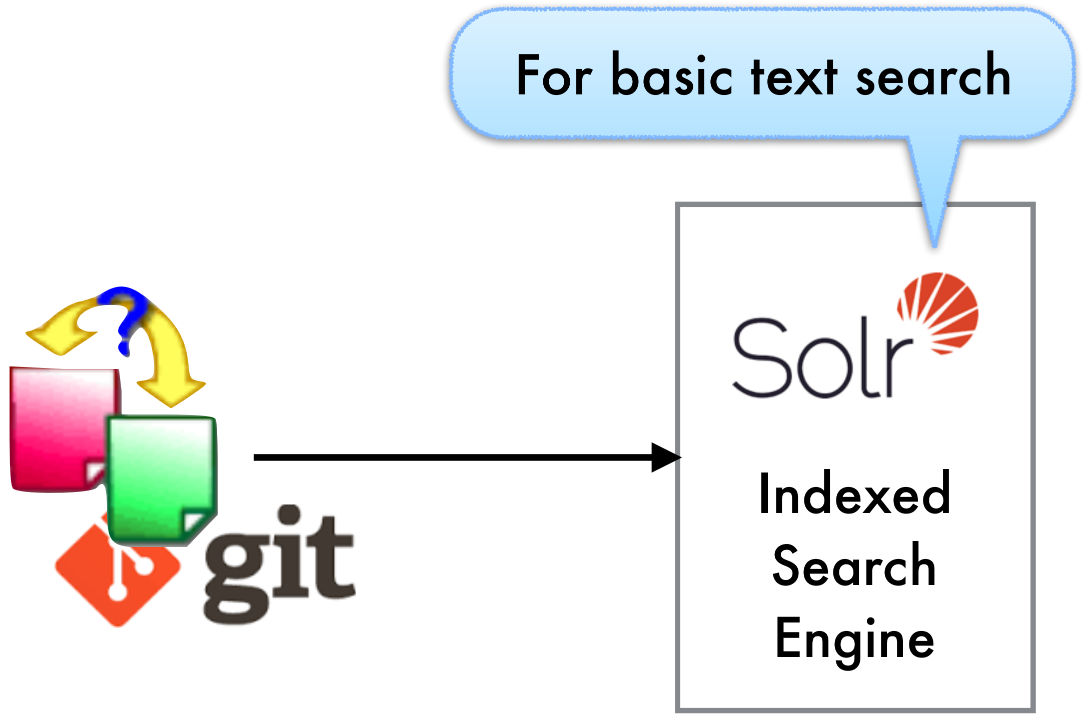
A prototype trend analysis



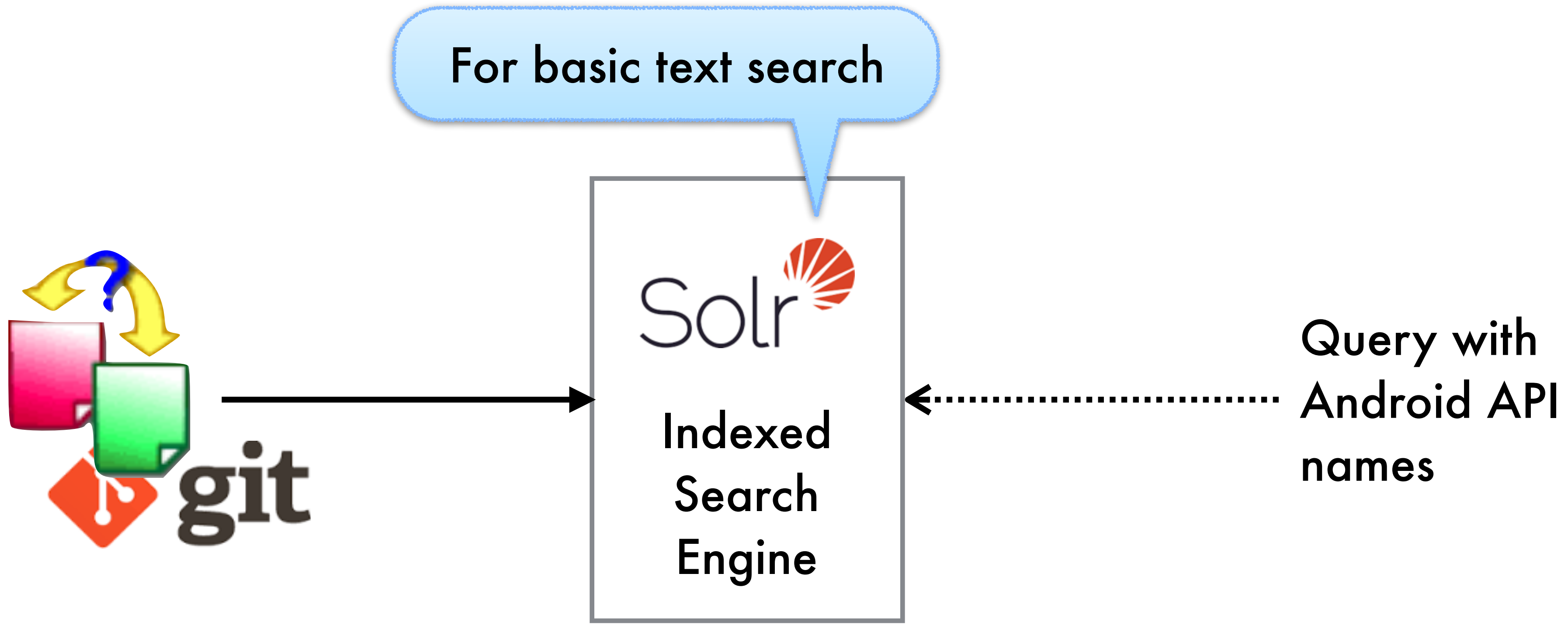
A prototype trend analysis



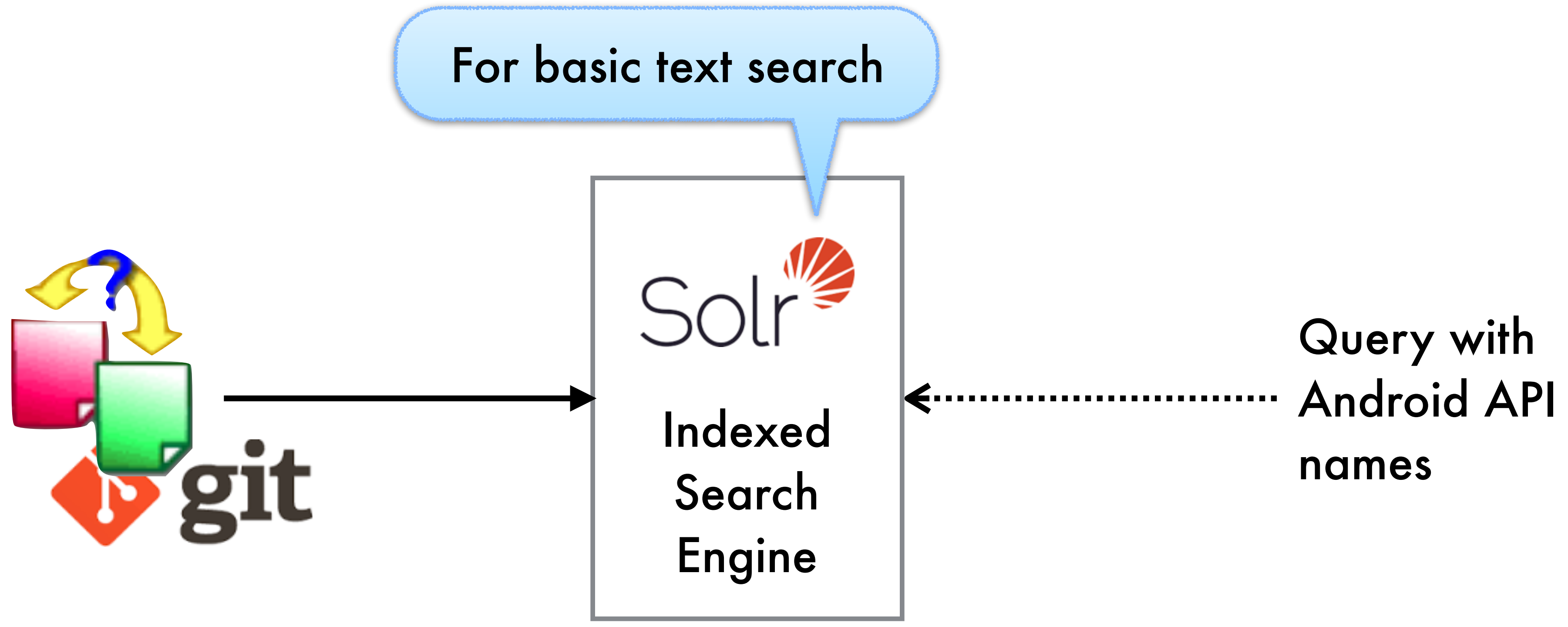
A prototype trend analysis

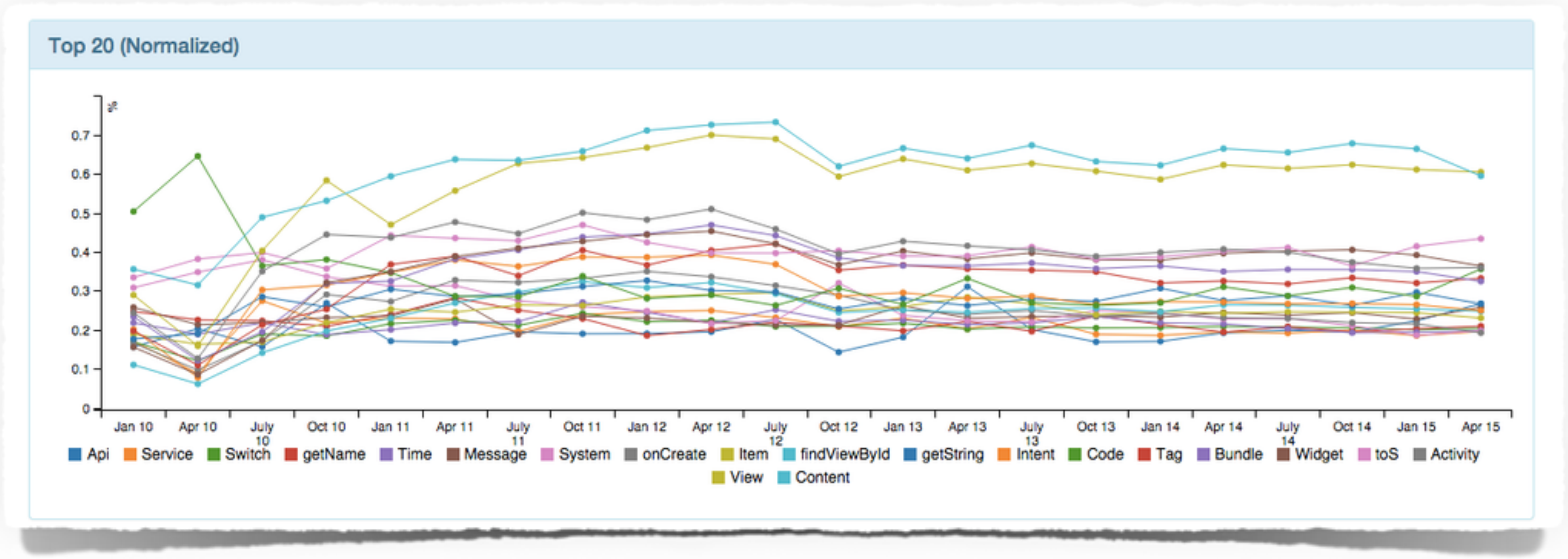


A prototype trend analysis

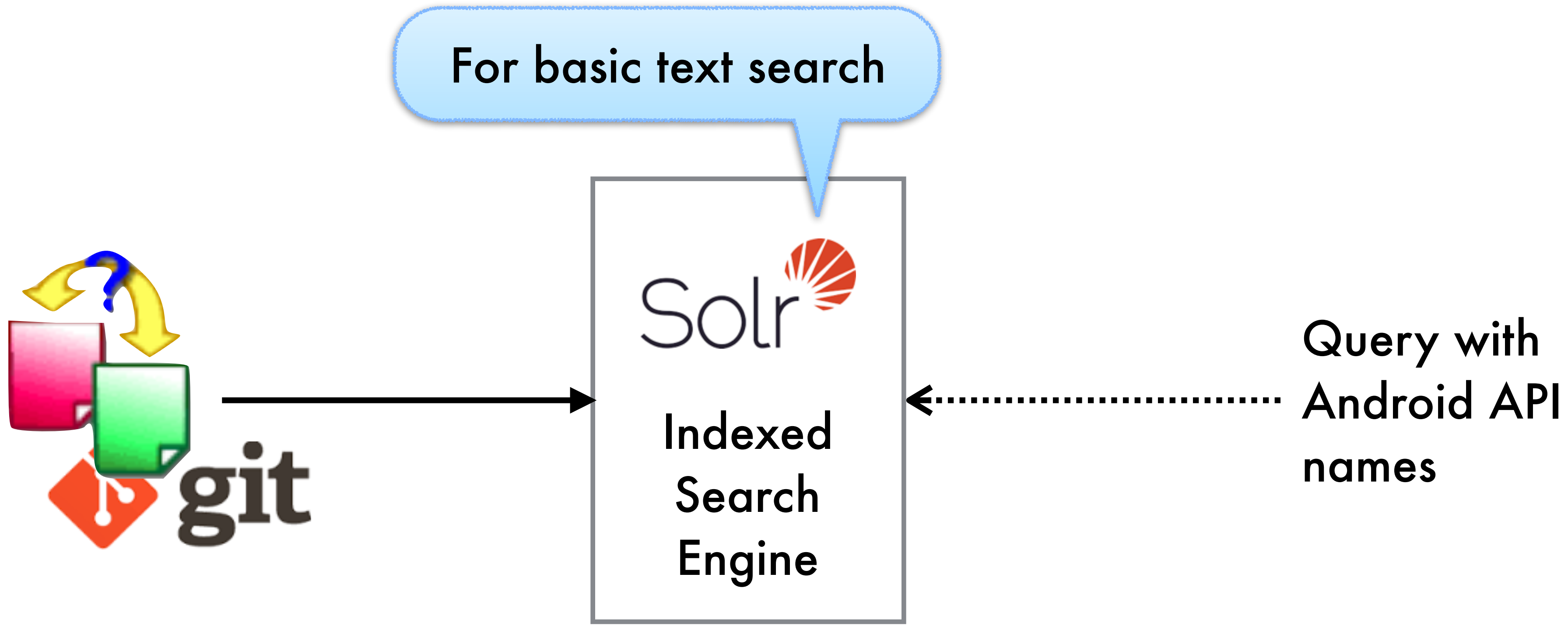


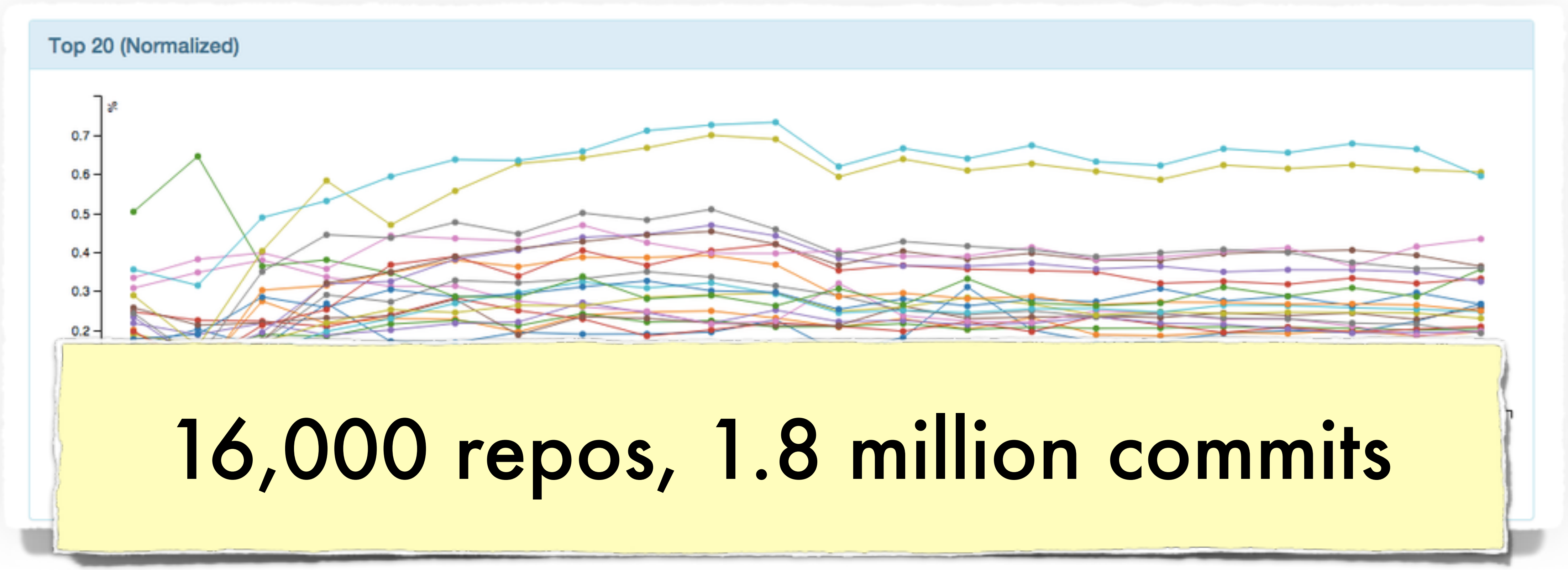
A prototype trend analysis





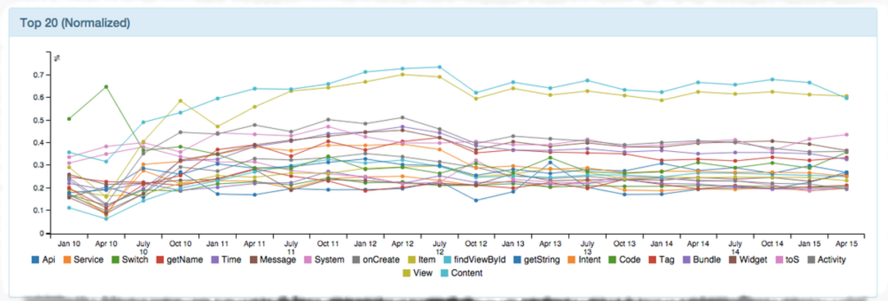
A prototype trend analysis

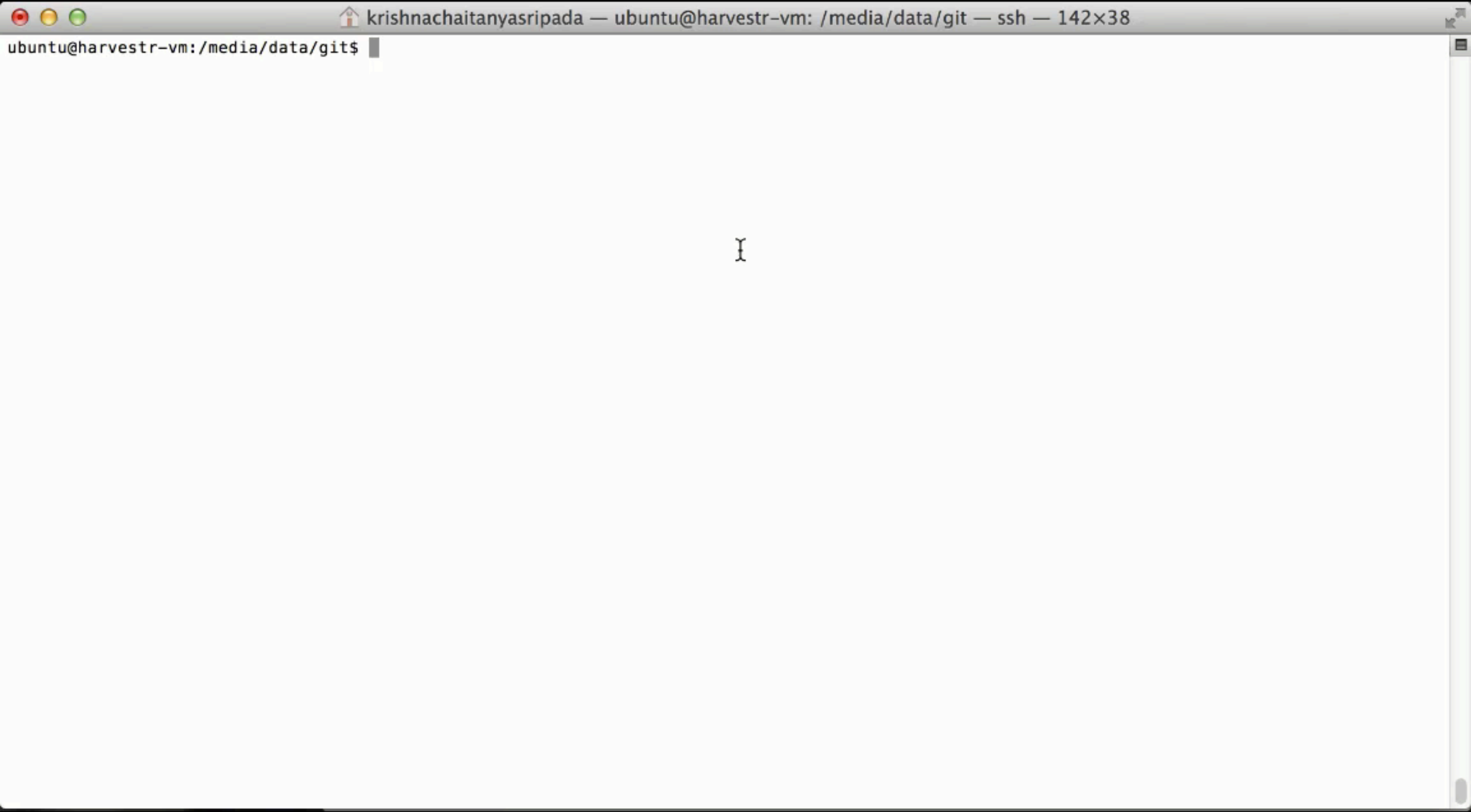




16,000 repos, 1.8 million commits

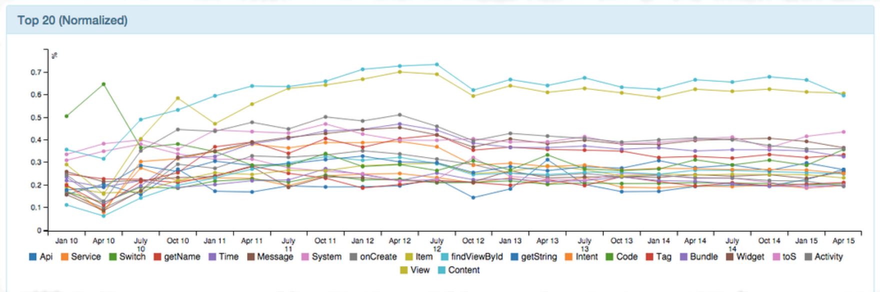
A prototype trend analysis

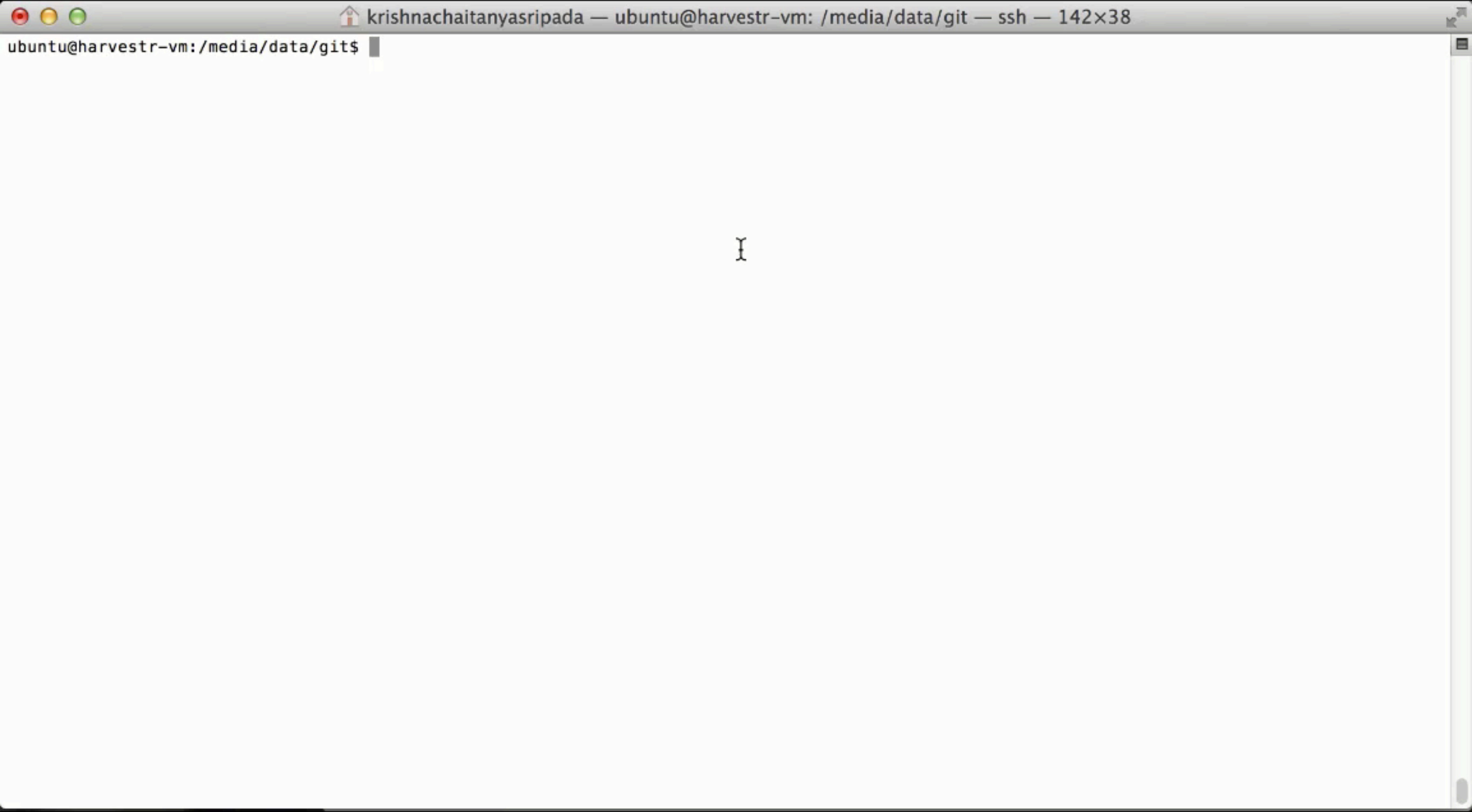




```
krishnachaitanyasripada — ubuntu@harvestr-vm: /media/data/git — ssh — 142×38
ubuntu@harvestr-vm:/media/data/git$ 
```

