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





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

University of Colorado Boulder

MUSE Kickoff
October 15, 2014

PLV

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 …

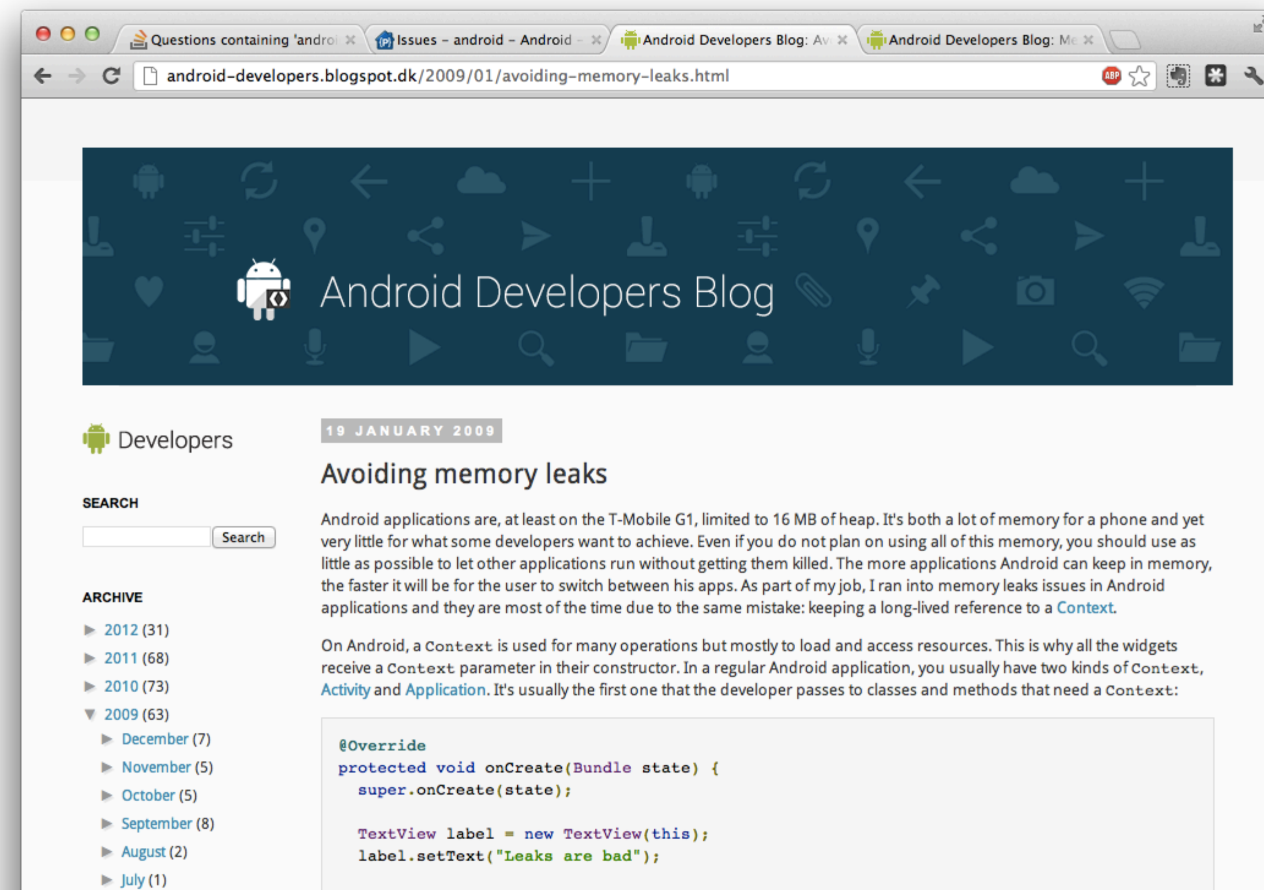
caused by an app-created memory leak

Ask framework devs ...