A prototype trend analysis

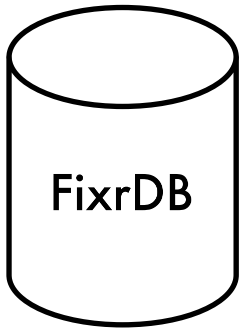



krishnachaitanyasripada — ubuntu@harvestr-vm: /media/data/git — ssh — 142×38

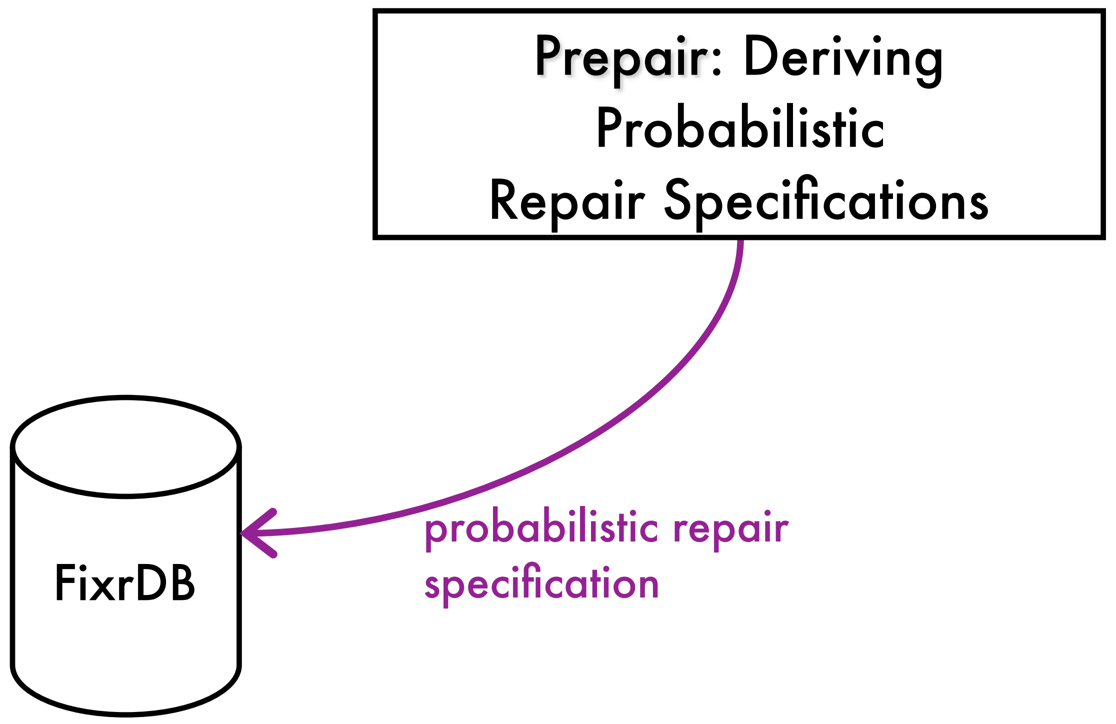
```
ubuntu@harvestr-vm:/media/data/git$ 
```

Workflow: Synthesizing patches

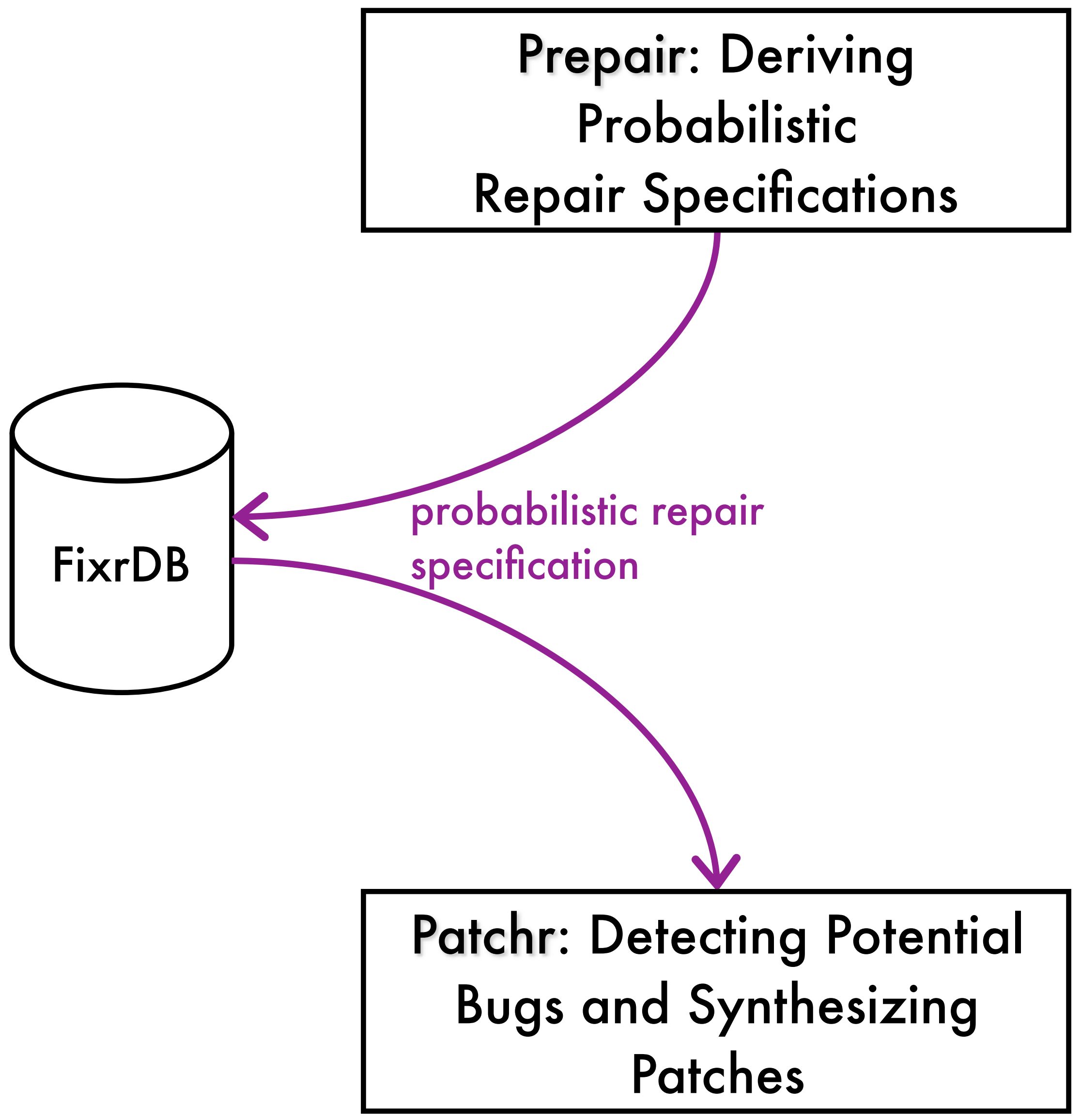
Prepair: Deriving
Probabilistic
Repair Specifications



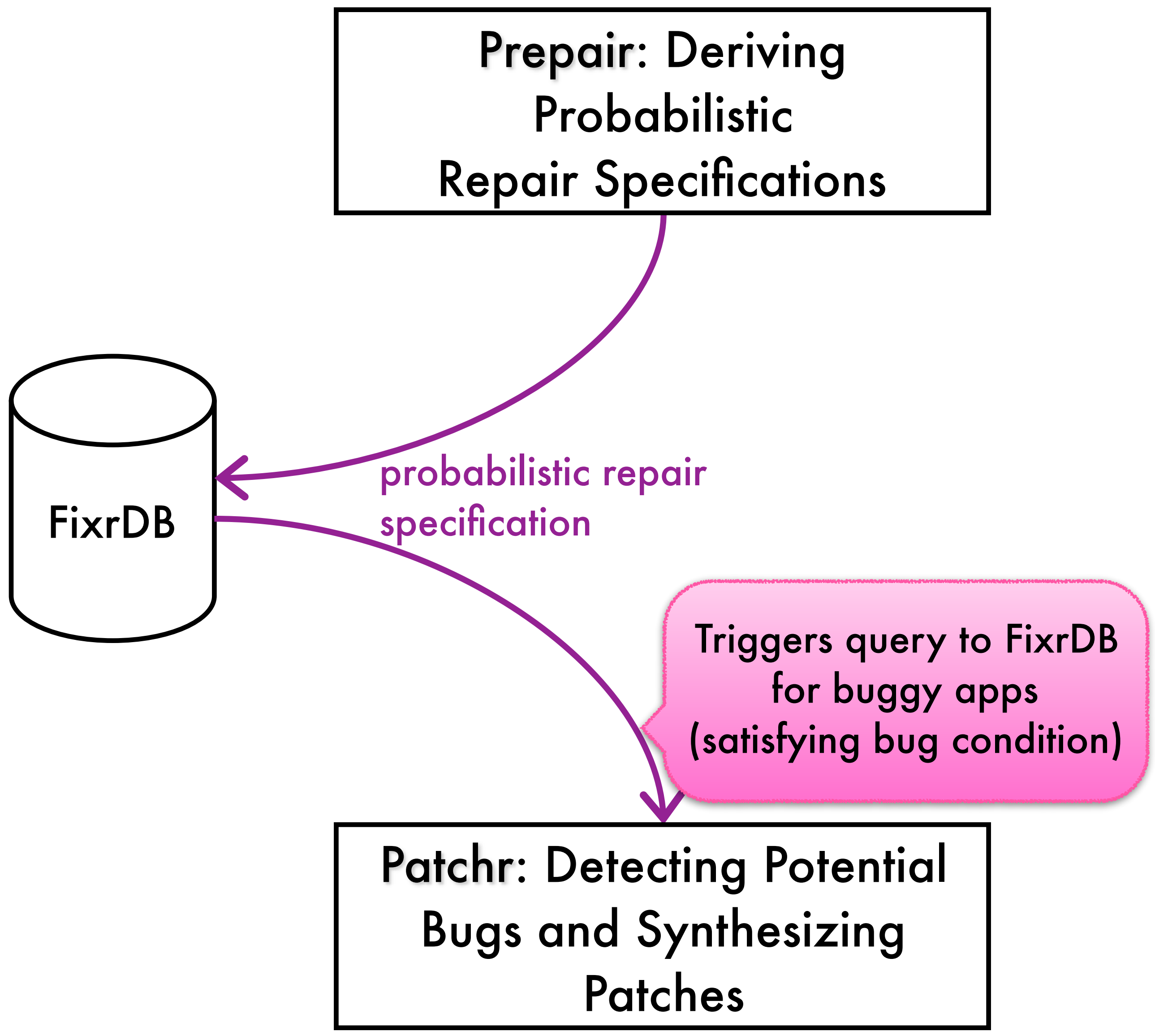
Workflow: Synthesizing patches


Prepair: Deriving Probabilistic Repair Specifications
probabilistic repair specification
FixrDB

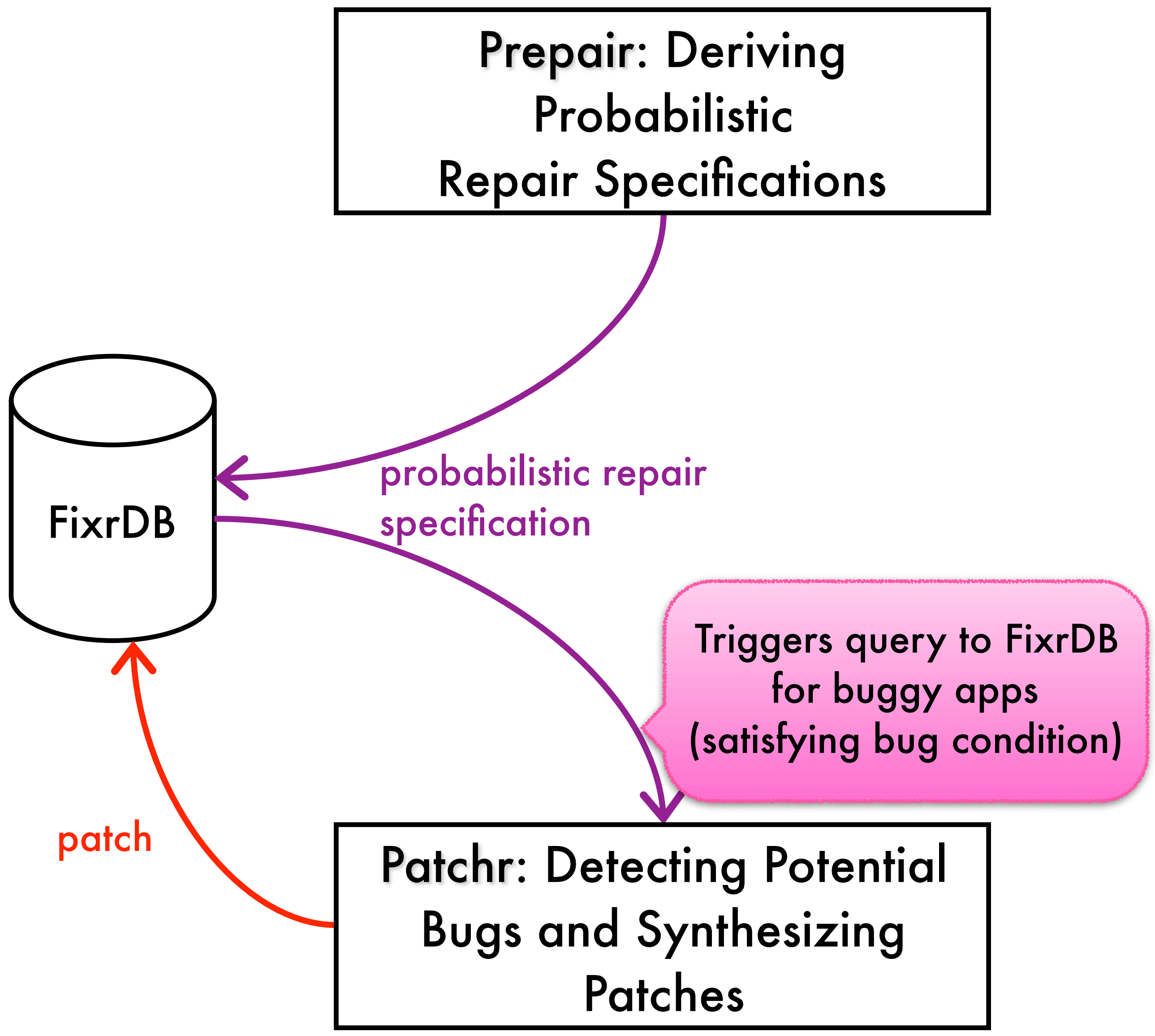
Workflow: Synthesizing patches


Prepair: Deriving Probabilistic Repair Specifications
FixrDB
probabilistic repair specification
Patchr: Detecting Potential Bugs and Synthesizing Patches

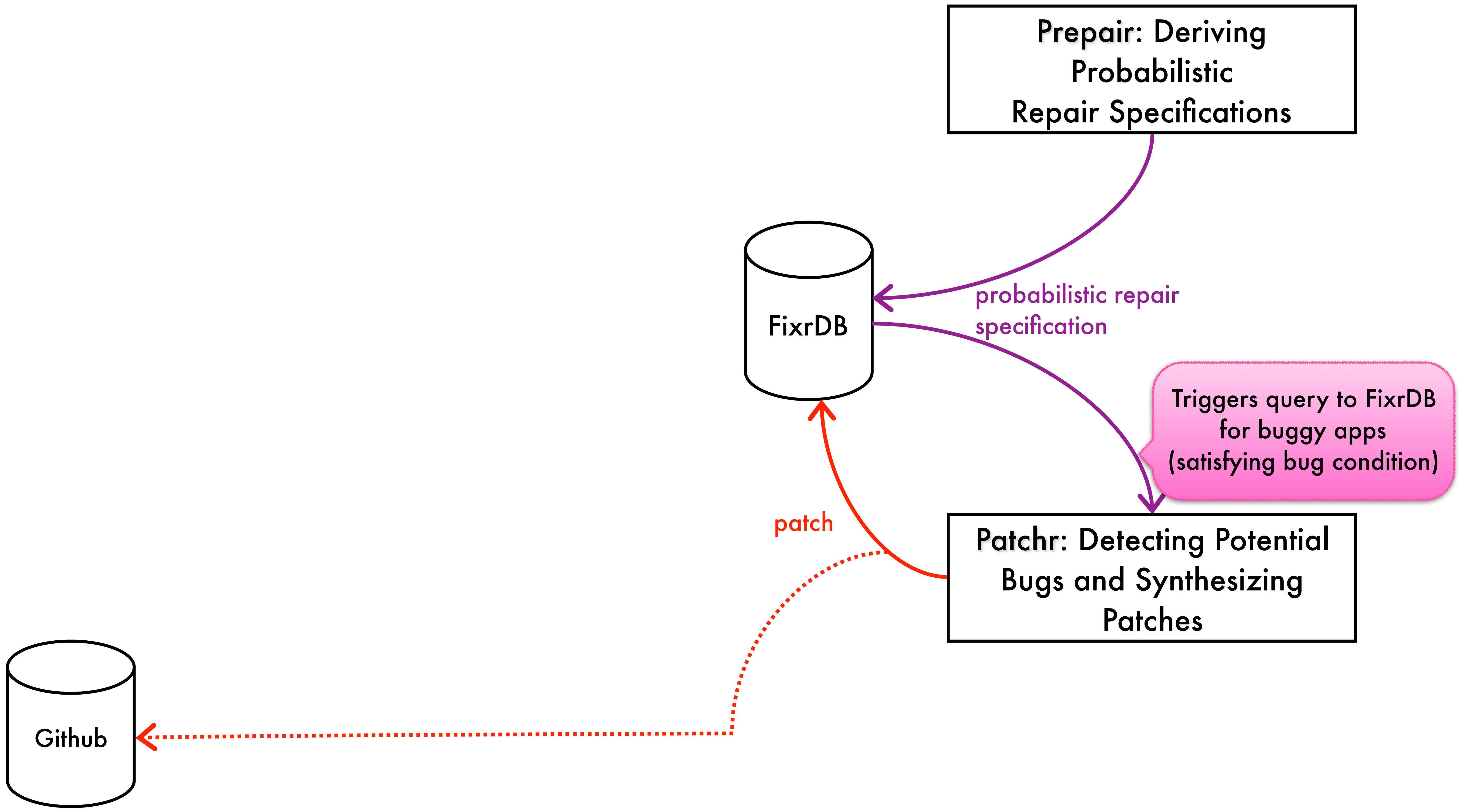
Workflow: Synthesizing patches


Prepair: Deriving Probabilistic Repair Specifications
FixrDB
probabilistic repair specification
Triggers query to FixrDB for buggy apps (satisfying bug condition)
Patchr: Detecting Potential Bugs and Synthesizing Patches

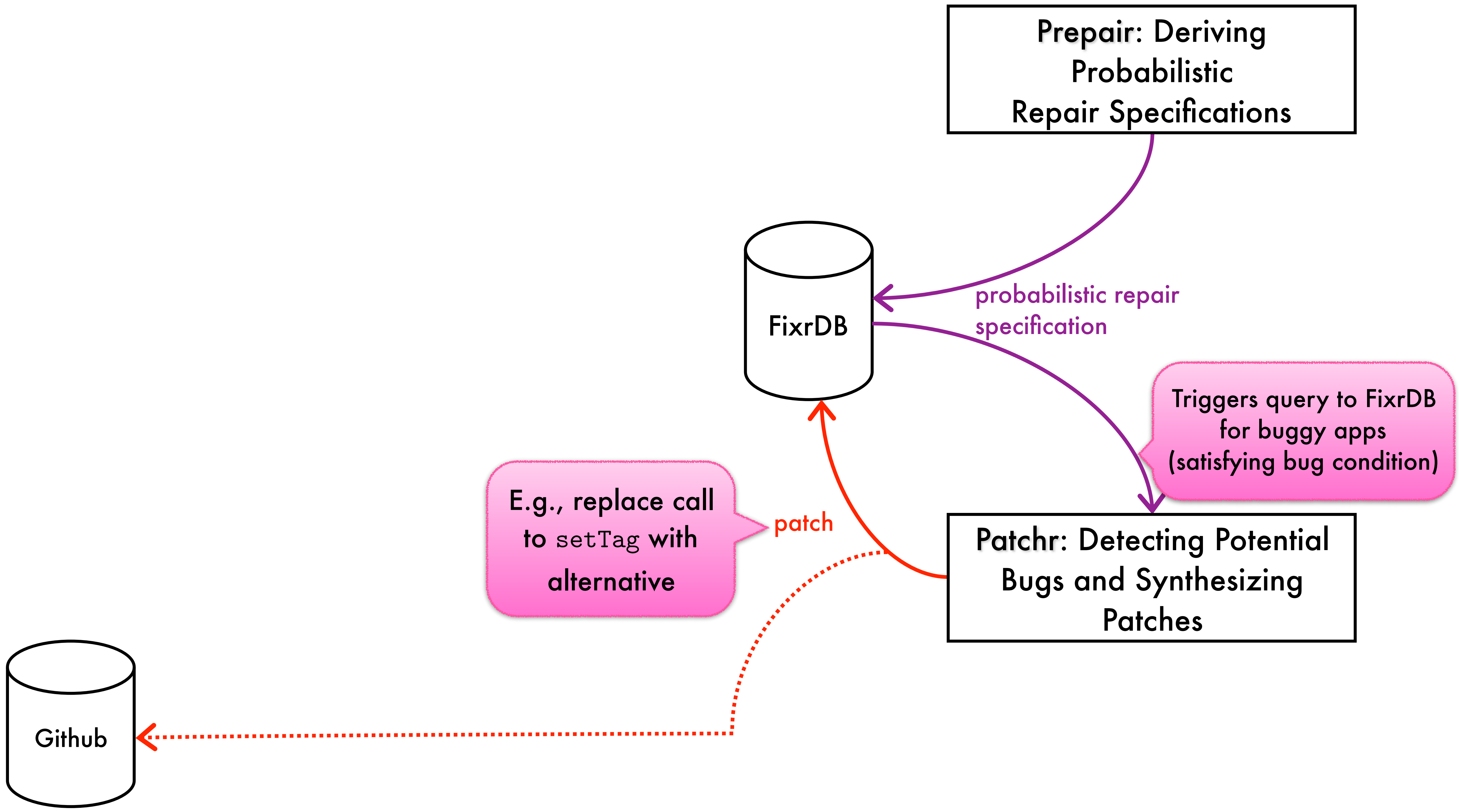
Workflow: Synthesizing patches


Prepair: Deriving Probabilistic Repair Specifications
FixrDB
probabilistic repair specification
Triggers query to FixrDB for buggy apps (satisfying bug condition)
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches

Workflow: Synthesizing patches

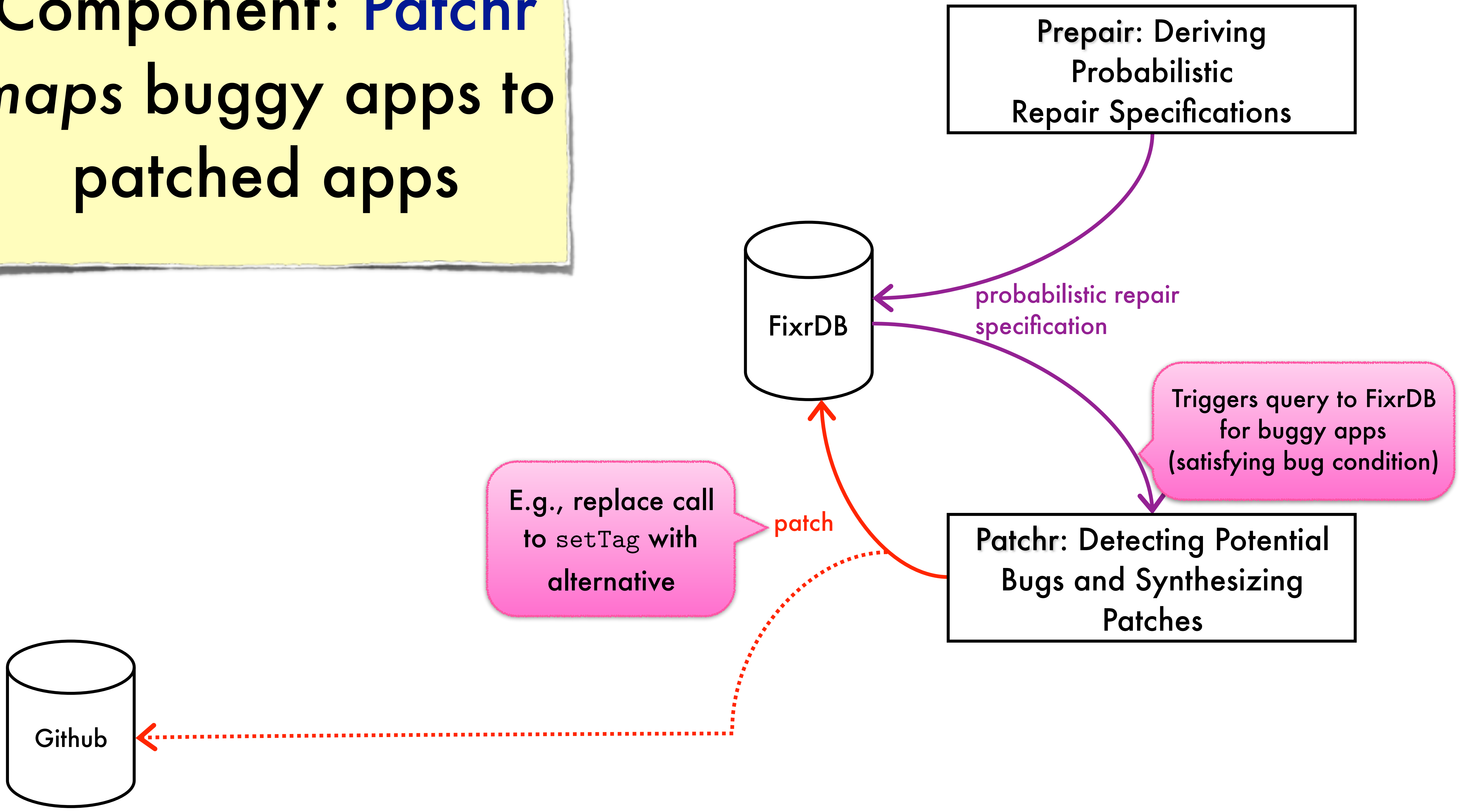

Prepair: Deriving Probabilistic Repair Specifications
probabilistic repair specification
FixrDB
Triggers query to FixrDB for buggy apps (satisfying bug condition)
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
Github

Workflow: Synthesizing patches


Prepair: Deriving Probabilistic Repair Specifications
probabilistic repair specification
FixrDB
Triggers query to FixrDB for buggy apps (satisfying bug condition)
Patchr: Detecting Potential Bugs and Synthesizing Patches
patch
E.g., replace call to setTag with alternative
Github

Workflow: Synthesizing patches

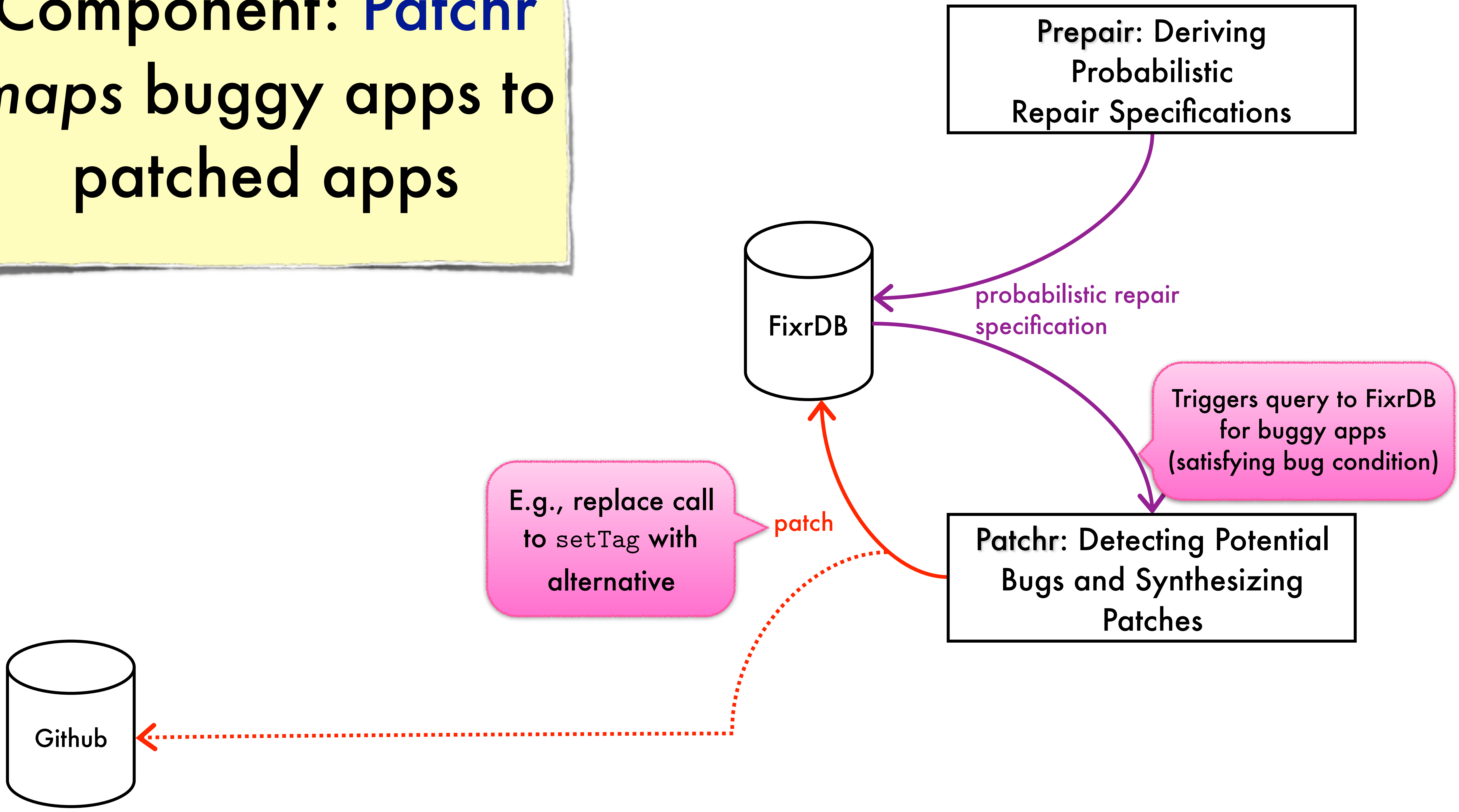
Component: Patchr *maps* buggy apps to patched apps



Prepair: Deriving Probabilistic Repair Specifications
FixrDB
probabilistic repair specification
Triggers query to FixrDB for buggy apps (satisfying bug condition)
Patchr: Detecting Potential Bugs and Synthesizing Patches
patch
E.g., replace call to setTag with alternative
Github

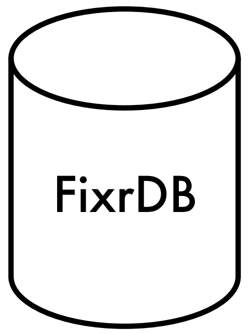
Workflow: Synthesizing patches

Component: Patchr *maps* buggy apps to patched apps


Prepair: Deriving Probabilistic Repair Specifications
FixrDB
probabilistic repair specification
Triggers query to FixrDB for buggy apps (satisfying bug condition)
Patchr: Detecting Potential Bugs and Synthesizing Patches
patch
E.g., replace call to setTag with alternative
Github

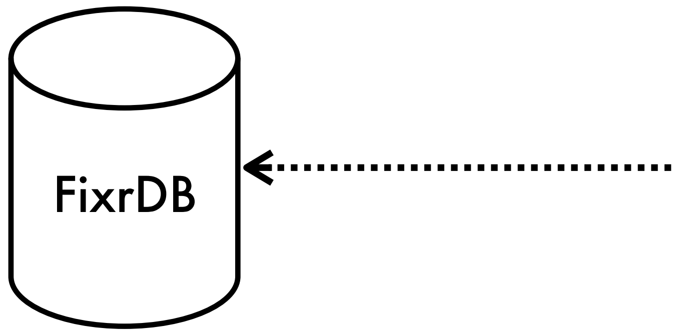
Query for buggy apps

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



Query for buggy apps

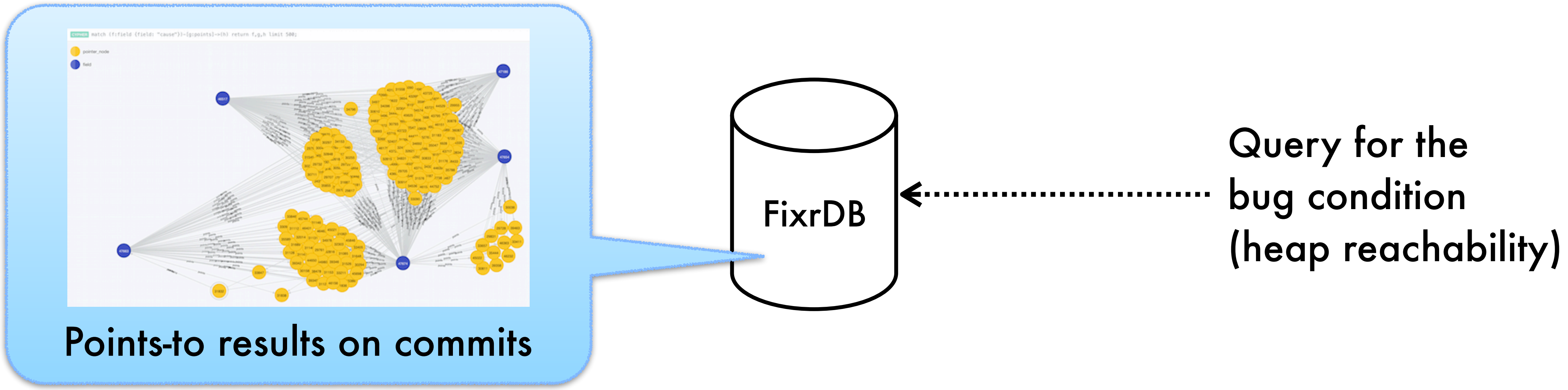
Triggers query to FixrDB for buggy apps (satisfying bug condition)



Query for the bug condition (heap reachability)

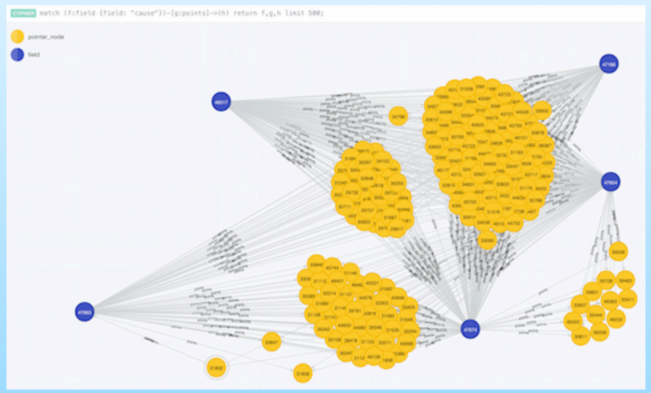
Query for buggy apps

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

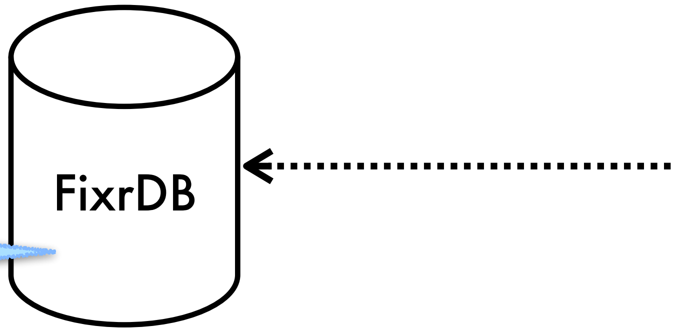

Points-to results on commits
FixrDB
Query for the bug condition (heap reachability)

Query for buggy apps

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



Points-to results on commits

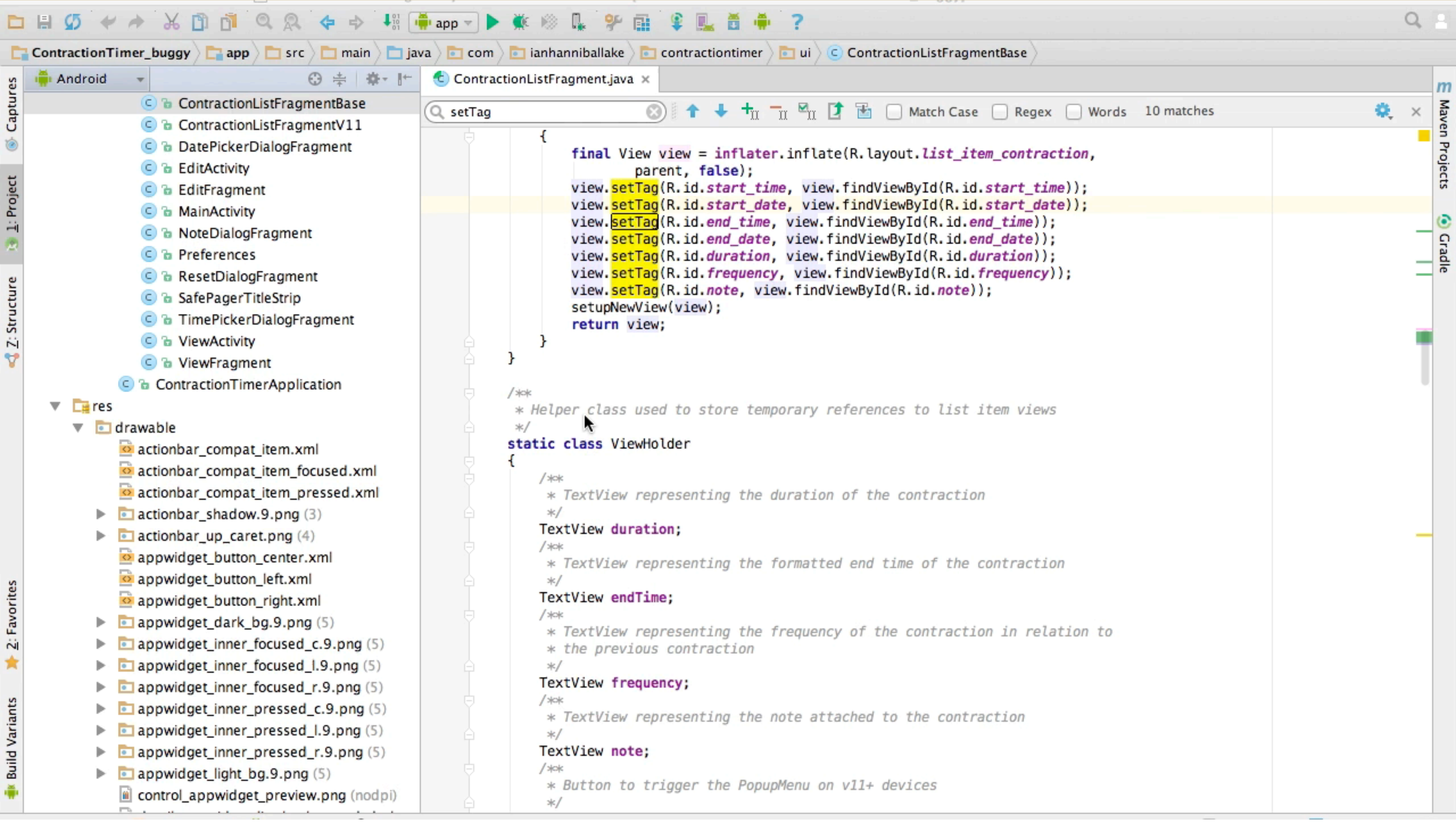


Query for the bug condition (heap reachability)

Currently, points-to results and program structure

Prototype querying for buggy apps

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



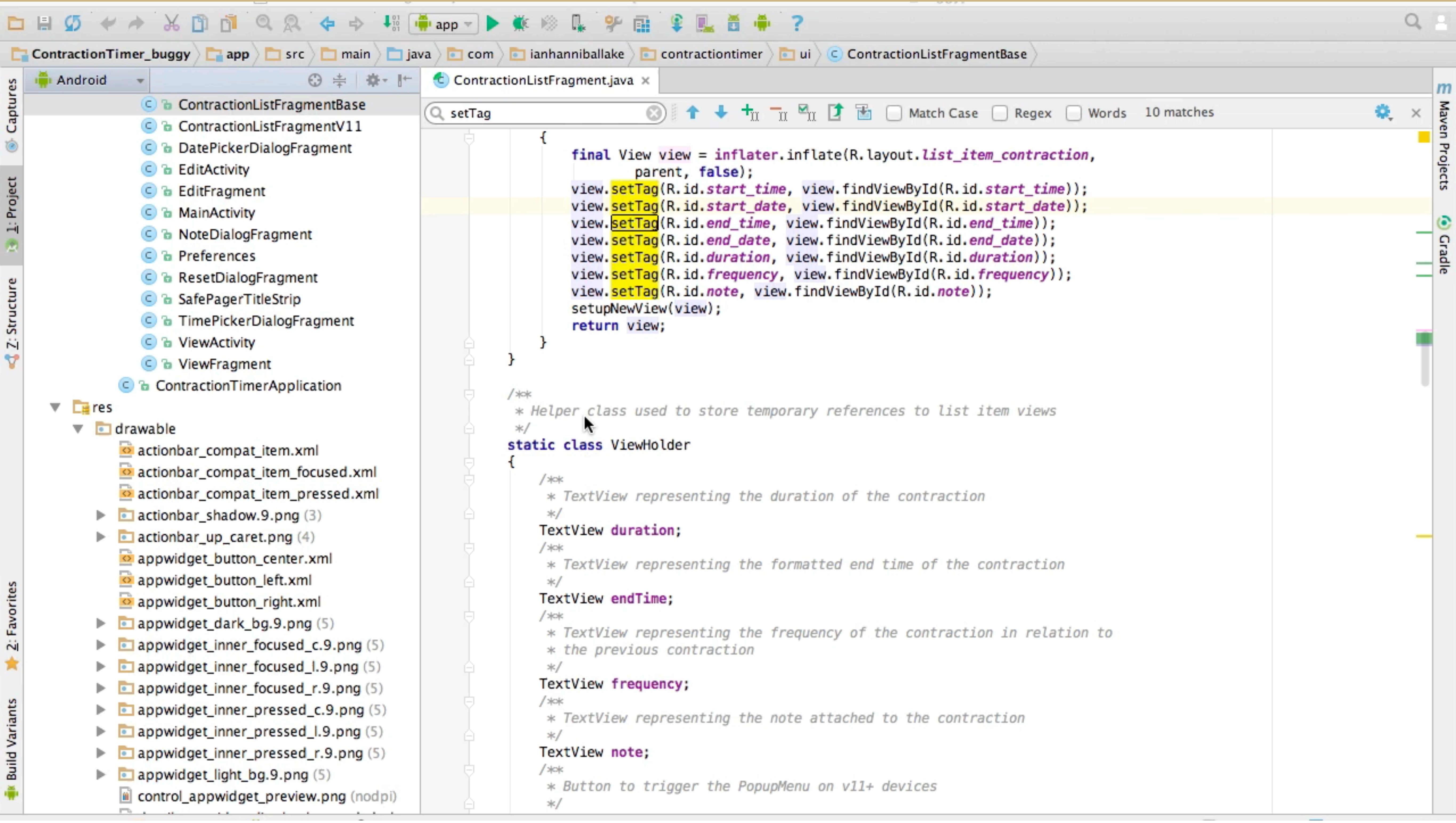
ContractionListFragmentBase
ContractionListFragmentV11
DatePickerDialogFragment
EditActivity
EditFragment
MainActivity
NoteDialogFragment
Preferences
ResetDialogFragment
SafePagerTitleStrip
TimePickerDialogFragment
ViewActivity
ViewFragment
ContractionTimerApplication
res
drawable
actionbar_compat_item.xml
actionbar_compat_item_focused.xml
actionbar_compat_item_pressed.xml
actionbar_shadow.9.png (3)
actionbar_up_caret.png (4)
appwidget_button_center.xml
appwidget_button_left.xml
appwidget_button_right.xml
appwidget_dark_bg.9.png (5)
appwidget_inner_focused_c.9.png (5)
appwidget_inner_focused_l.9.png (5)
appwidget_inner_focused_r.9.png (5)
appwidget_inner_pressed_c.9.png (5)
appwidget_inner_pressed_l.9.png (5)
appwidget_inner_pressed_r.9.png (5)
appwidget_light_bg.9.png (5)
control_appwidget_preview.png (nodpi)

```
        {
            final View view = inflater.inflate(R.layout.list_item_contraction,
                    parent, false);
            view.setTag(R.id.start_time, view.findViewById(R.id.start_time));
            view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
            view.setTag(R.id.end_time, view.findViewById(R.id.end_time));
            view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
            view.setTag(R.id.duration, view.findViewById(R.id.duration));
            view.setTag(R.id.frequency, view.findViewById(R.id.frequency));
            view.setTag(R.id.note, view.findViewById(R.id.note));
            setupNewView(view);
            return view;
        }
    }

    /**
     * Helper class used to store temporary references to list item views
     */
    static class ViewHolder
    {
        /**
         * TextView representing the duration of the contraction
         */
        TextView duration;
        /**
         * TextView representing the formatted end time of the contraction
         */
        TextView endTime;
        /**
         * TextView representing the frequency of the contraction in relation to
         * the previous contraction
         */
        TextView frequency;
        /**
         * TextView representing the note attached to the contraction
         */
        TextView note;
        /**
         * Button to trigger the PopupMenu on v11+ devices
         */
```

Prototype querying for buggy apps

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



ContractionTimer_buggy › app › src › main › java › com › ianhanniballake › contractiontimer › ui › ContractionListFragmentBase

Android

- ContractionListFragmentBase
- ContractionListFragmentV11
- DatePickerDialogFragment
- EditActivity
- EditFragment
- MainActivity
- NoteDialogFragment
- Preferences
- ResetDialogFragment
- SafePagerTitleStrip
- TimePickerDialogFragment
- ViewActivity
- ViewFragment
- ContractionTimerApplication
- res
  - drawable
    - actionbar_compat_item.xml
    - actionbar_compat_item_focused.xml
    - actionbar_compat_item_pressed.xml
    - actionbar_shadow.9.png (3)
    - actionbar_up_caret.png (4)
    - appwidget_button_center.xml
    - appwidget_button_left.xml
    - appwidget_button_right.xml
    - appwidget_dark_bg.9.png (5)
    - appwidget_inner_focused_c.9.png (5)
    - appwidget_inner_focused_l.9.png (5)
    - appwidget_inner_focused_r.9.png (5)
    - appwidget_inner_pressed_c.9.png (5)
    - appwidget_inner_pressed_l.9.png (5)
    - appwidget_inner_pressed_r.9.png (5)
    - appwidget_light_bg.9.png (5)
    - control_appwidget_preview.png (nodpi)

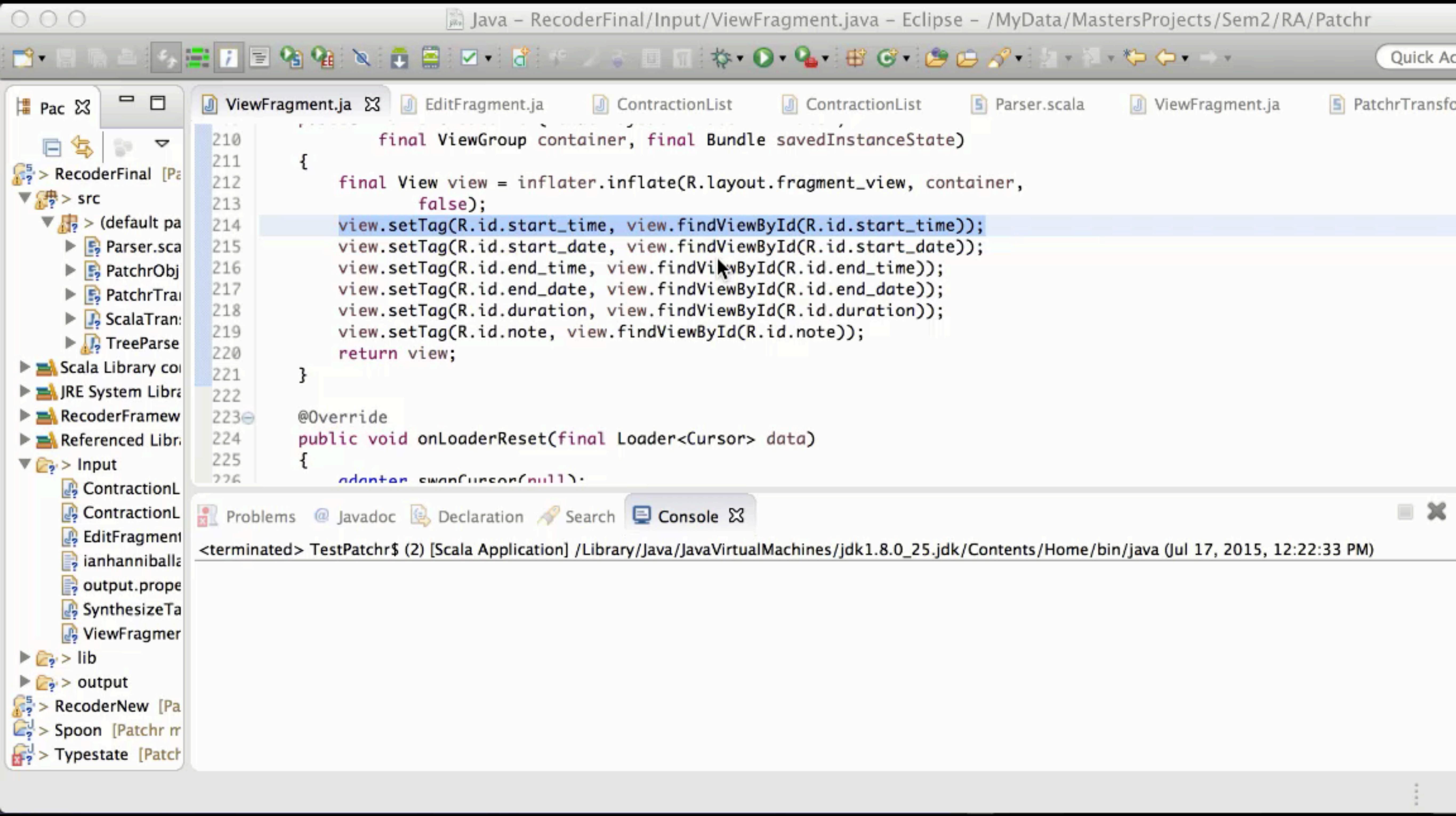
ContractionListFragment.java

setTag — Match Case — Regex — Words — 10 matches

```java
        {
            final View view = inflater.inflate(R.layout.list_item_contraction,
                    parent, false);
            view.setTag(R.id.start_time, view.findViewById(R.id.start_time));
            view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
            view.setTag(R.id.end_time, view.findViewById(R.id.end_time));
            view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
            view.setTag(R.id.duration, view.findViewById(R.id.duration));
            view.setTag(R.id.frequency, view.findViewById(R.id.frequency));
            view.setTag(R.id.note, view.findViewById(R.id.note));
            setupNewView(view);
            return view;
        }
    }

    /**
     * Helper class used to store temporary references to list item views
     */
    static class ViewHolder
    {
        /**
         * TextView representing the duration of the contraction
         */
        TextView duration;
        /**
         * TextView representing the formatted end time of the contraction
         */
        TextView endTime;
        /**
         * TextView representing the frequency of the contraction in relation to
         * the previous contraction
         */
        TextView frequency;
        /**
         * TextView representing the note attached to the contraction
         */
        TextView note;
        /**
         * Button to trigger the PopupMenu on v11+ devices
         */
```