Ask framework devs …

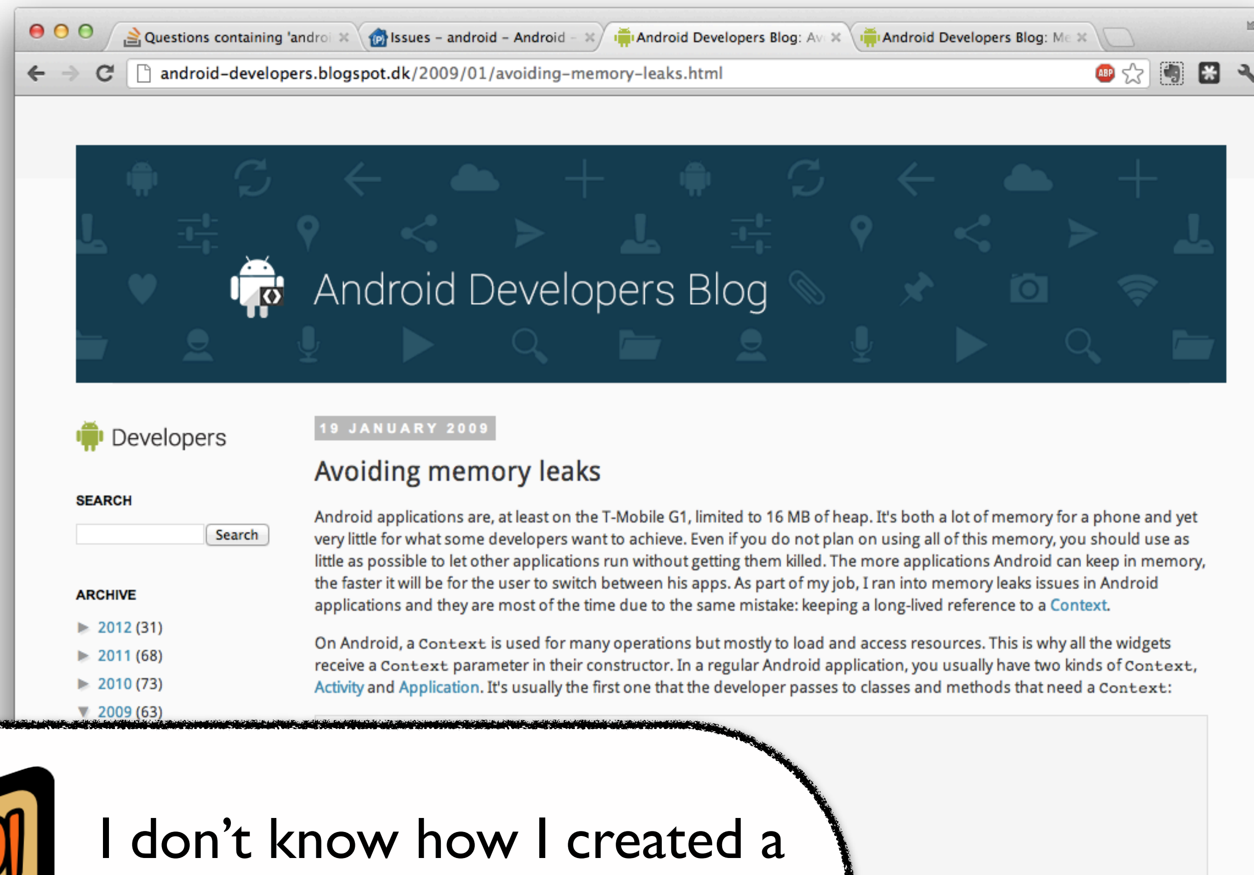
Ask framework devs …

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



Ask framework devs ...

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



Questions containing 'andro | Issues – android – Android | Android Developers Blog: A | Android Developers Blog: M