Prototype patching



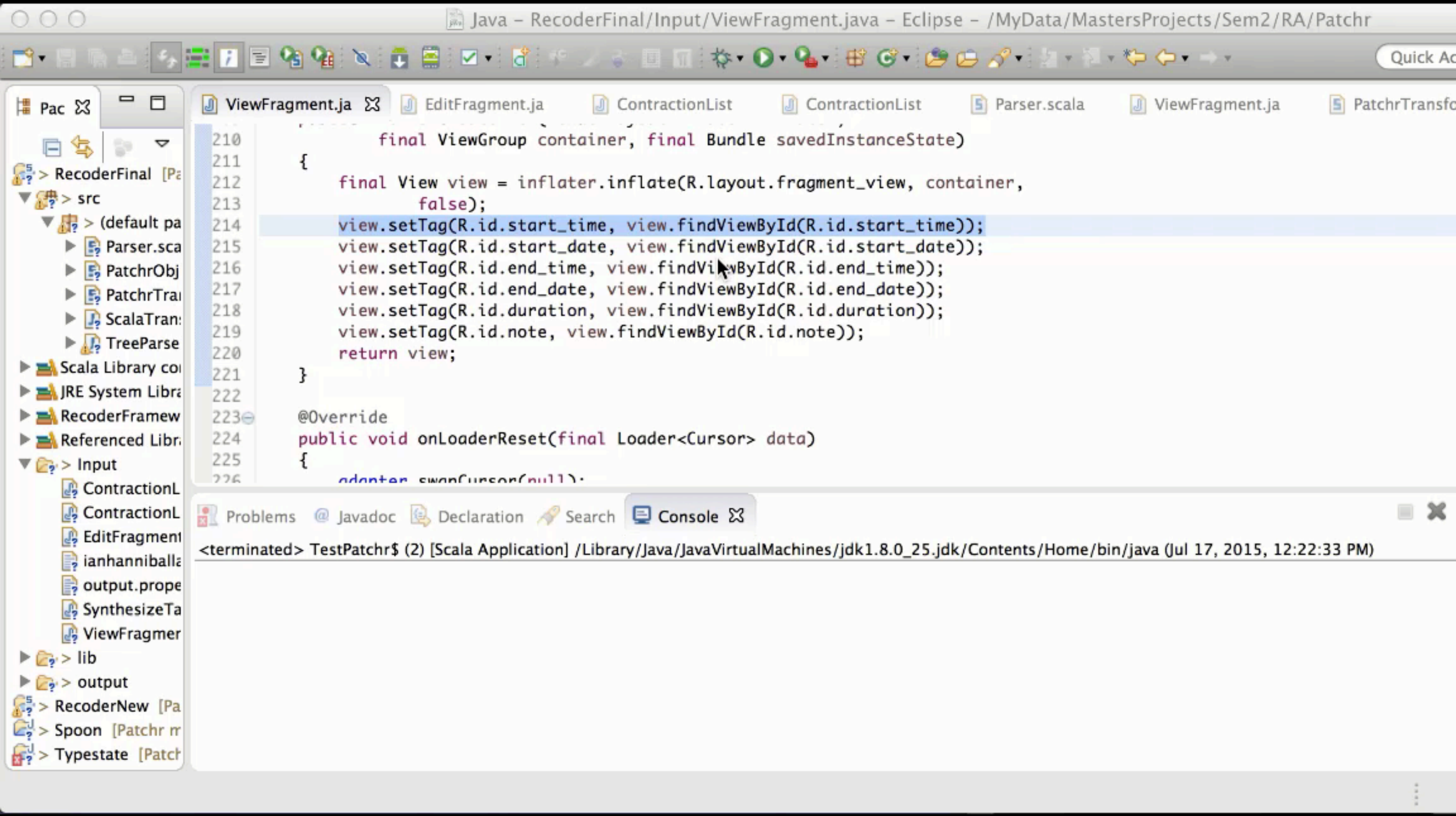
Java – RecoderFinal/Input/ViewFragment.java – Eclipse – /MyData/MastersProjects/Sem2/RA/Patchr

```
            final ViewGroup container, final Bundle savedInstanceState)
    {
        final View view = inflater.inflate(R.layout.fragment_view, container,
                false);
        view.setTag(R.id.start_time, view.findViewById(R.id.start_time));
        view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
        view.setTag(R.id.end_time, view.findViewById(R.id.end_time));
        view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
        view.setTag(R.id.duration, view.findViewById(R.id.duration));
        view.setTag(R.id.note, view.findViewById(R.id.note));
        return view;
    }

    @Override
    public void onLoaderReset(final Loader<Cursor> data)
    {
```

```
<terminated> TestPatchr$ (2) [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (Jul 17, 2015, 12:22:33 PM)
```

Prototype patching



```
            final ViewGroup container, final Bundle savedInstanceState)
    {
        final View view = inflater.inflate(R.layout.fragment_view, container,
                false);
        view.setTag(R.id.start_time, view.findViewById(R.id.start_time));
        view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
        view.setTag(R.id.end_time, view.findViewById(R.id.end_time));
        view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
        view.setTag(R.id.duration, view.findViewById(R.id.duration));
        view.setTag(R.id.note, view.findViewById(R.id.note));
        return view;
    }

    @Override
    public void onLoaderReset(final Loader<Cursor> data)
    {
```

```
<terminated> TestPatchr$ (2) [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (Jul 17, 2015, 12:22:33 PM)
```

Challenges

How do we find bugfix commits?

an instance of a View.setTag fix

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

a specification of the View.setTag repair

Challenges

How do we find

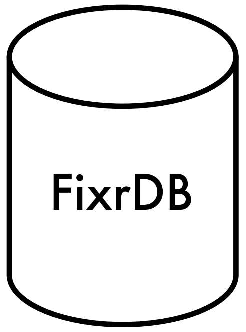
an instance of a View.setTag fix

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

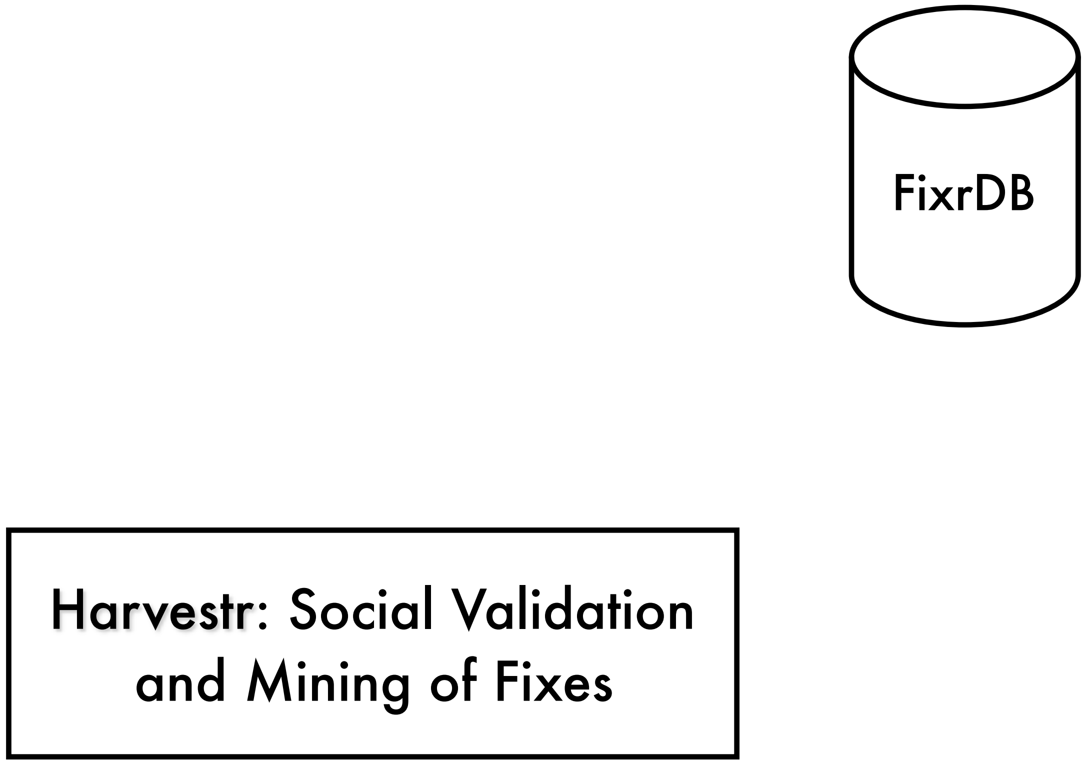
a specification of the View.setTag repair

Workflow: Processing bugfix commits

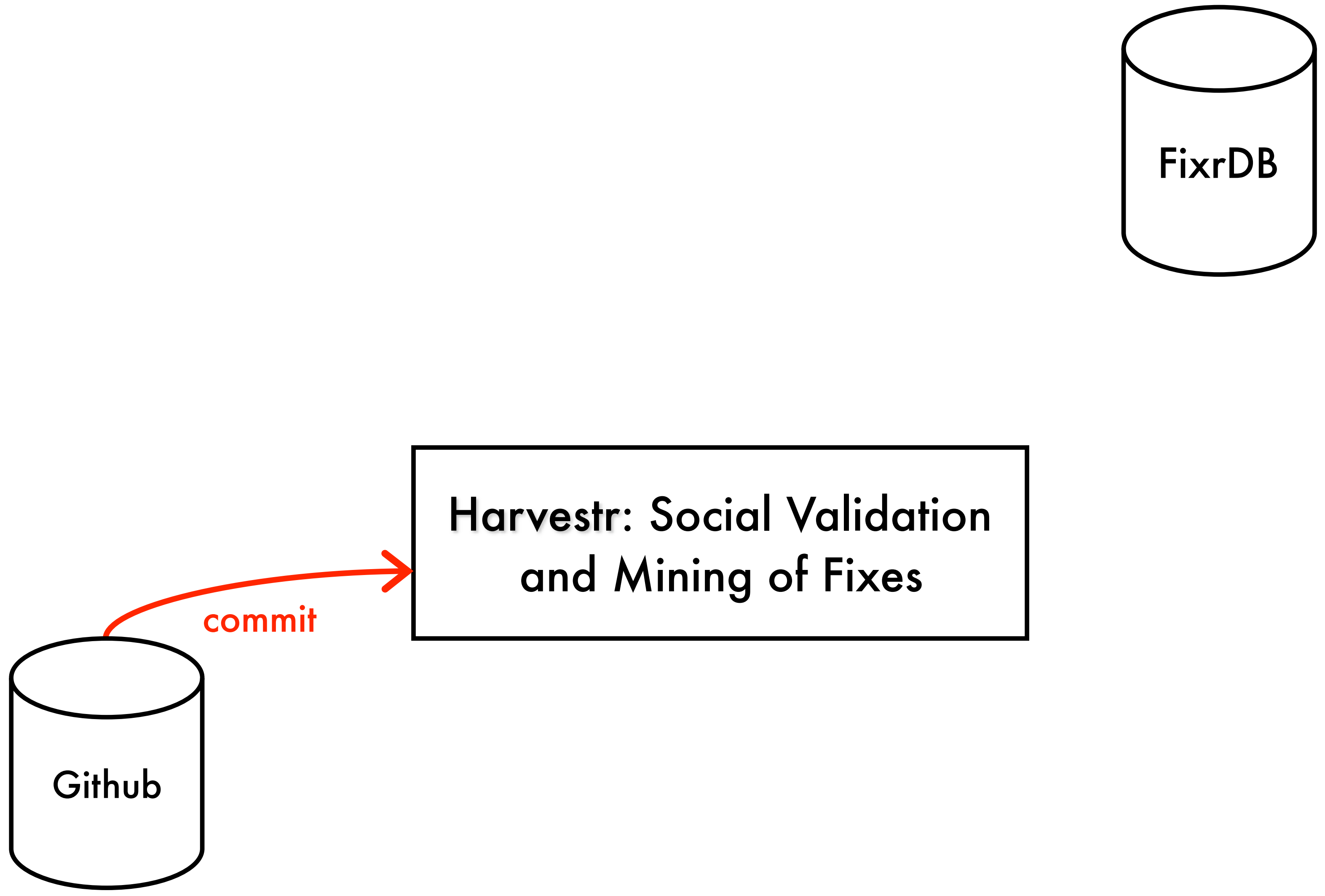
Workflow: Processing bugfix commits



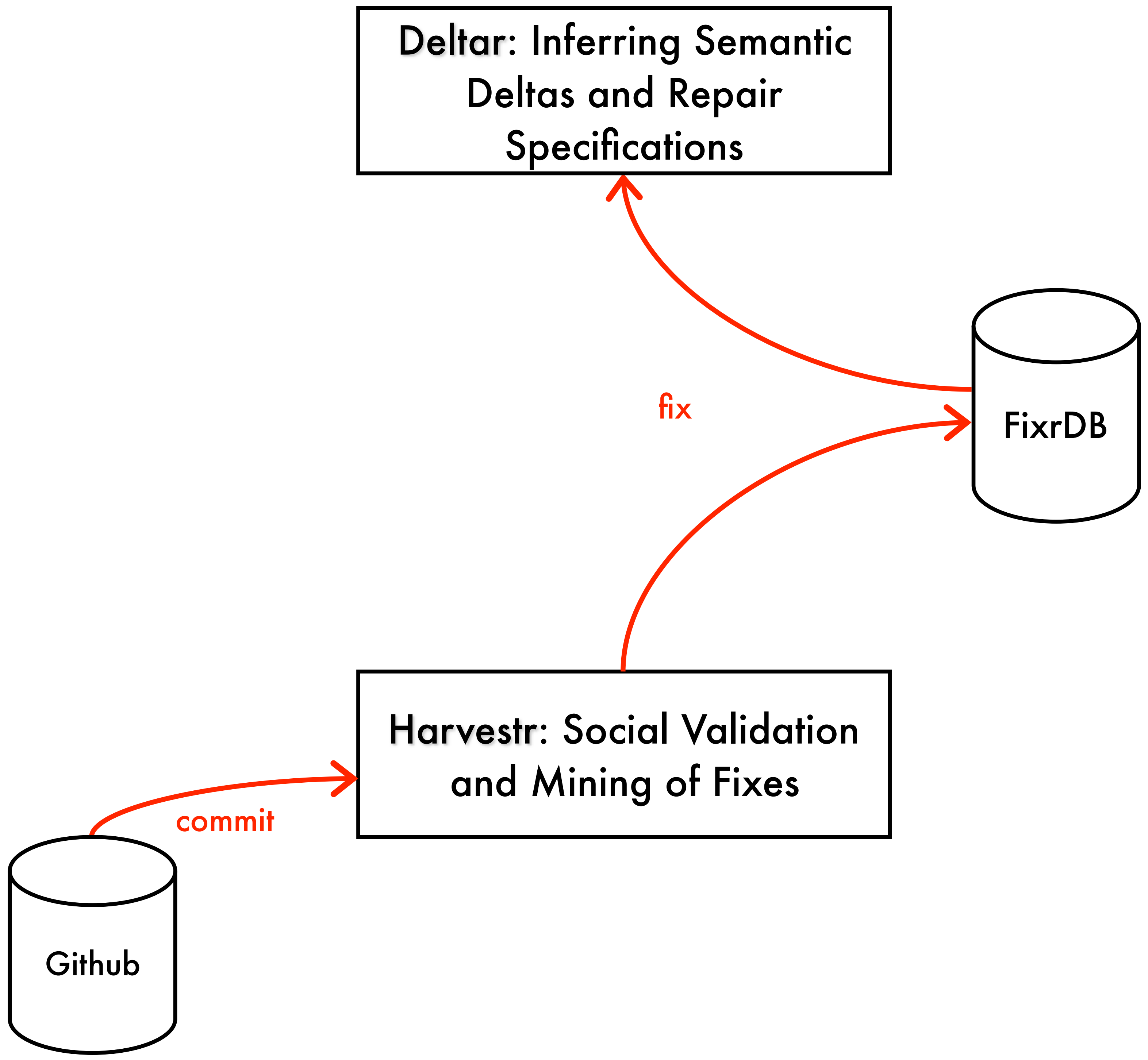
Workflow: Processing bugfix commits



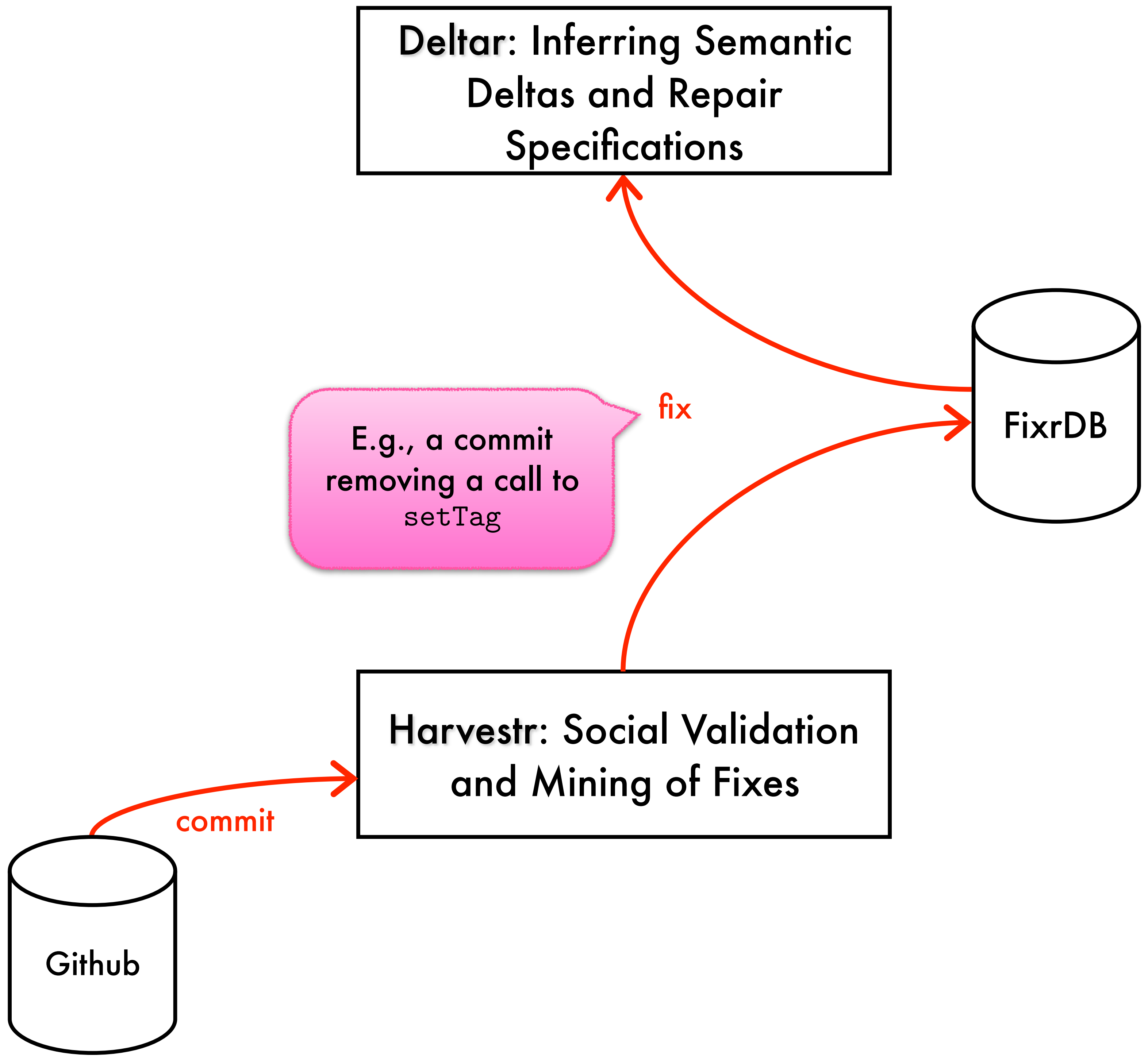
Workflow: Processing bugfix commits



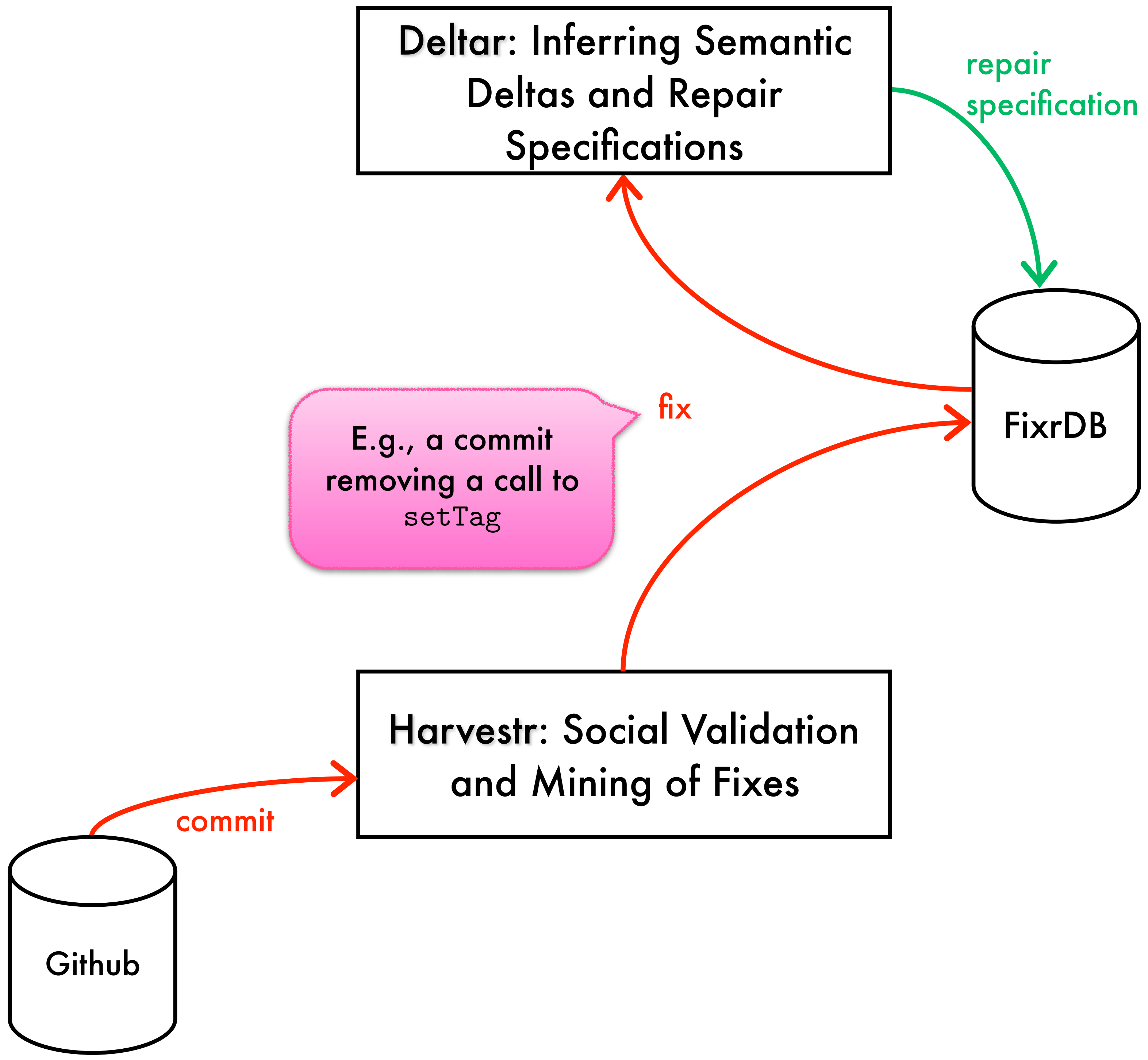
Workflow: Processing bugfix commits


Deltar: Inferring Semantic Deltas and Repair Specifications
fix
FixrDB
Harvestr: Social Validation and Mining of Fixes
commit
Github

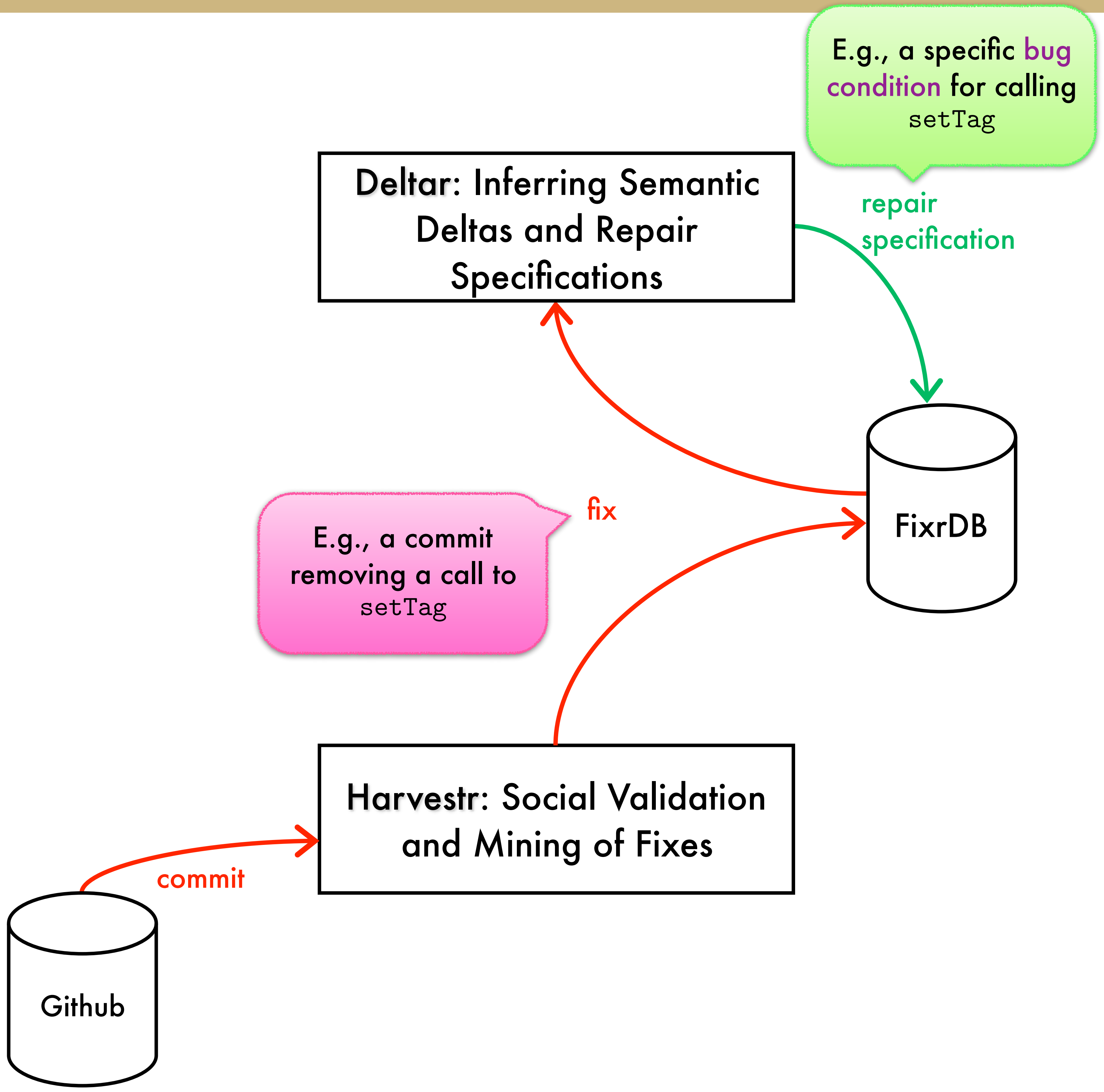
Workflow: Processing bugfix commits


Deltar: Inferring Semantic Deltas and Repair Specifications
FixrDB
fix
E.g., a commit removing a call to setTag
Harvestr: Social Validation and Mining of Fixes
commit
Github