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

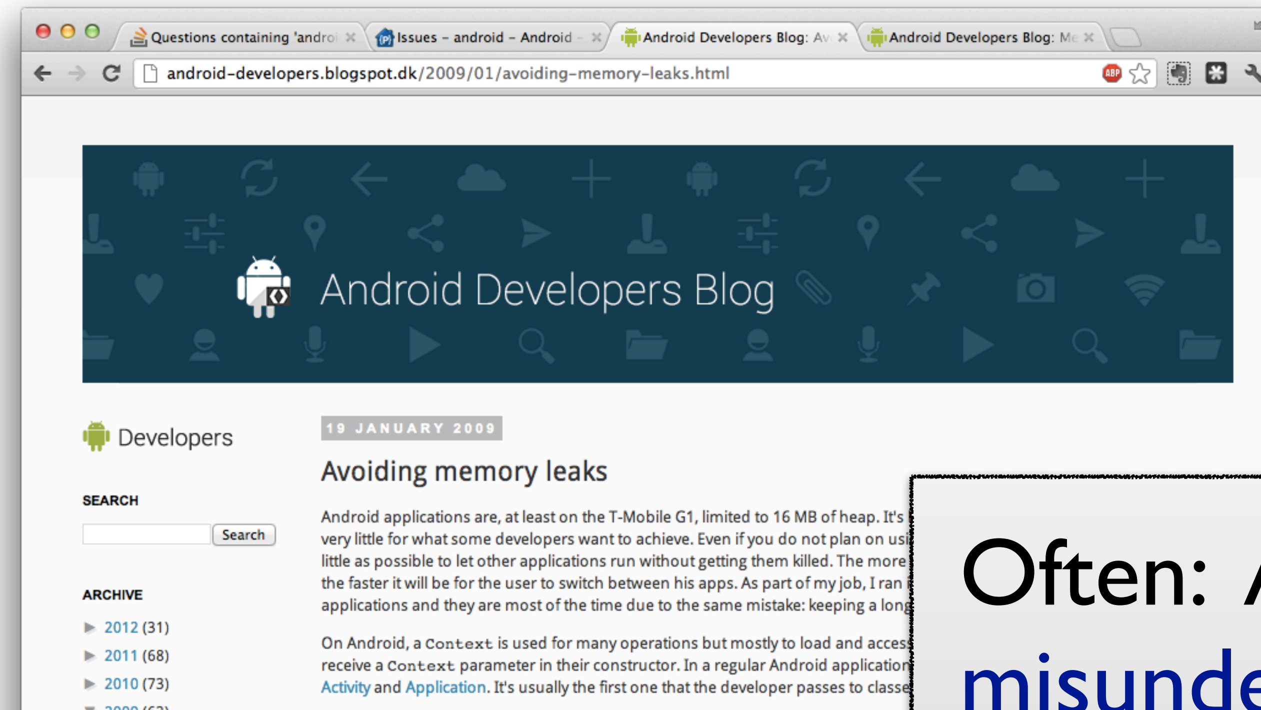
Avoiding memory leaks

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`:

Ask framework devs ...

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

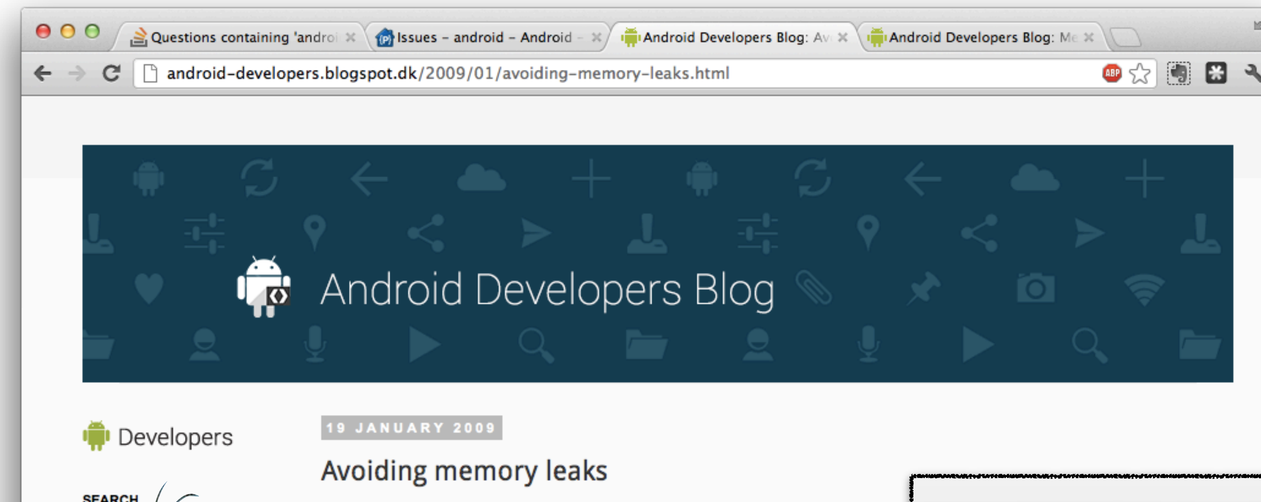


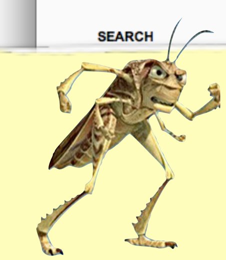
I don't know how I created a long-lived reference to an `Activity`!