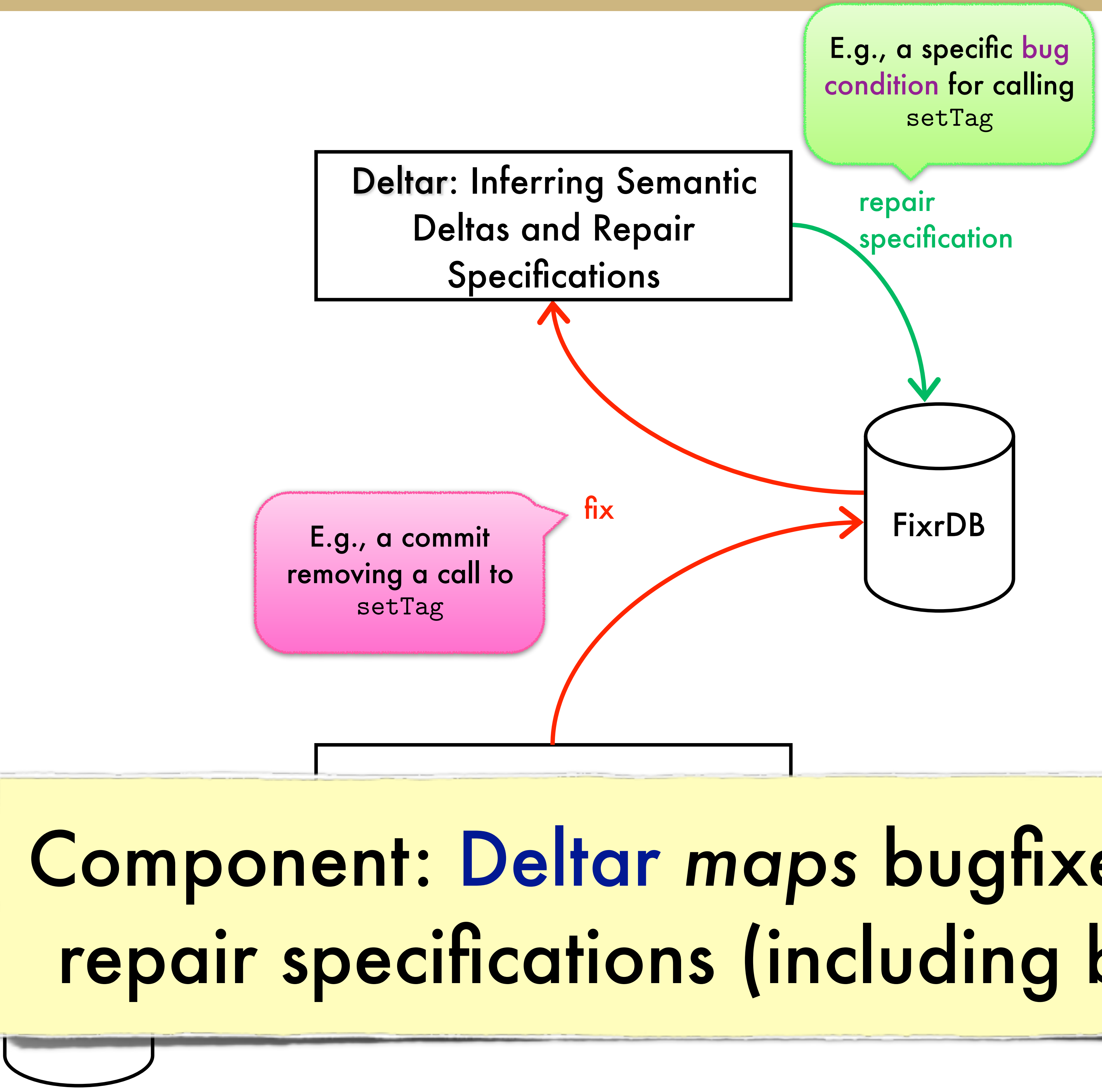
Workflow: Processing bugfix commits


Deltar: Inferring Semantic Deltas and Repair Specifications
repair specification
FixrDB
fix
E.g., a commit removing a call to setTag
Harvestr: Social Validation and Mining of Fixes
commit
Github

Workflow: Processing bugfix commits

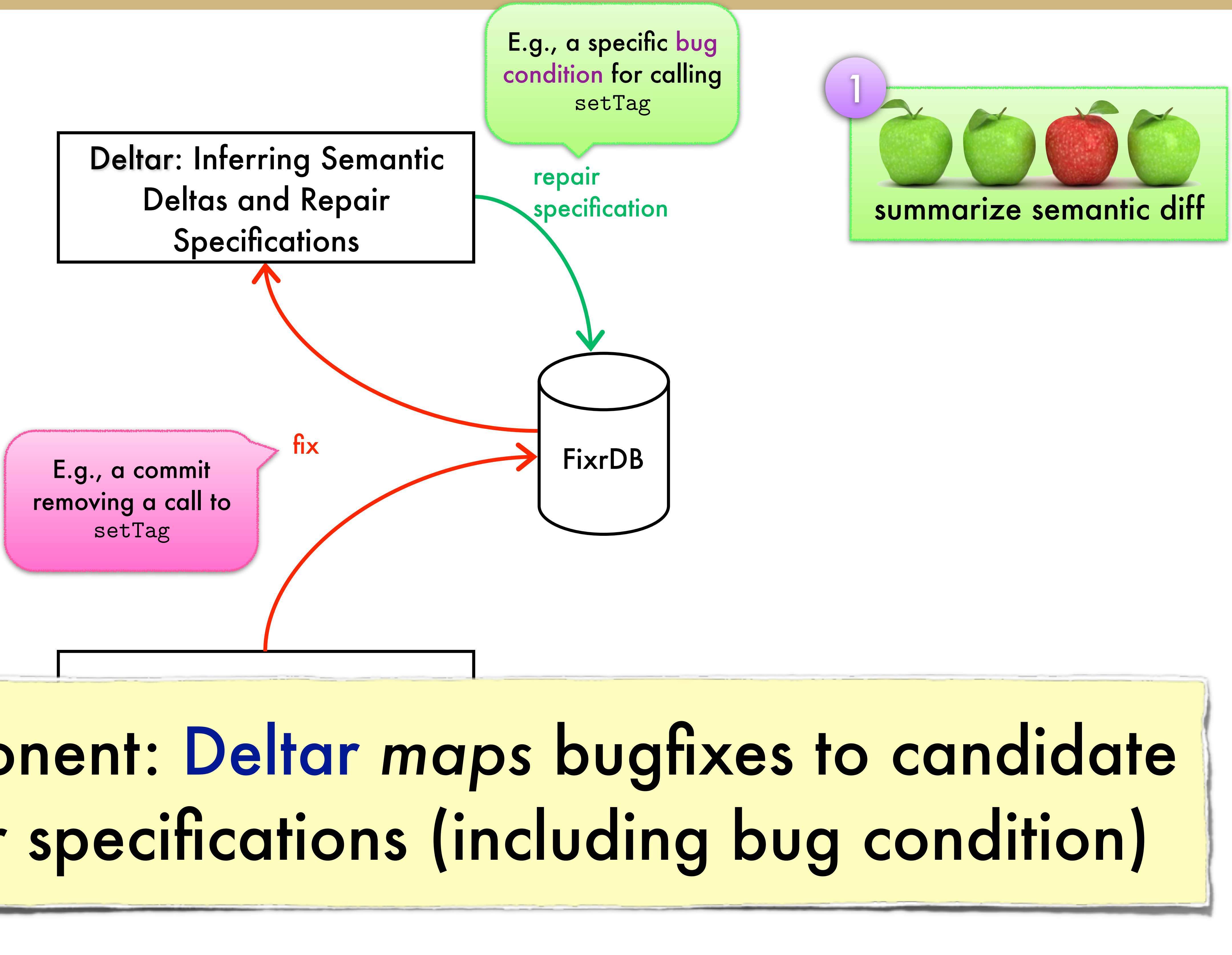

E.g., a specific bug condition for calling `setTag`
repair specification
Deltar: Inferring Semantic Deltas and Repair Specifications
FixrDB
fix
E.g., a commit removing a call to `setTag`
Harvestr: Social Validation and Mining of Fixes
commit
Github

Workflow: Processing bugfix commits


E.g., a specific bug condition for calling setTag
Deltar: Inferring Semantic Deltas and Repair Specifications
repair specification
FixrDB
fix
E.g., a commit removing a call to setTag

Component: Deltar *maps* bugfixes to candidate repair specifications (including bug condition)

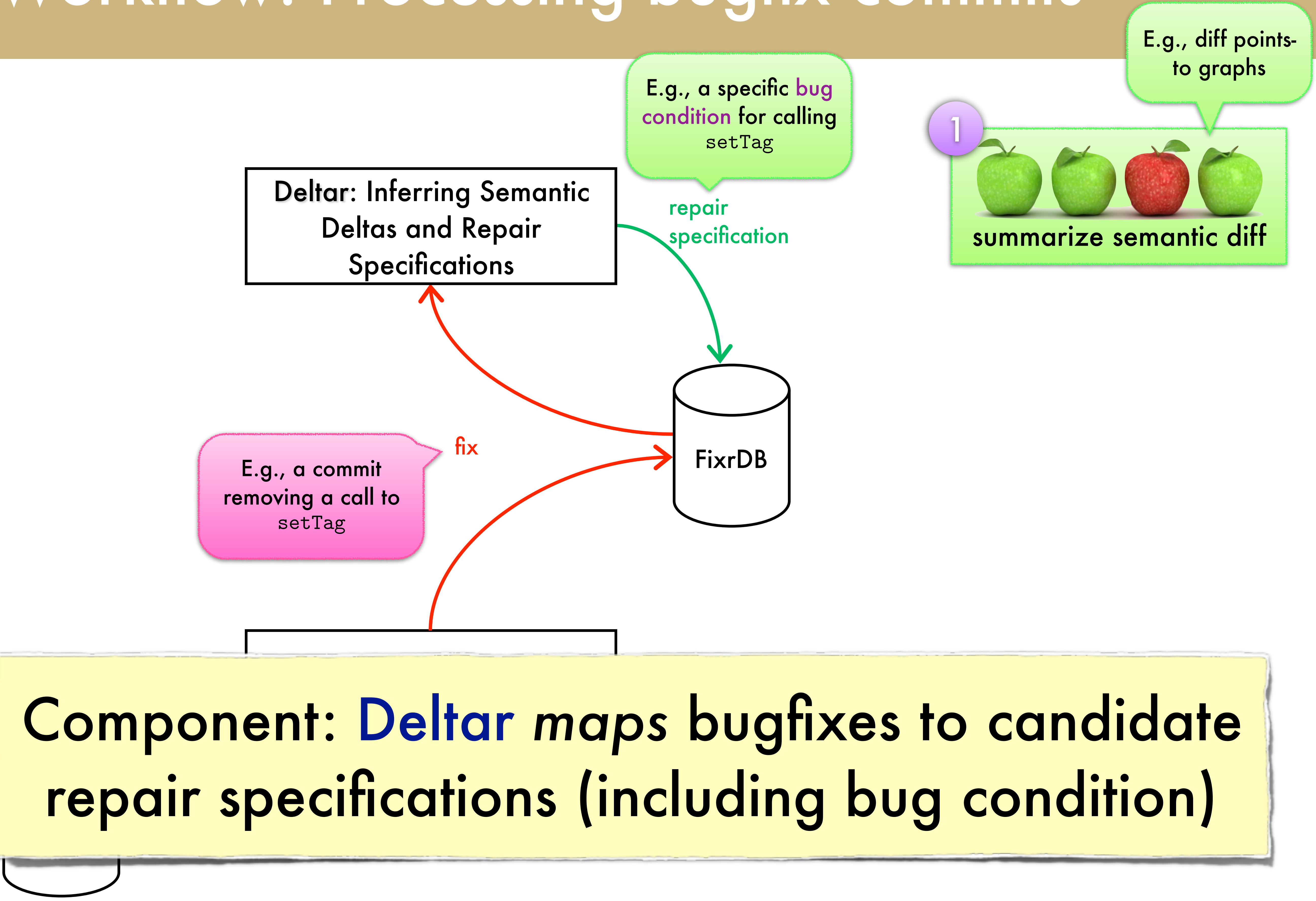
Workflow: Processing bugfix commits



E.g., a specific bug condition for calling setTag
1
summarize semantic diff
Deltar: Inferring Semantic Deltas and Repair Specifications
repair specification
FixrDB
fix
E.g., a commit removing a call to setTag

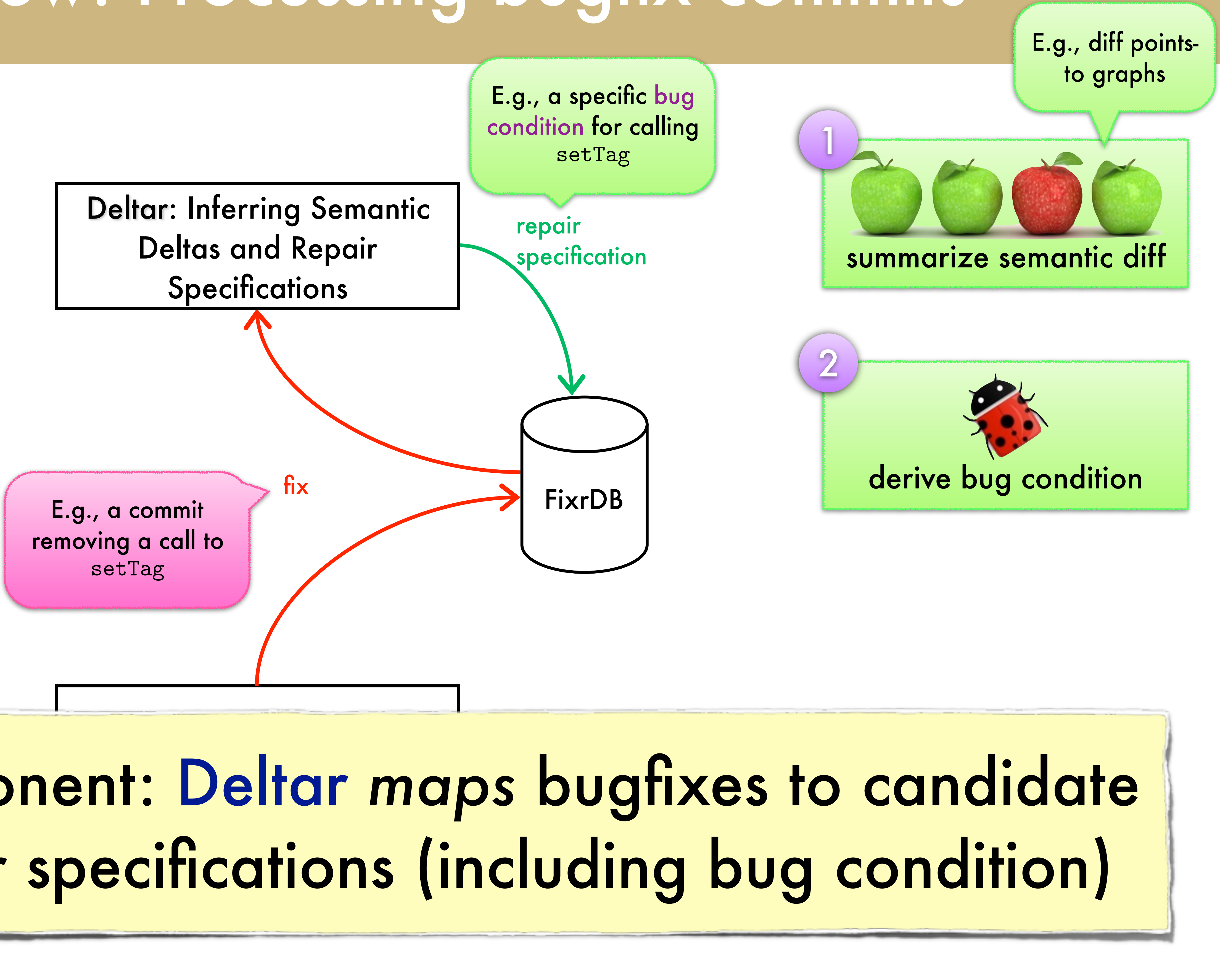
Component: Deltar *maps* bugfixes to candidate repair specifications (including bug condition)

Workflow: Processing bugfix commits


E.g., a specific bug condition for calling setTag
E.g., diff points-to graphs
1
summarize semantic diff
Deltar: Inferring Semantic Deltas and Repair Specifications
repair specification
FixrDB
fix
E.g., a commit removing a call to setTag

Component: Deltar *maps* bugfixes to candidate repair specifications (including bug condition)

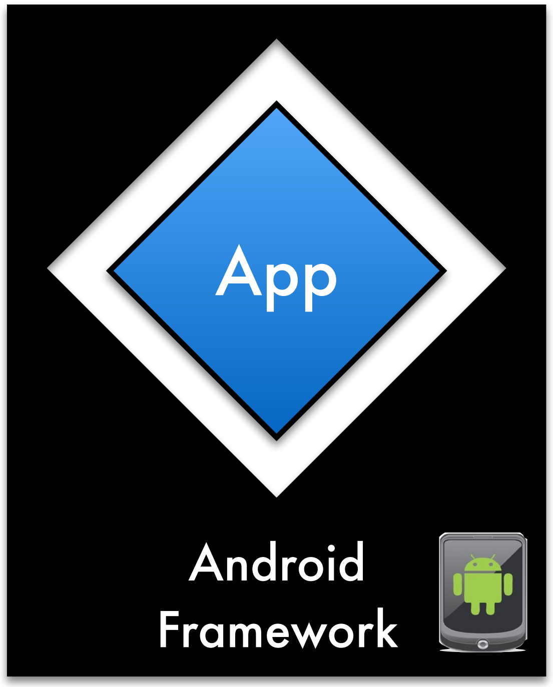
Workflow: Processing bugfix commits


Deltar: Inferring Semantic Deltas and Repair Specifications
E.g., a specific bug condition for calling setTag
repair specification
FixrDB
E.g., a commit removing a call to setTag
fix
E.g., diff points-to graphs
1
summarize semantic diff
2
derive bug condition

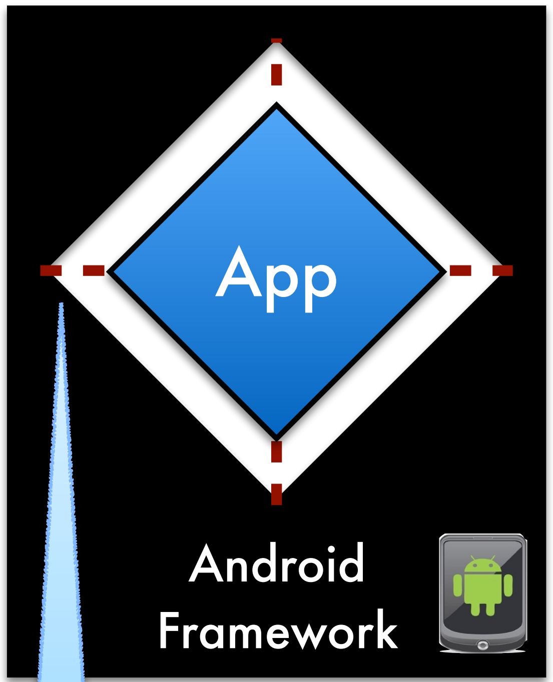
Component: Deltar *maps* bugfixes to candidate repair specifications (including bug condition)

Beyond memory properties …

Beyond memory properties …

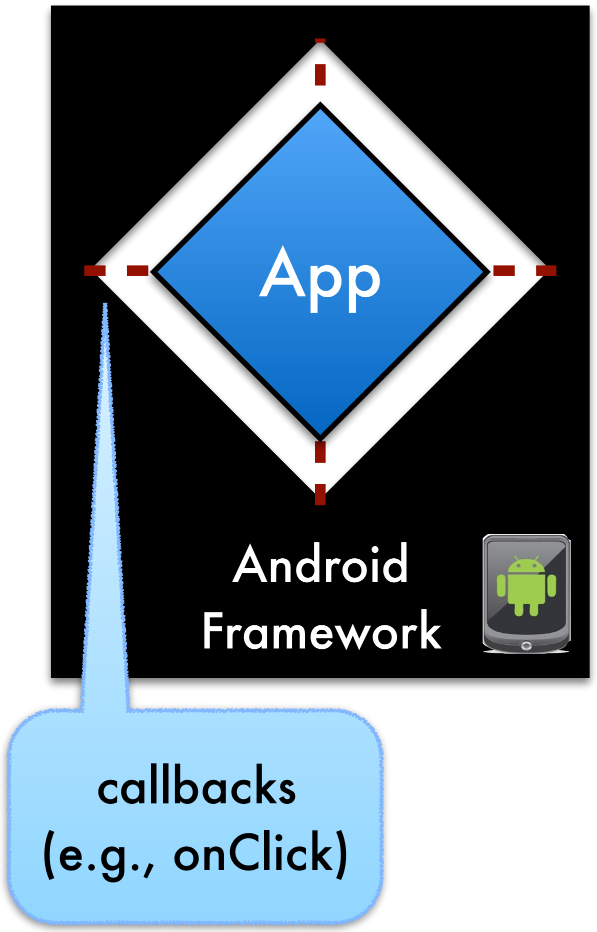


Beyond memory properties …



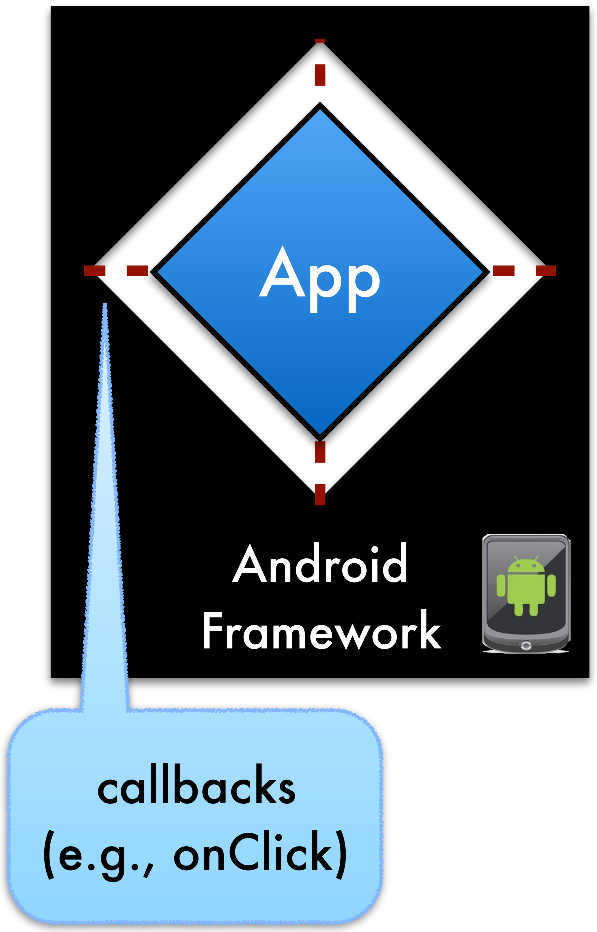
callbacks (e.g., onClick)

Beyond memory properties ...



```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp) {
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties ...

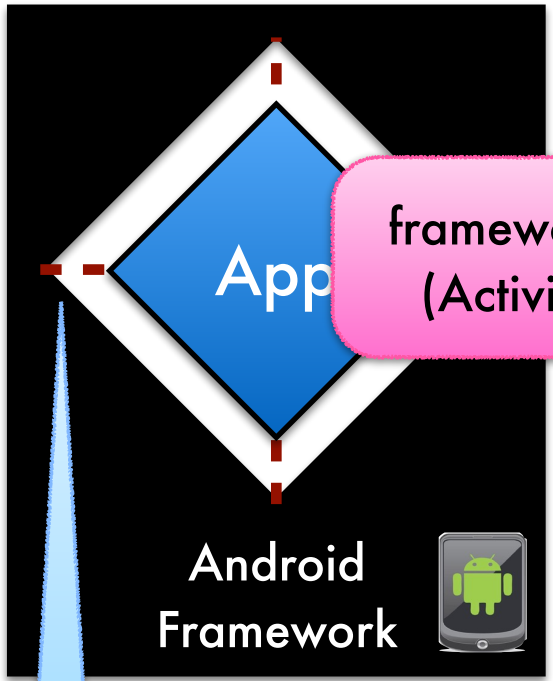


Lots of challenges ...

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp) {
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties ...

Lots of challenges ...

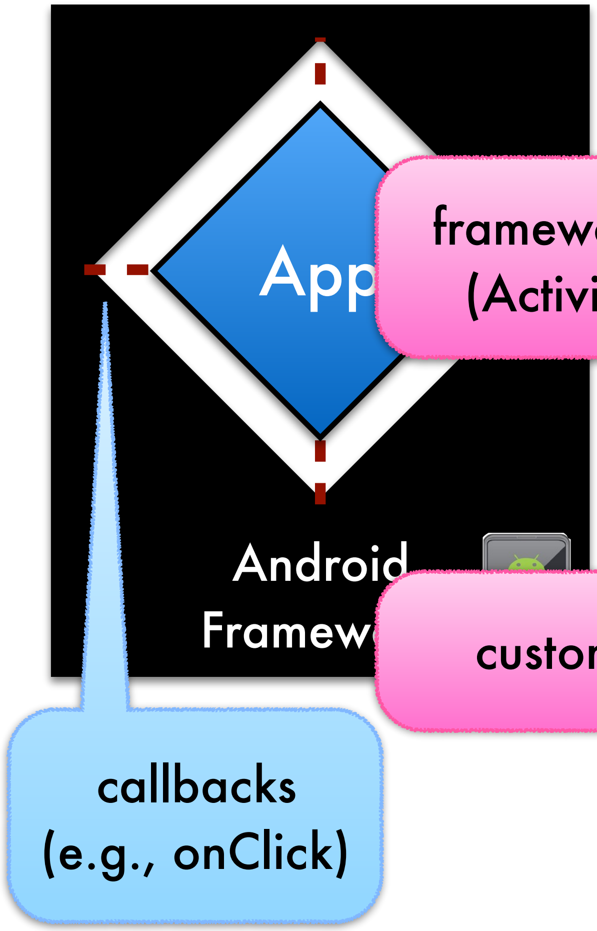


framework callbacks
(Activity lifecycle)

callbacks
(e.g., onClick)

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp) {
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties …



Lots of challenges …

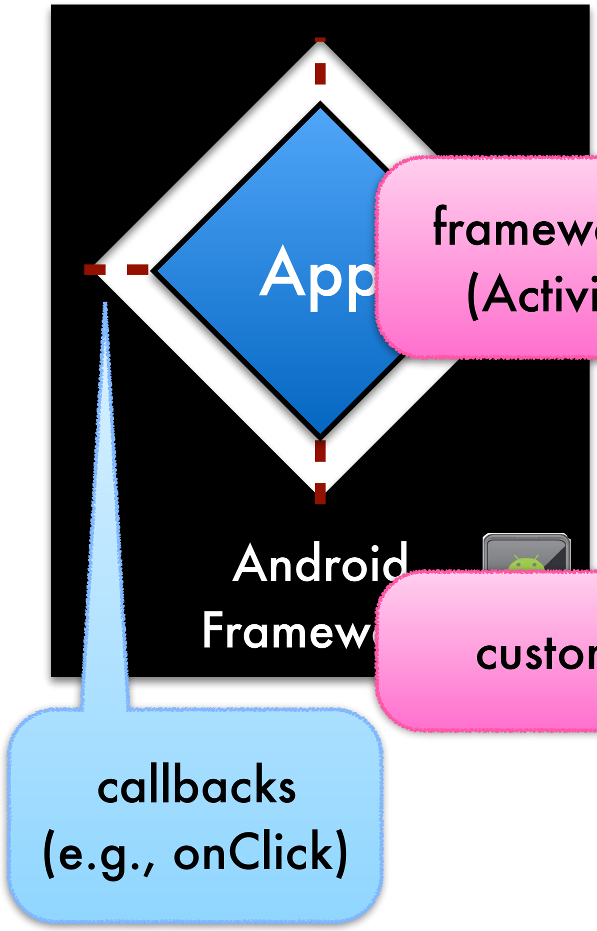
framework callbacks (Activity lifecycle)

custom callbacks

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp) {
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties ...

Lots of challenges ...



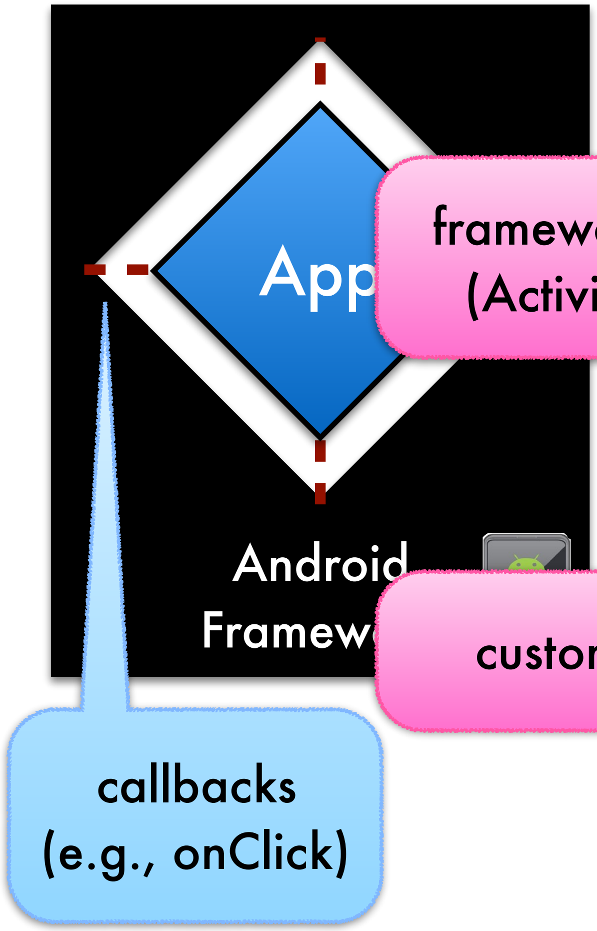
framework callbacks (Activity lifecycle)

triggering calls

custom callbacks

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp) {
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties ...



Lots of challenges ...

framework callbacks (Activity lifecycle)

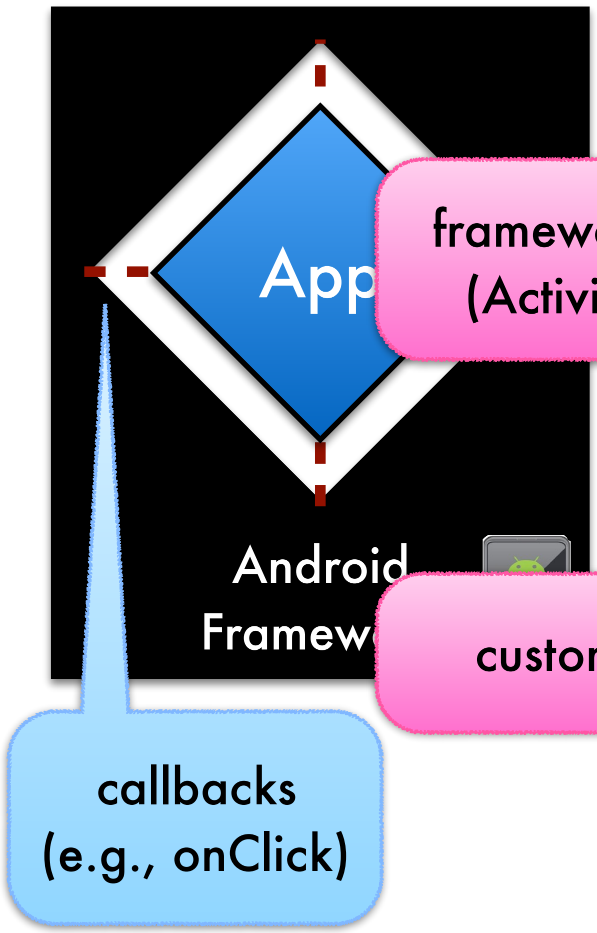
triggering calls

callback registration

custom callbacks

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp)
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties …



Lots of challenges …

framework callbacks (Activity lifecycle)

custom callbacks

triggering calls

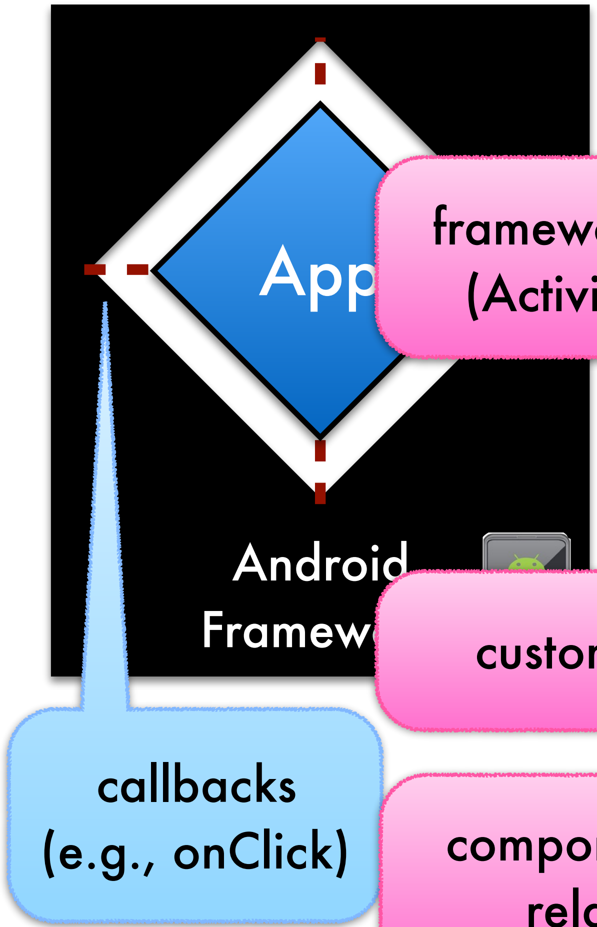
callback registration

enabling-disabling of callbacks

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp)
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

Beyond memory properties ...

Lots of challenges ...



framework callbacks (Activity lifecycle)

custom callbacks

component lifecycle relationships

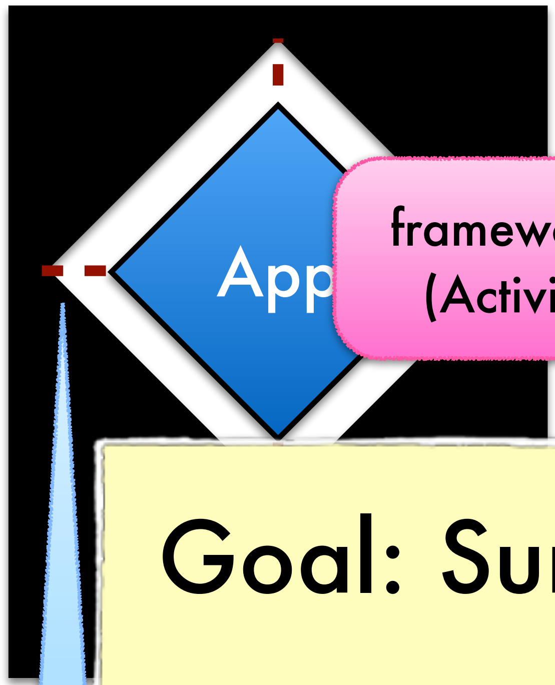
```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
  mediaPlayer.setOnPreparedListener(this);
}
void onPrepared(MediaPlayer mp)
  button.setEnabled(true);
}
void onClick(View v) {
  mediaPlayer.play();
}
```

triggering calls

callback registration

enabling-disabling of callbacks

Beyond memory properties …



Lots of challenges …

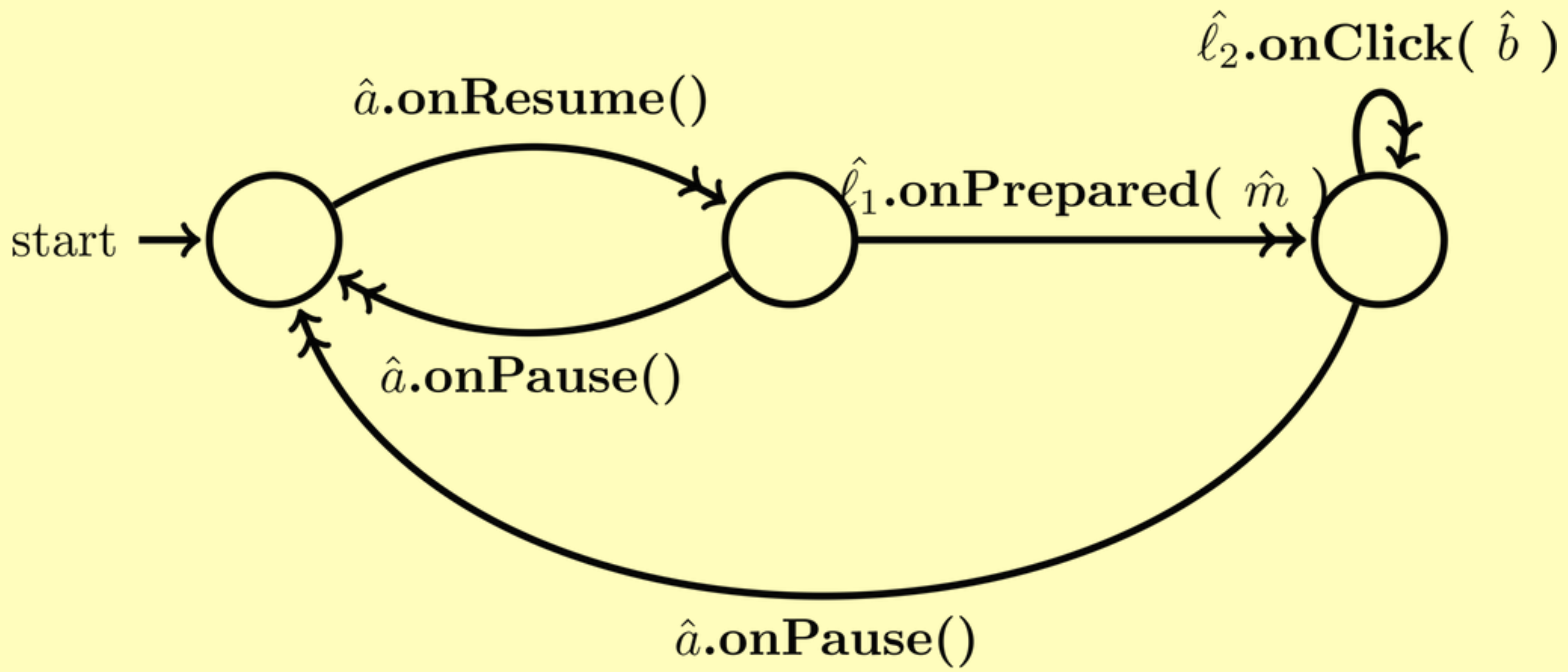
framework callbacks
(Activity lifecycle)