Often: A **misunderstanding** of a library causes the **library** to keep the `Activity` reference.

Ask framework devs ...

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





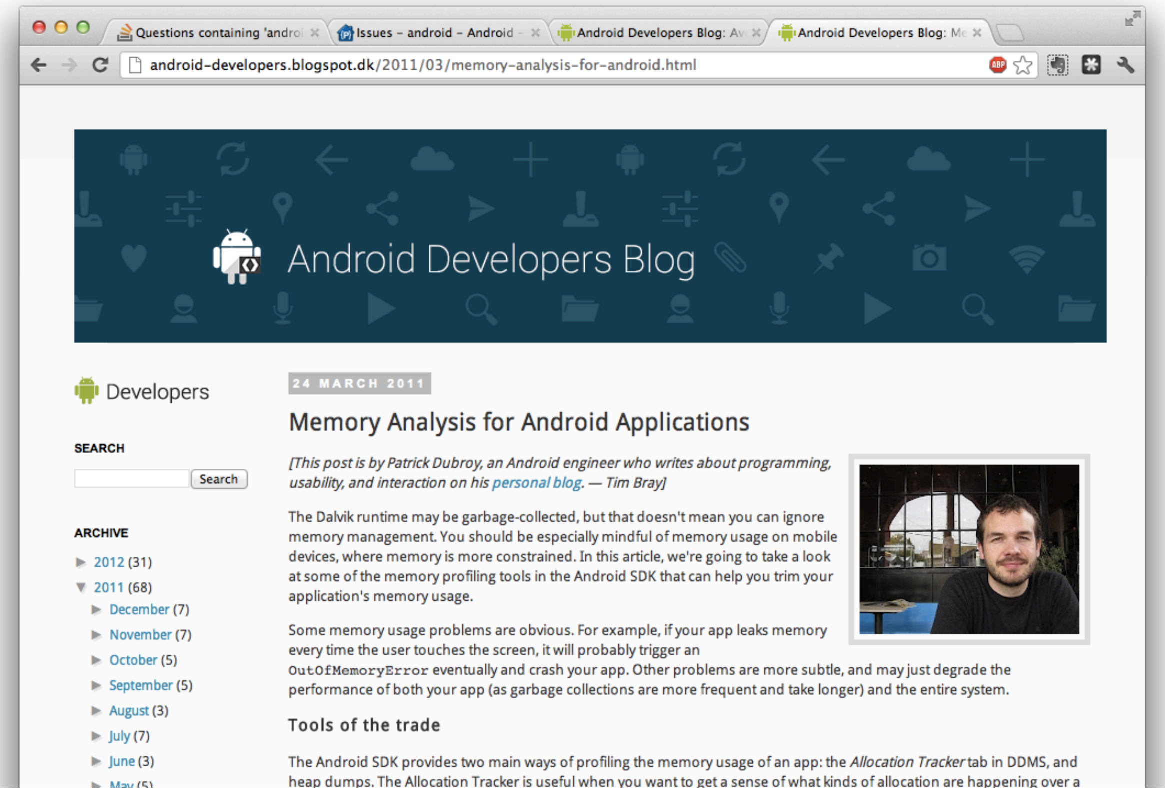
Bug from violating
(implicit) framework protocol rules

Imagining a post-MUSE scenario ...