```
void onResume() {
  button.setEnabled(false);
  mediaPlayer.prepareAsync();
```

triggering calls

co
(e.g.

Goal: Summarize possible callback traces


start
$\hat{a}$.onResume()
$\hat{a}$.onPause()
$\hat{\ell}_1$.onPrepared($\hat{m}$)
$\hat{\ell}_2$.onClick($\hat{b}$)
$\hat{a}$.onPause()

Summarizing callback traces

Summarizing callback traces

Compute trace summaries
via static analysis of
enabled-disabled callbacks

Summarizing callback traces

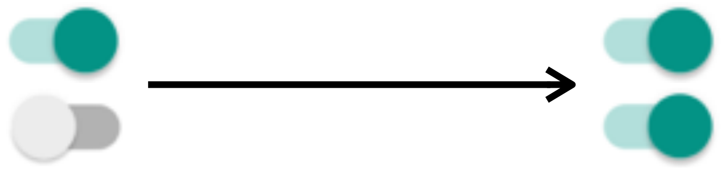
Compute trace summaries via static analysis of enabled-disabled callbacks

relying on specification of call-triggers-callback relations

Summarizing callback traces

Compute trace summaries via static analysis of enabled-disabled callbacks

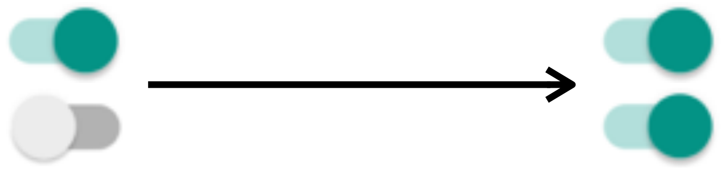
relying on specification of call-triggers-callback relations



Summarizing callback traces

Compute trace summaries via static analysis of enabled-disabled callbacks

relying on specification of call-triggers-callback relations

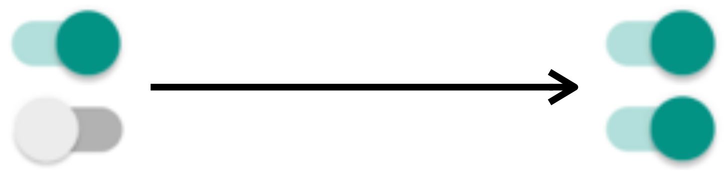


Compute call-triggers-callback relations via automata learning and client synthesis and execution

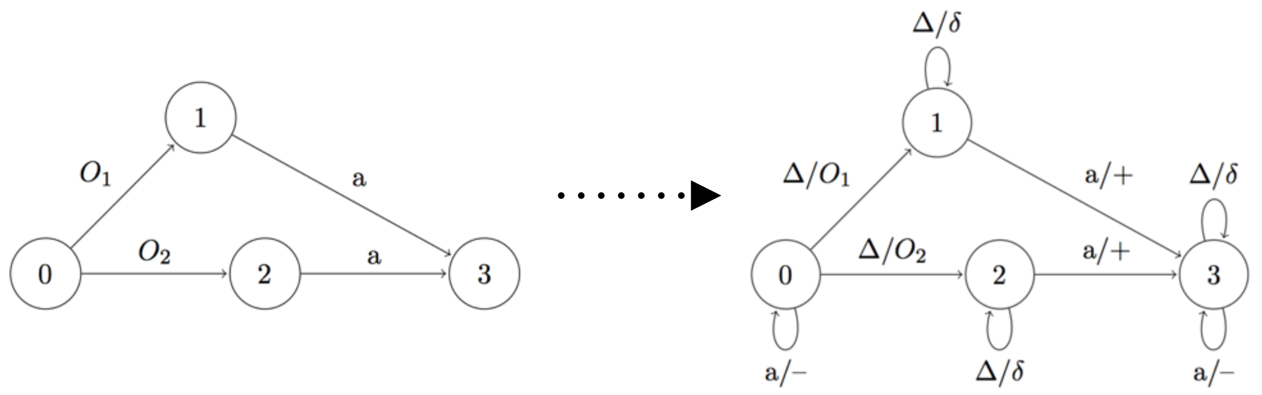
Summarizing callback traces

Compute trace summaries via static analysis of enabled-disabled callbacks

relying on specification of call-triggers-callback relations

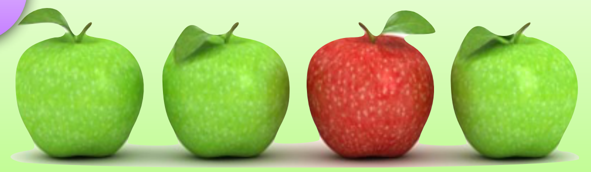


Compute call-triggers-callback relations via automata learning and client synthesis and execution



Summarizing callback traces

1



summarize semantic diff

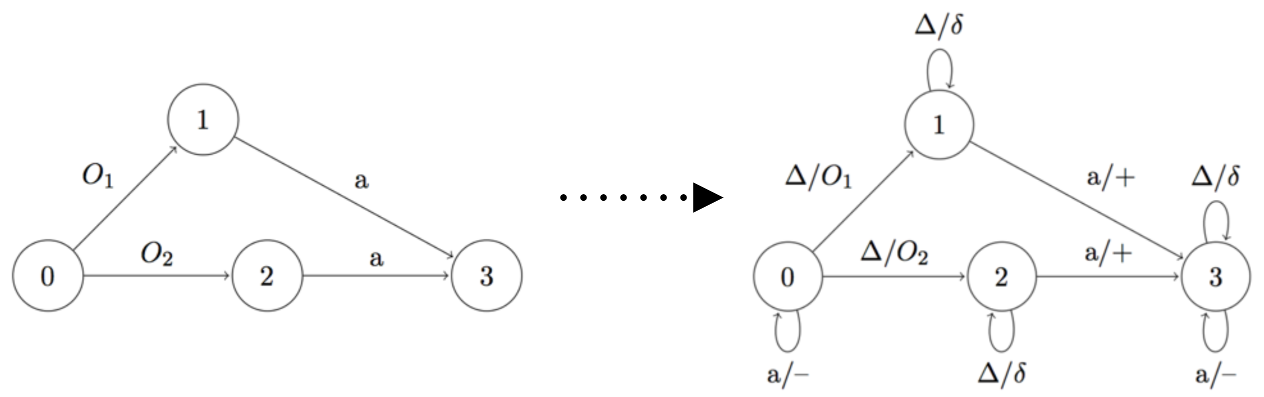
2

derive bug condition

Compute trace summaries via static analysis of enabled-disabled callbacks

relying on specification of call-triggers-callback relations

Compute call-triggers-callback relations via automata learning and client synthesis and execution



Deriving bug conditions

2

derive bug condition

Translate the root cause of the bug into a app-specific condition

Deriving bug conditions

2

derive bug condition

Translate the root cause of the bug into a app-specific condition

to apply to the corpus at scale

Deriving bug conditions

Translate the root cause of the bug into a app-specific condition

to apply to the corpus at scale

App
Android Framework

Deriving bug conditions

2

derive bug condition

Translate the root cause of the bug into a app-specific condition

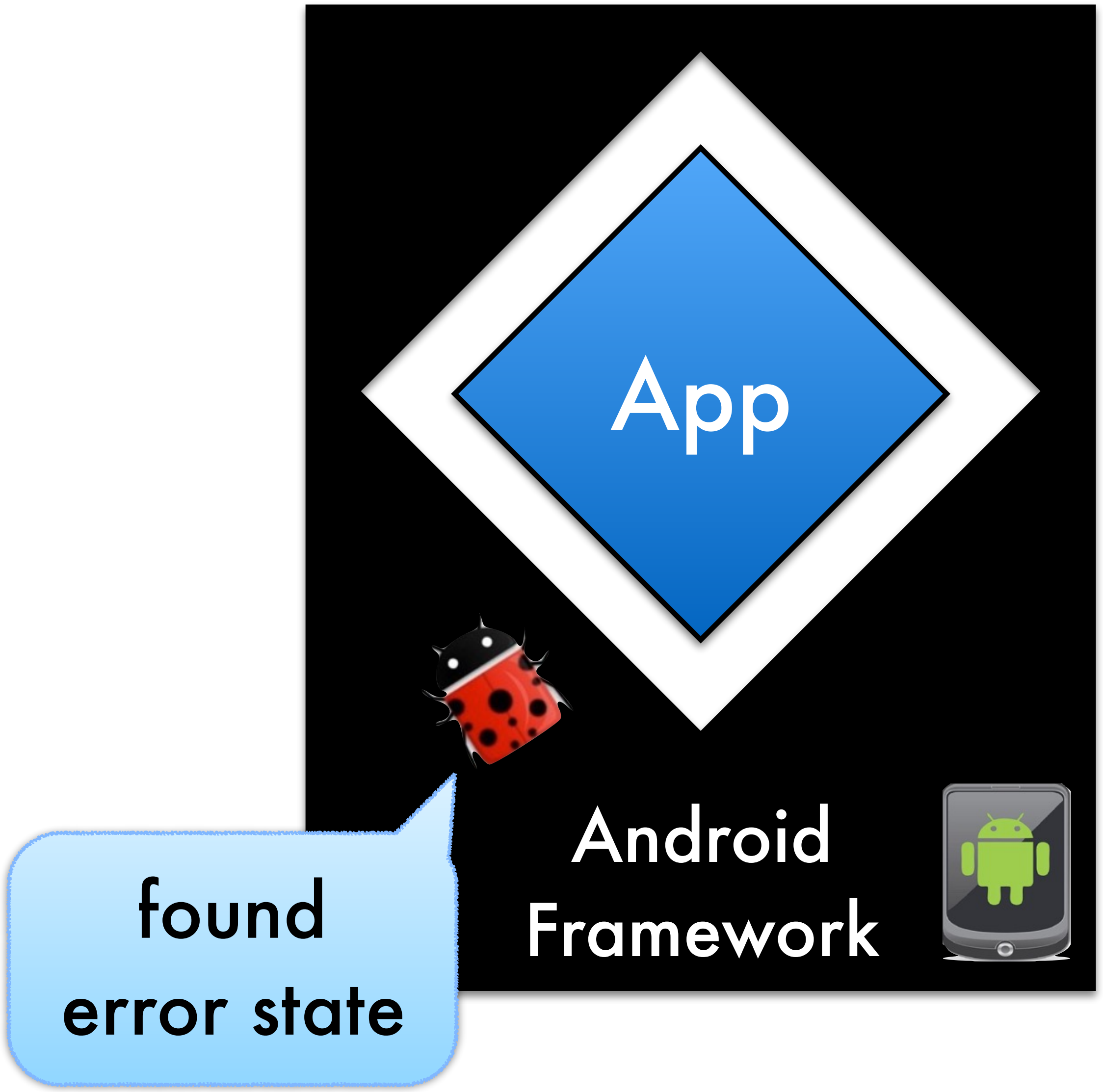
to apply to the corpus at scale

App
Android Framework

Deriving bug conditions

Translate the root cause of the bug into a app-specific condition

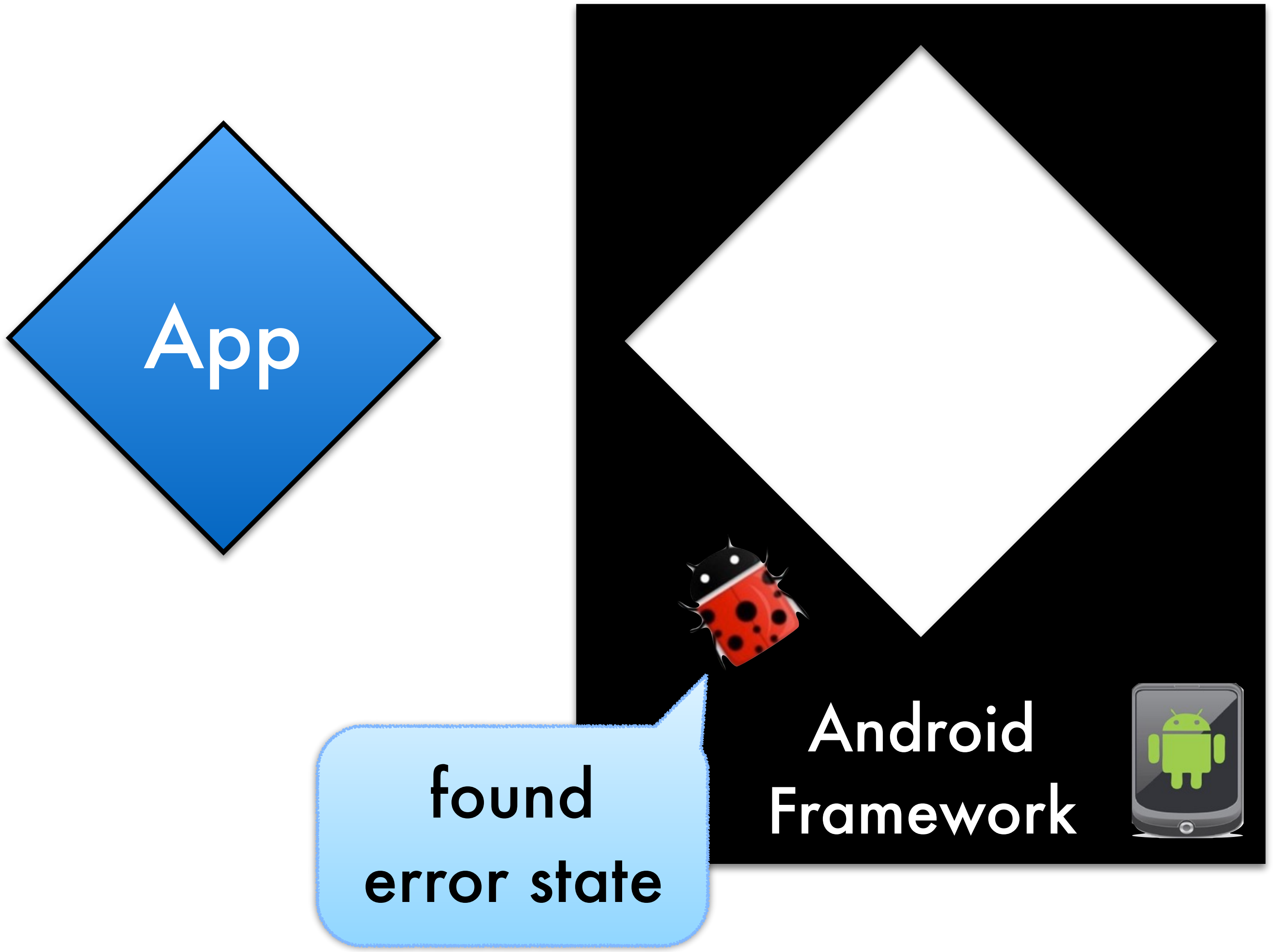
to apply to the corpus at scale



Deriving bug conditions

Translate the root cause of the bug into a app-specific condition

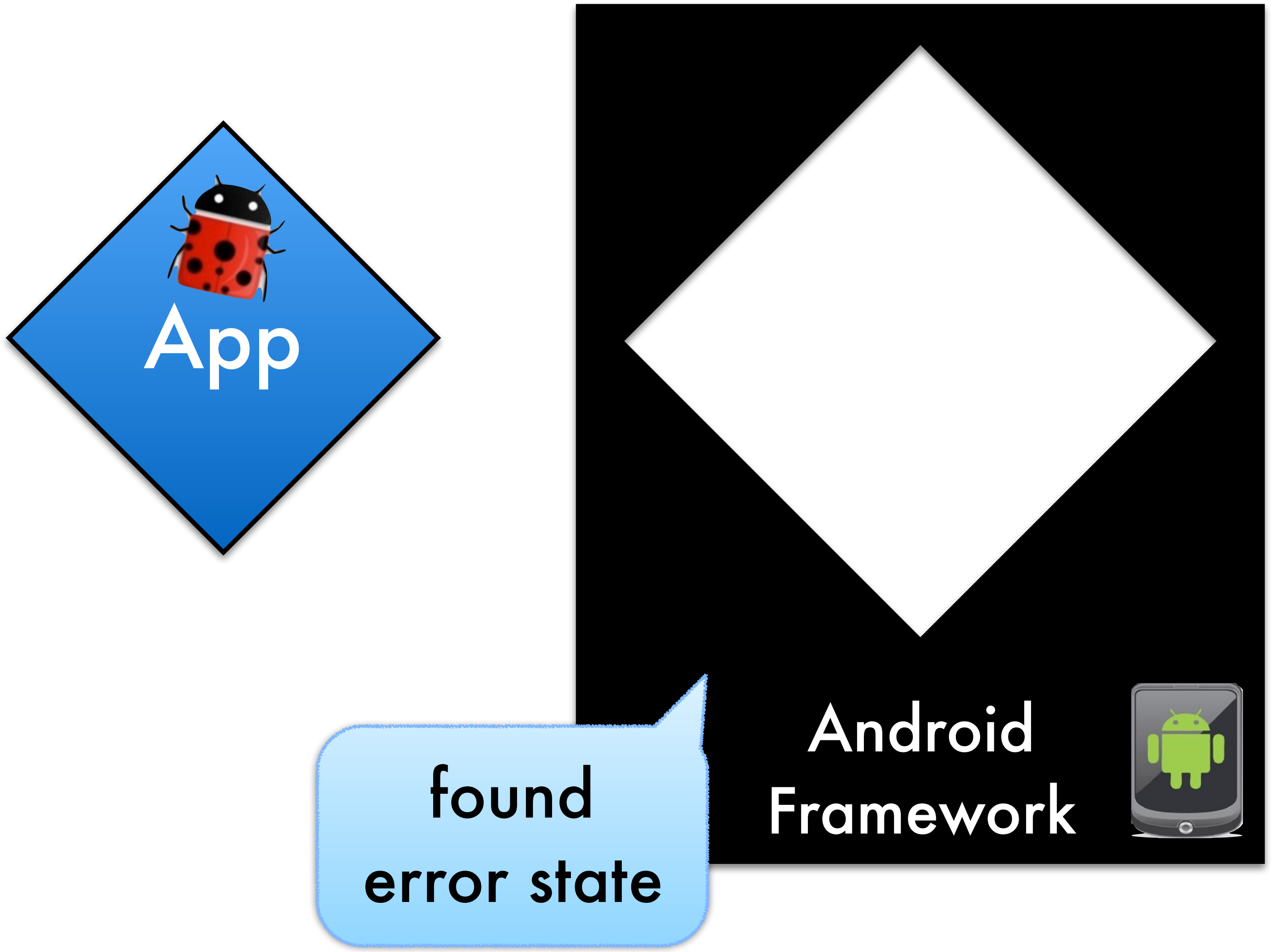
to apply to the corpus at scale



Deriving bug conditions

Translate the root cause of the bug into a app-specific condition

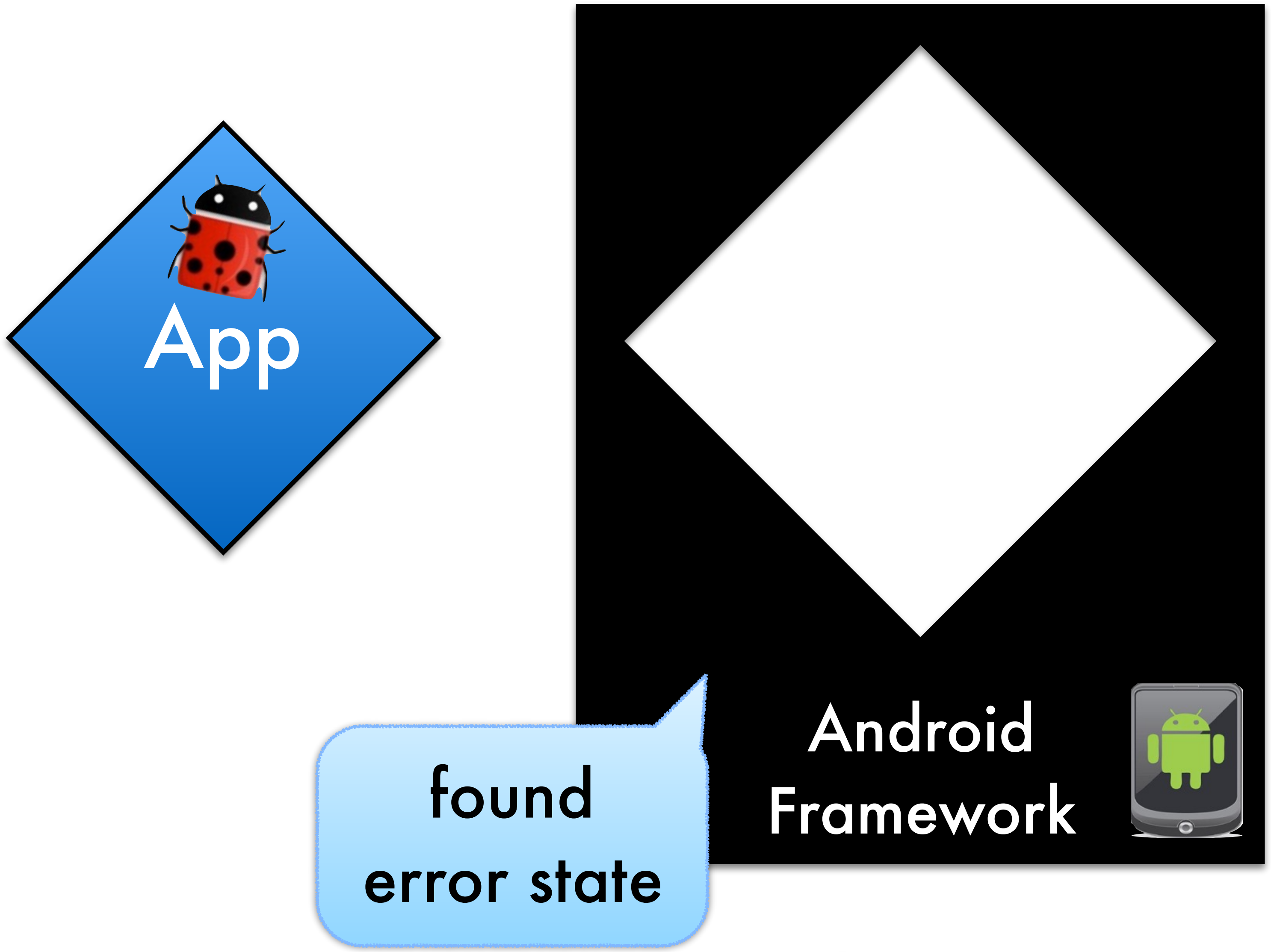
to apply to the corpus at scale



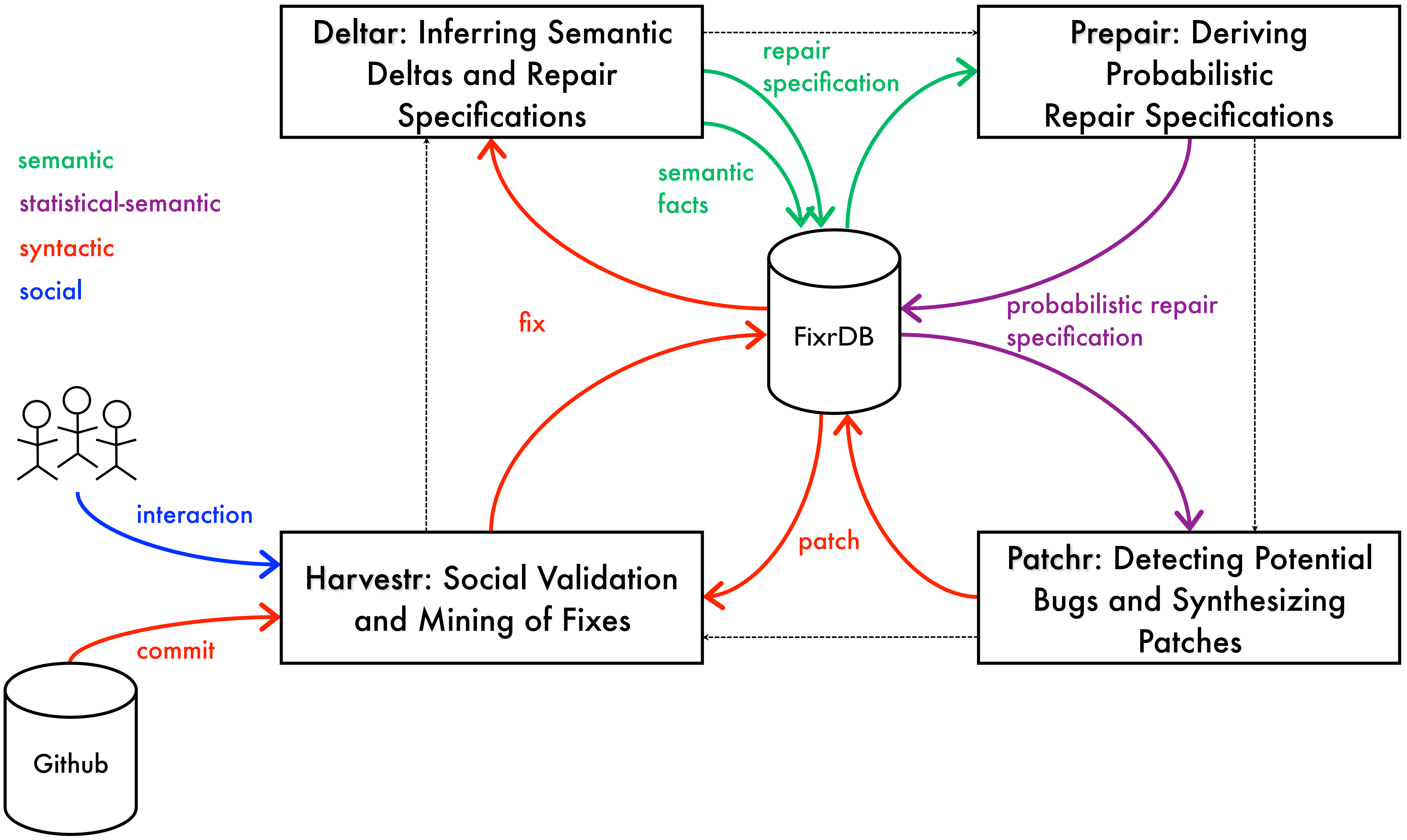
Deriving bug conditions

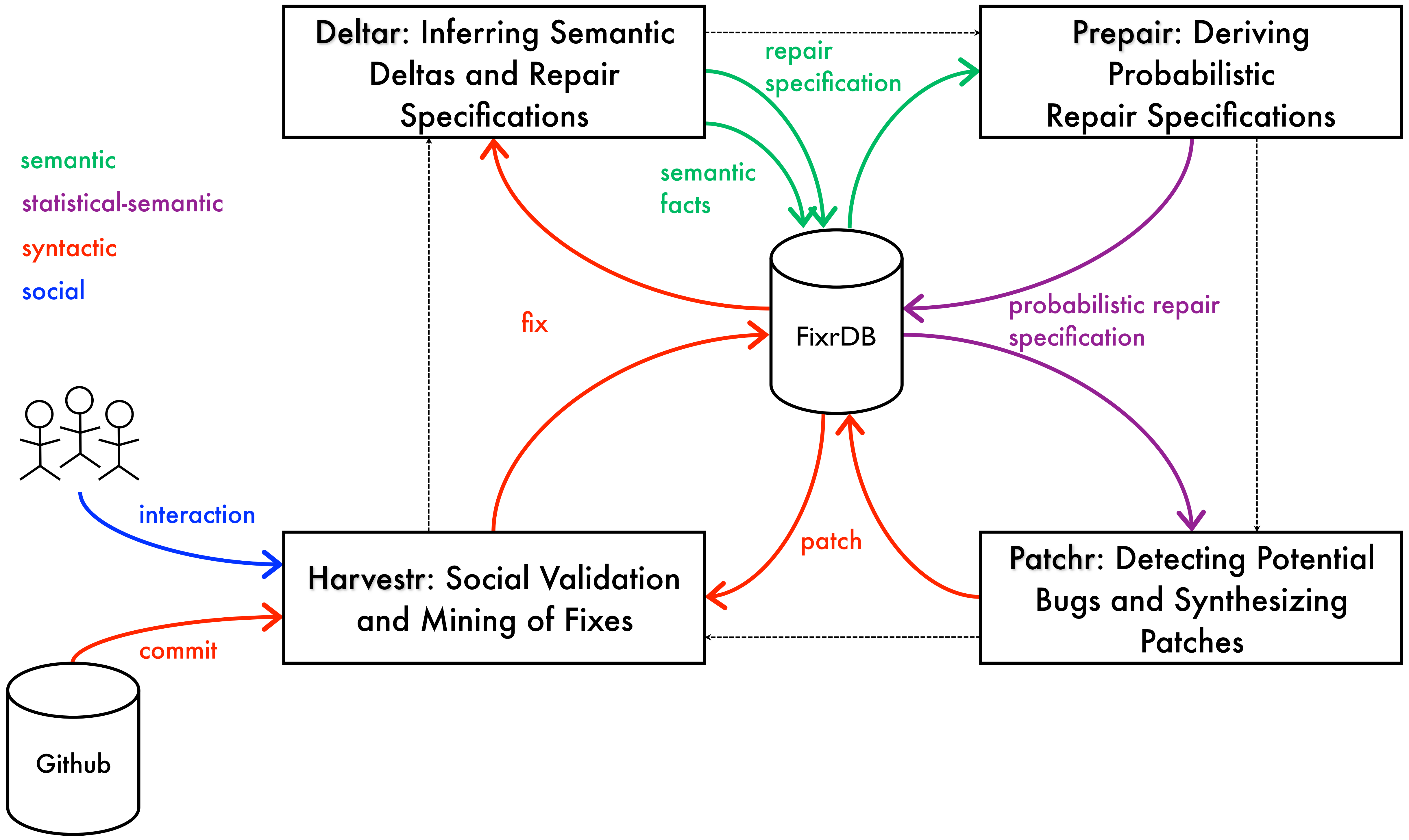
Translate the root cause of the bug into a app-specific condition

to apply to the corpus at scale



Direction: Applying goal-directed static analysis on the bugfix commit.


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
FixrDB
fix
probabilistic repair specification
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

Bor-Yuh Evan Chang

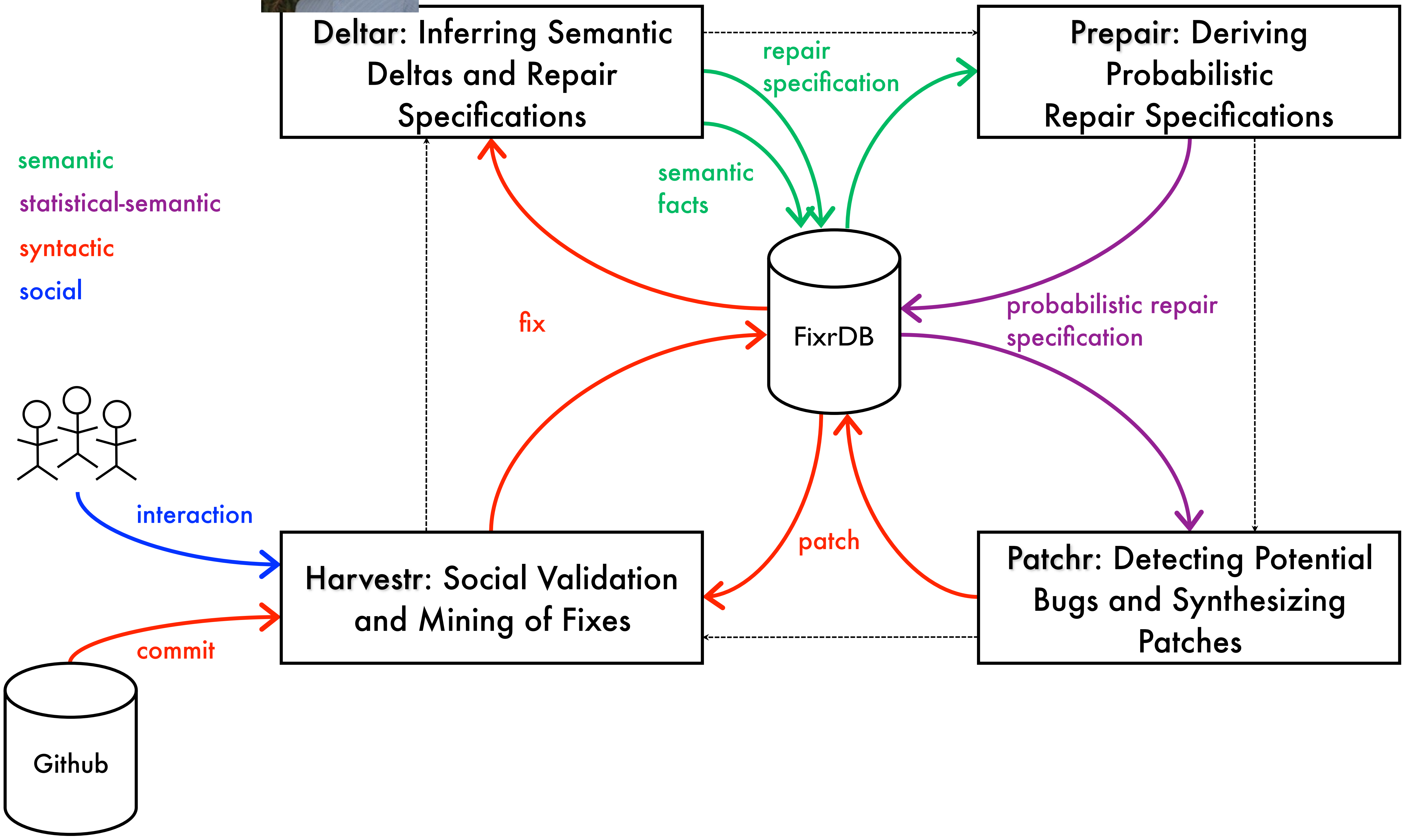
symbolic
program analysis

semantic

statistical-semantic

syntactic

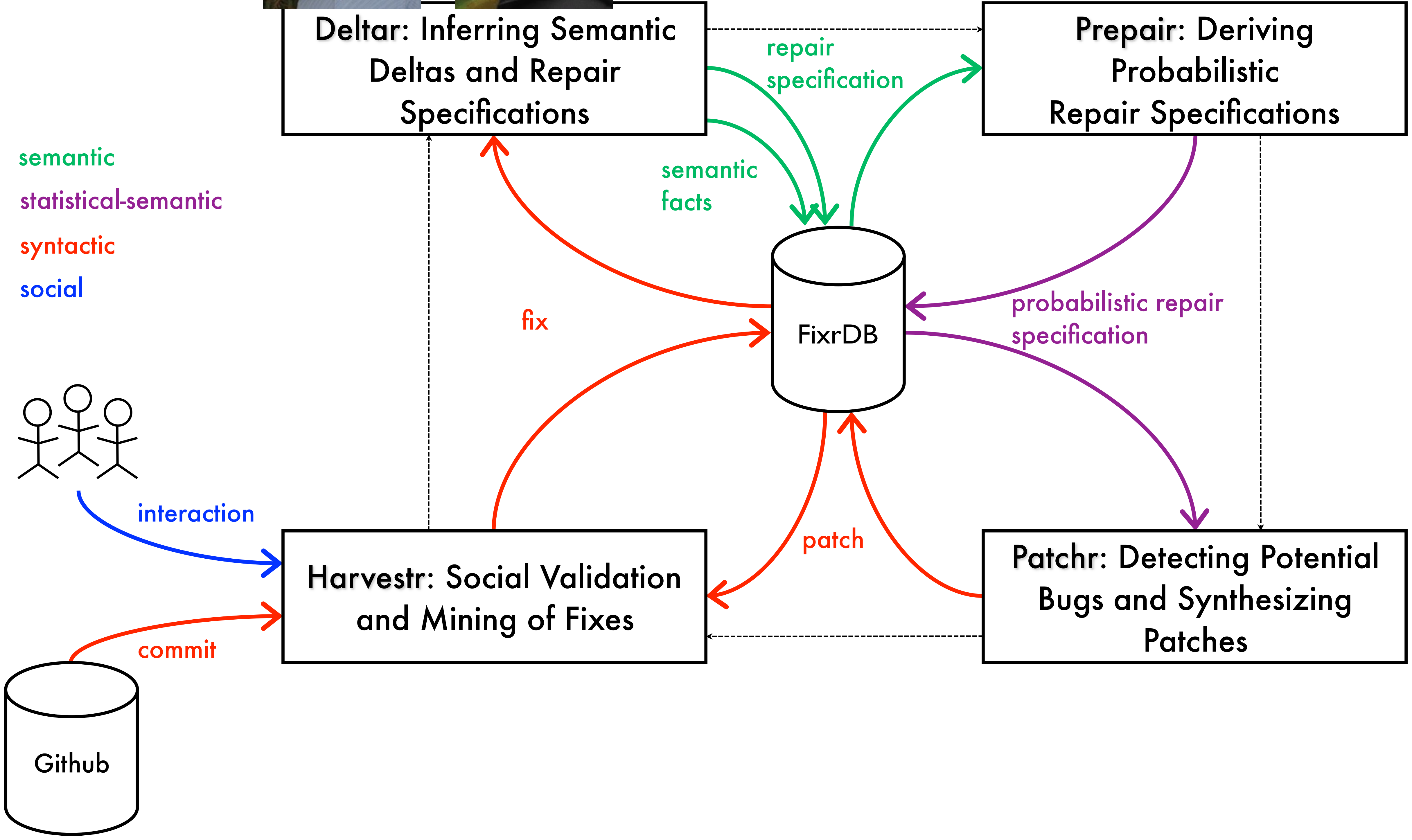
social


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
fix
FixrDB
probabilistic repair specification
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

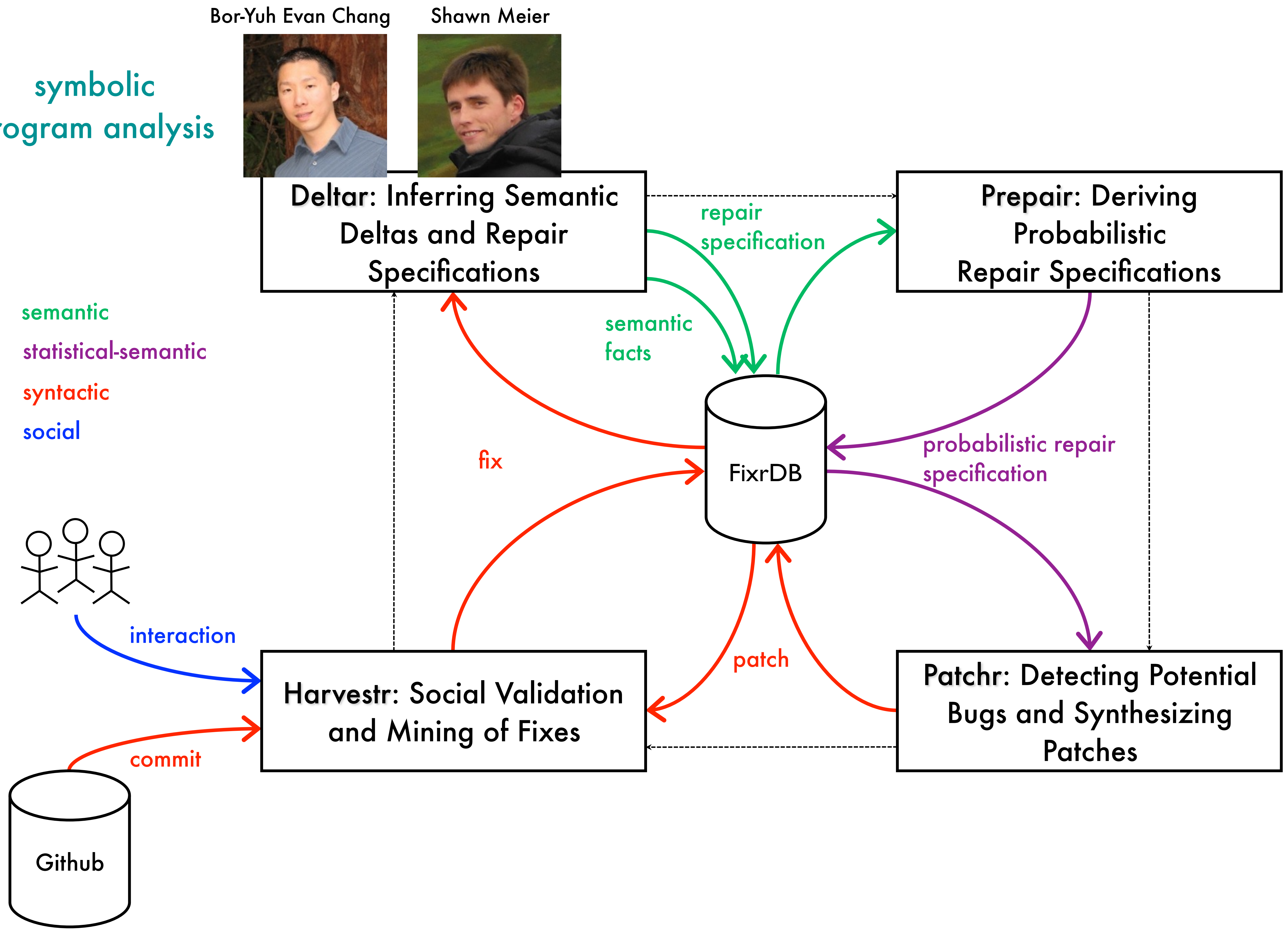
symbolic
program analysis

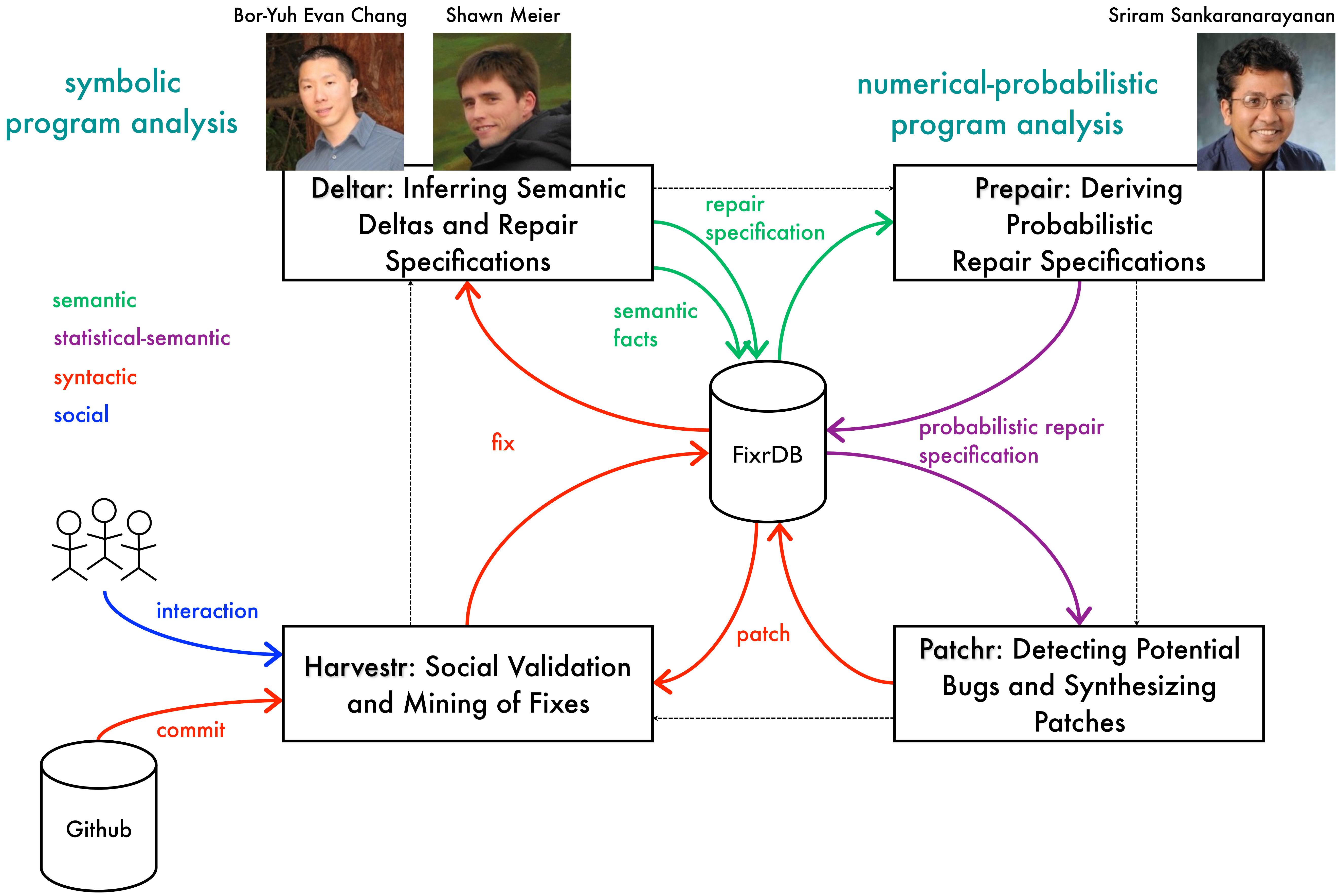
Bor-Yuh Evan Chang

Shawn Meier


Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
FixrDB
fix
probabilistic repair specification
interaction
Harvestr: Social Validation and Mining of Fixes
patch
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

symbolic program analysis


Bor-Yuh Evan Chang
Shawn Meier
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
Github
repair specification
semantic facts
probabilistic repair specification
fix
patch
interaction
commit
semantic
statistical-semantic
syntactic
social


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github

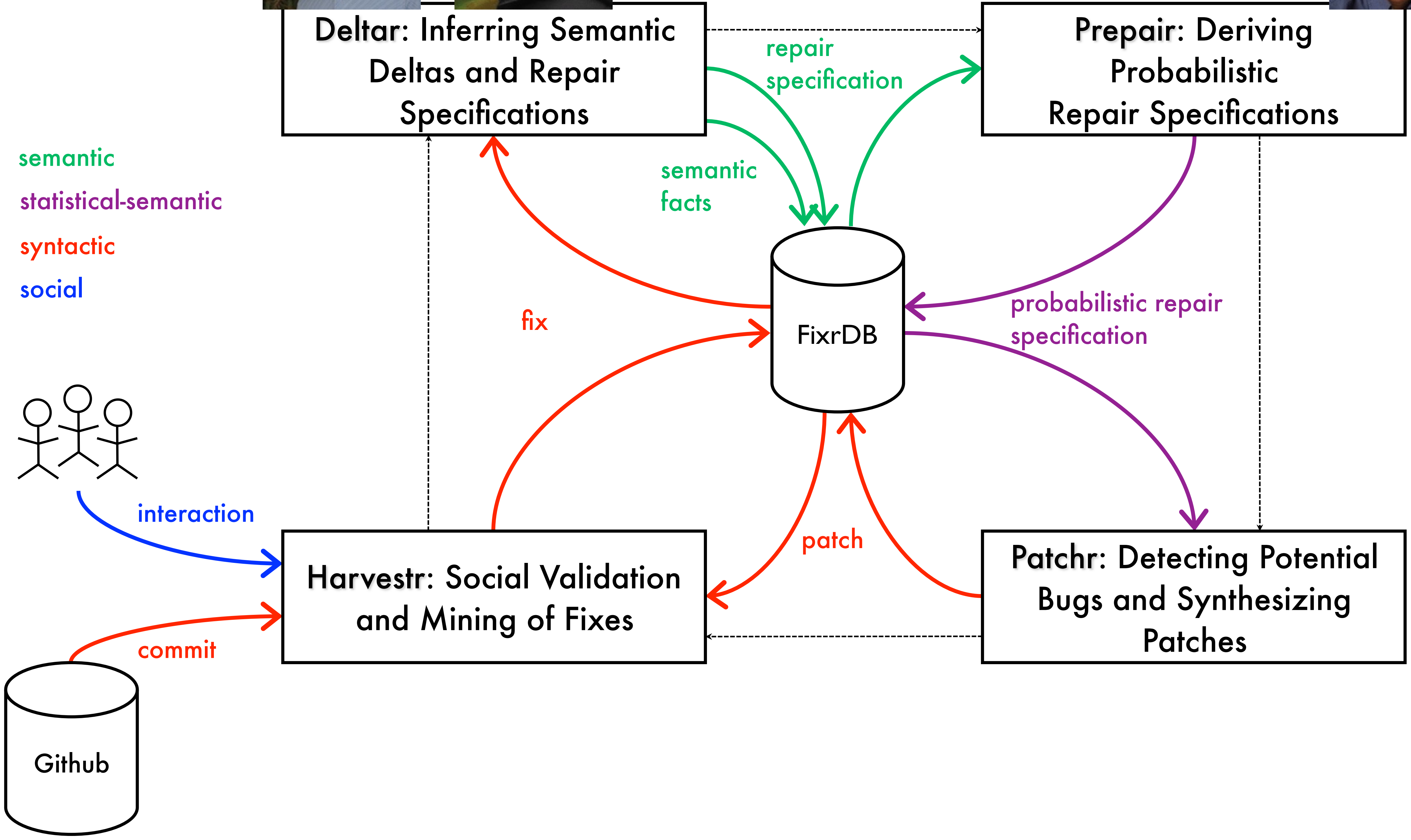
Bor-Yuh Evan Chang

Shawn Meier

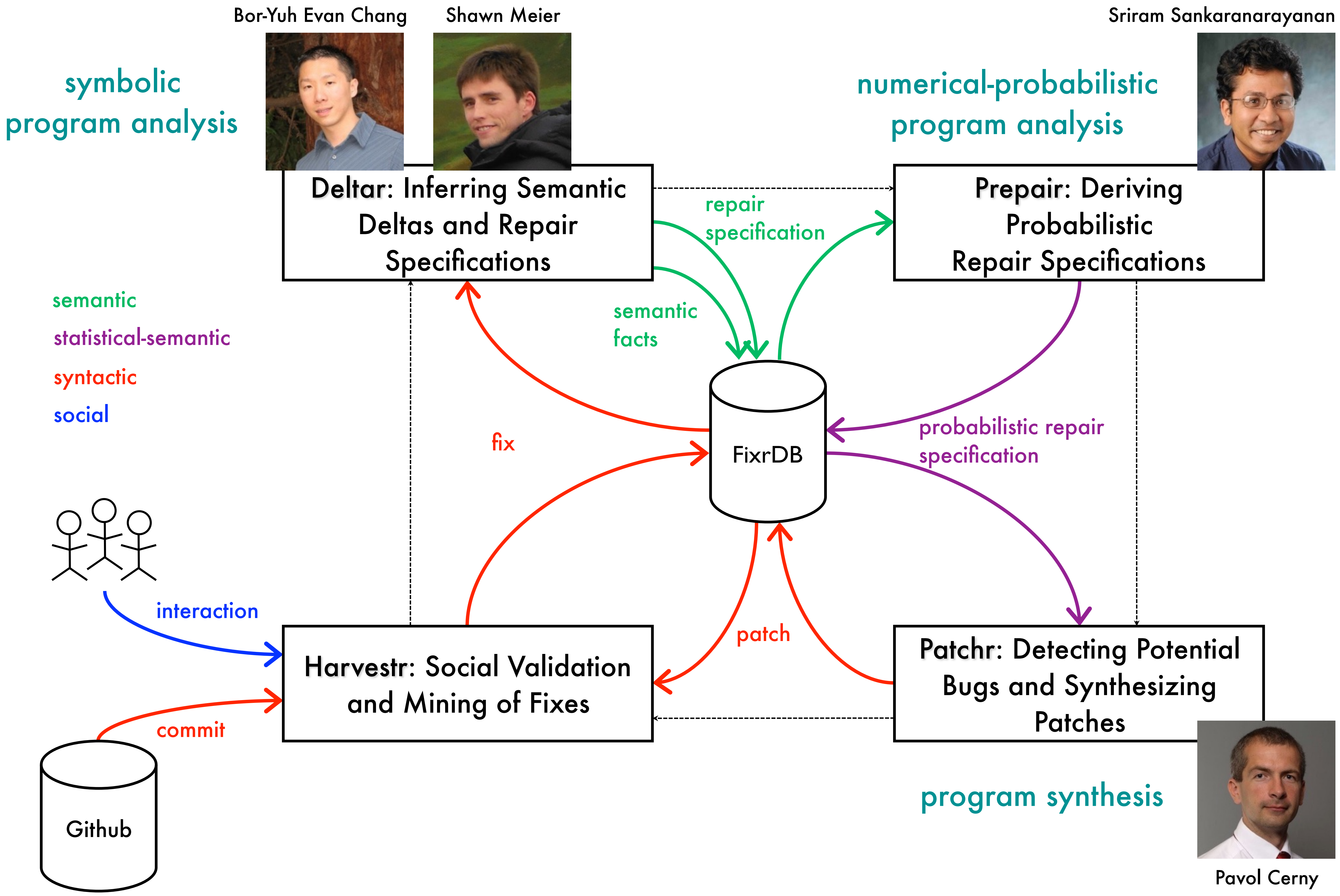
Sriram Sankaranarayanan

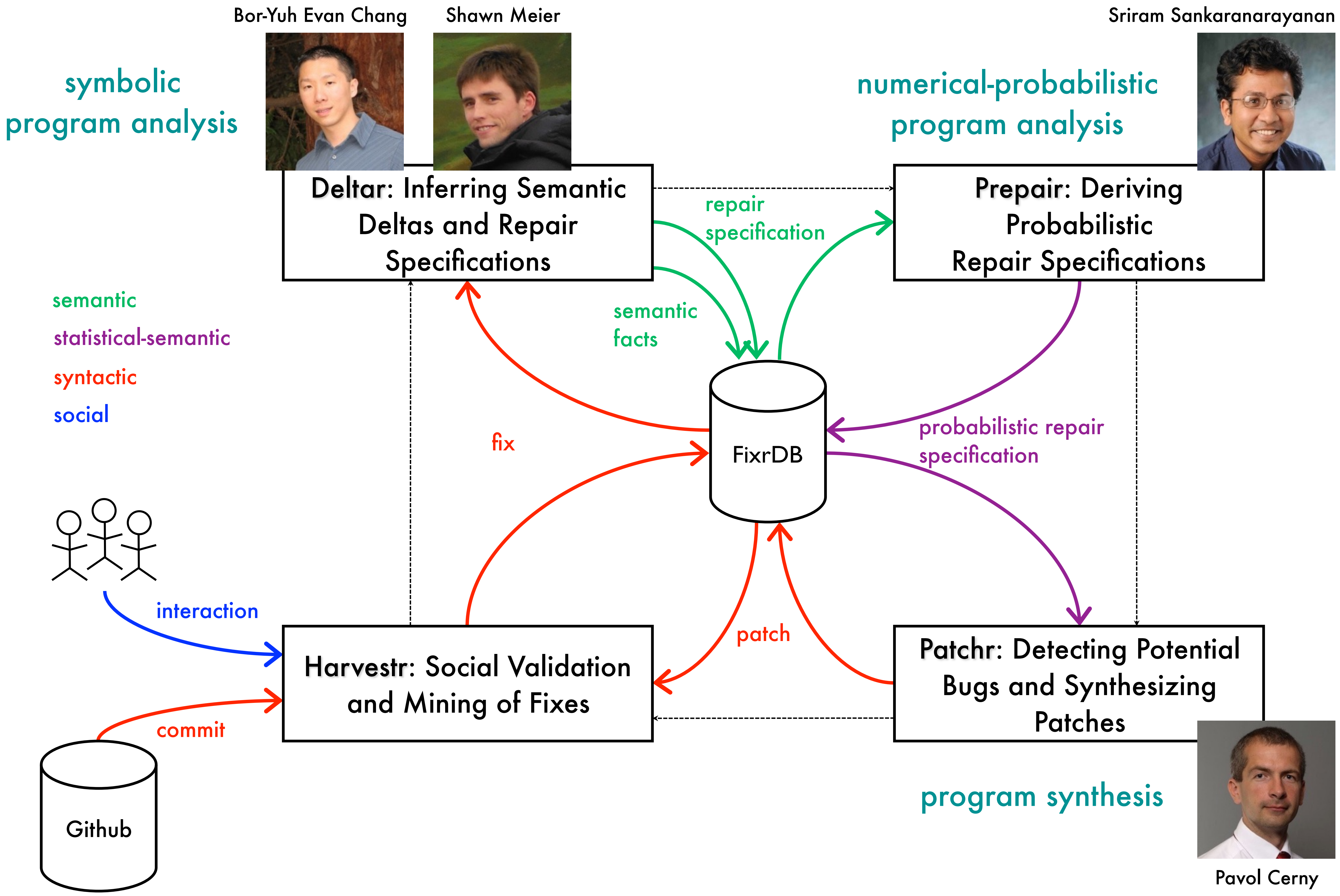
symbolic program analysis

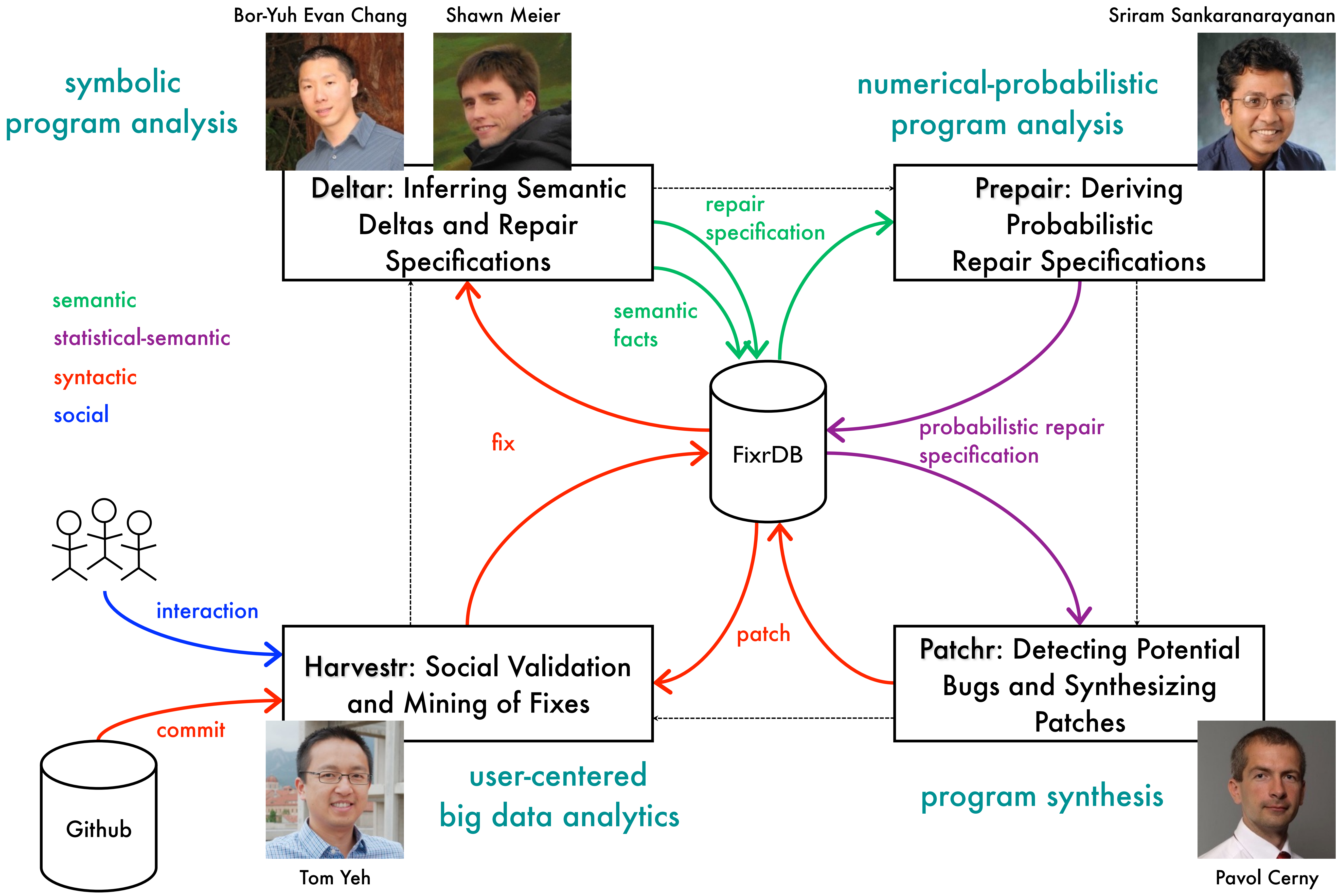
numerical-probabilistic program analysis

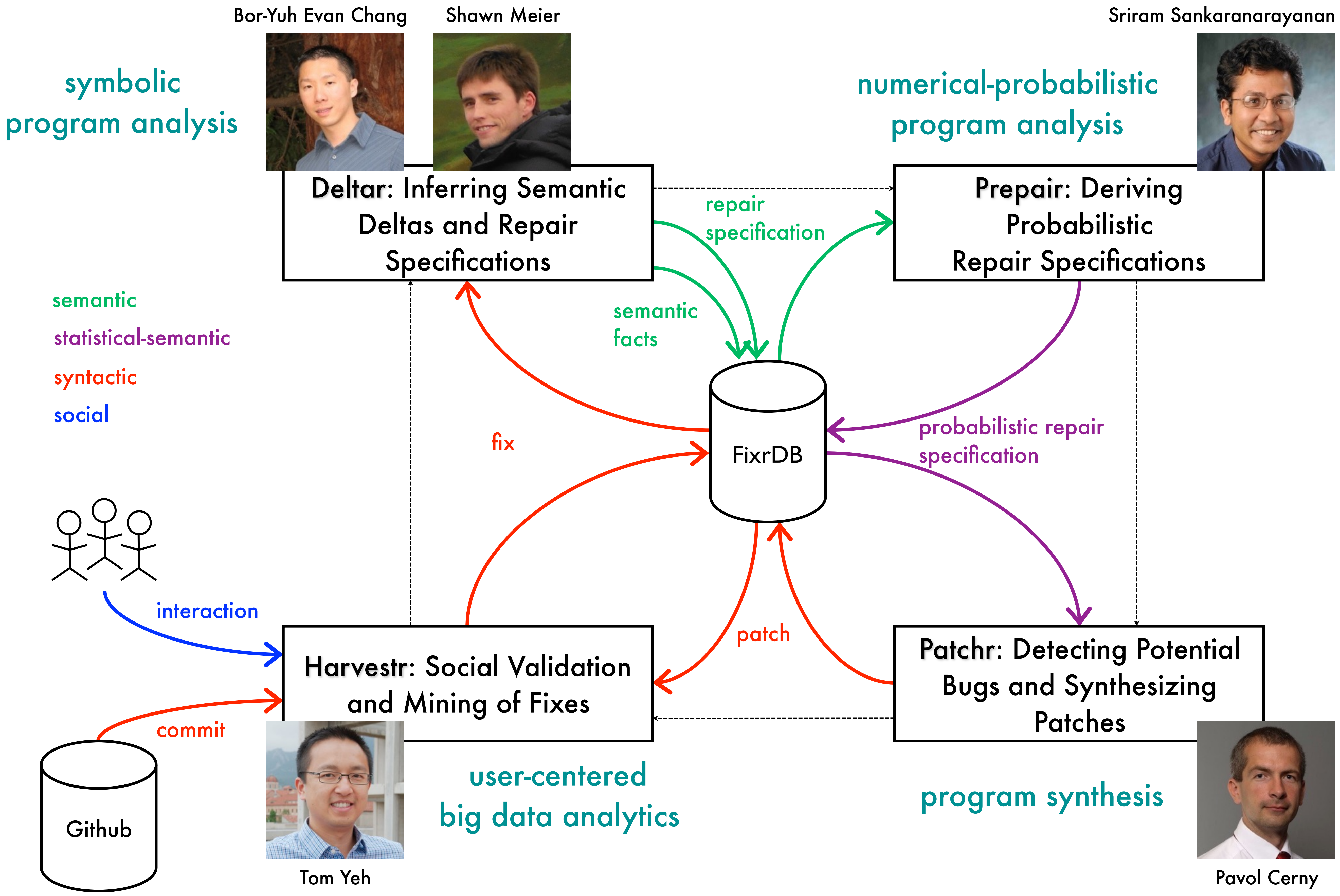


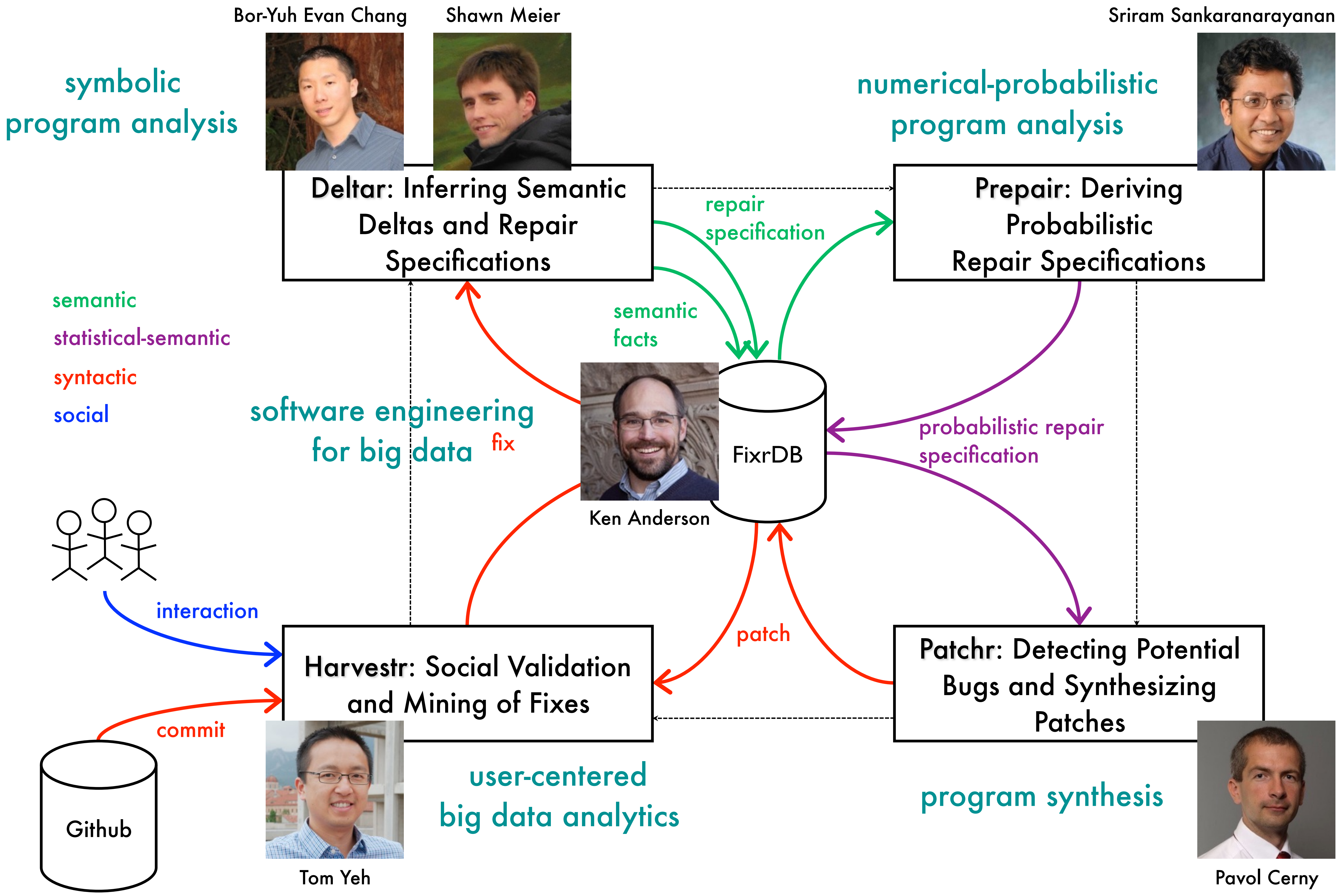
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github
program synthesis
Pavol Cerny


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github
program synthesis
Pavol Cerny


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github
user-centered big data analytics
program synthesis
Tom Yeh
Pavol Cerny


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
fix
FixrDB
probabilistic repair specification
interaction
patch
commit
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
Github
user-centered big data analytics
program synthesis
Tom Yeh
Pavol Cerny


Bor-Yuh Evan Chang
Shawn Meier
Sriram Sankaranarayanan
symbolic program analysis
numerical-probabilistic program analysis
Deltar: Inferring Semantic Deltas and Repair Specifications
Prepair: Deriving Probabilistic Repair Specifications
repair specification
semantic facts
semantic
statistical-semantic
syntactic
social
software engineering for big data
fix
FixrDB
Ken Anderson
probabilistic repair specification
interaction
patch
Harvestr: Social Validation and Mining of Fixes
Patchr: Detecting Potential Bugs and Synthesizing Patches
commit
Github
user-centered big data analytics
program synthesis
Tom Yeh
Pavol Cerny

Summary

Summary

The **Fixr** loop: Mine framework specifications from bugfix commits.

Challenge: Finding bugfix commits

Challenge: Summarizing and generalizing bugfix commits

Summary

The **Fixr** loop: Mine framework specifications from bugfix commits.

Challenge: Finding bugfix commits

Challenge: Summarizing and generalizing bugfix commits

Prototype infrastructure for finding trends (at scale), querying for bug conditions, and applying patches.

Summary

The **Fixr** loop: Mine framework specifications from bugfix commits.

Challenge: Finding bugfix commits

Challenge: Summarizing and generalizing bugfix commits

Prototype infrastructure for finding trends (at scale), querying for bug conditions, and applying patches.

Research directions in summarizing callback traces and translating root cause to bug conditions.