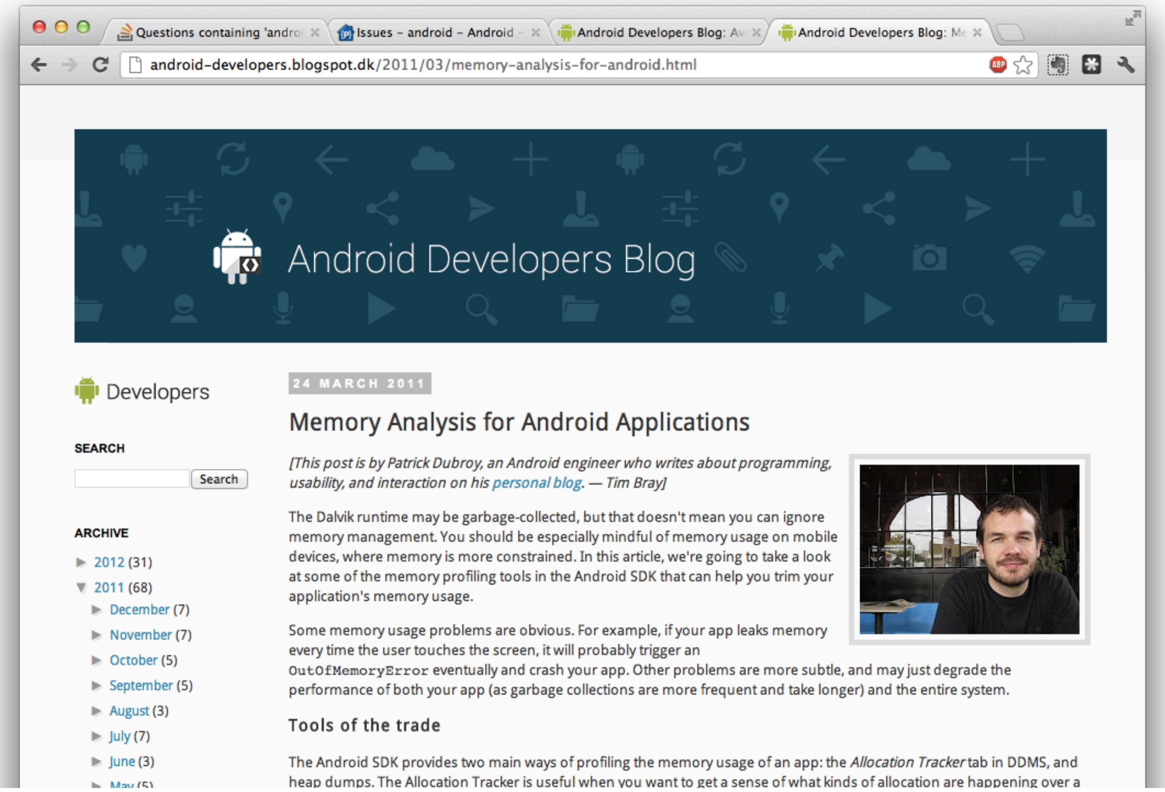
for 

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

Elsewhere, following the state of practice for debugging leaks ...

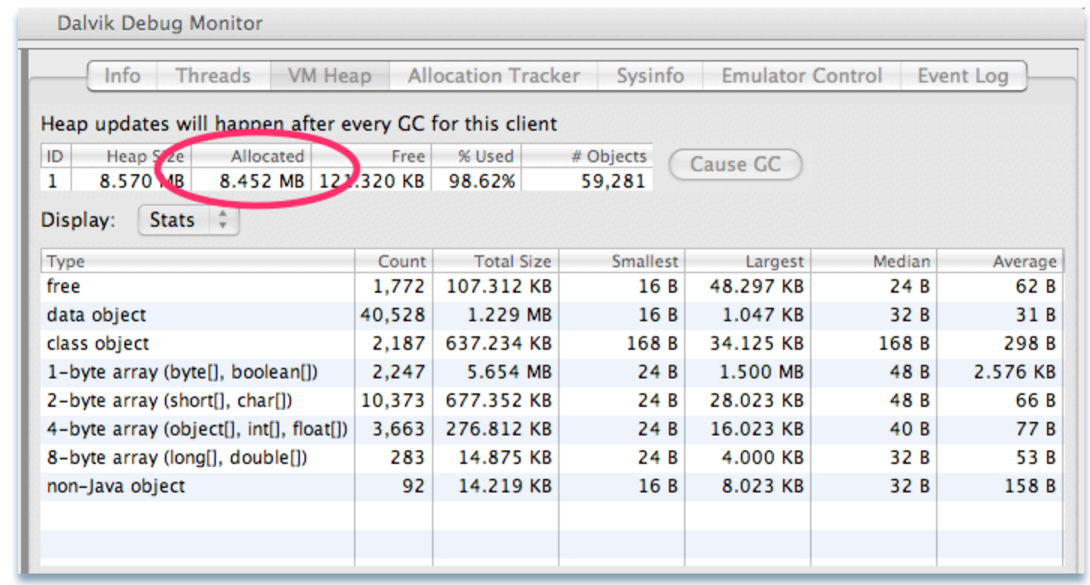


Elsewhere, following the state of practice for debugging leaks ...



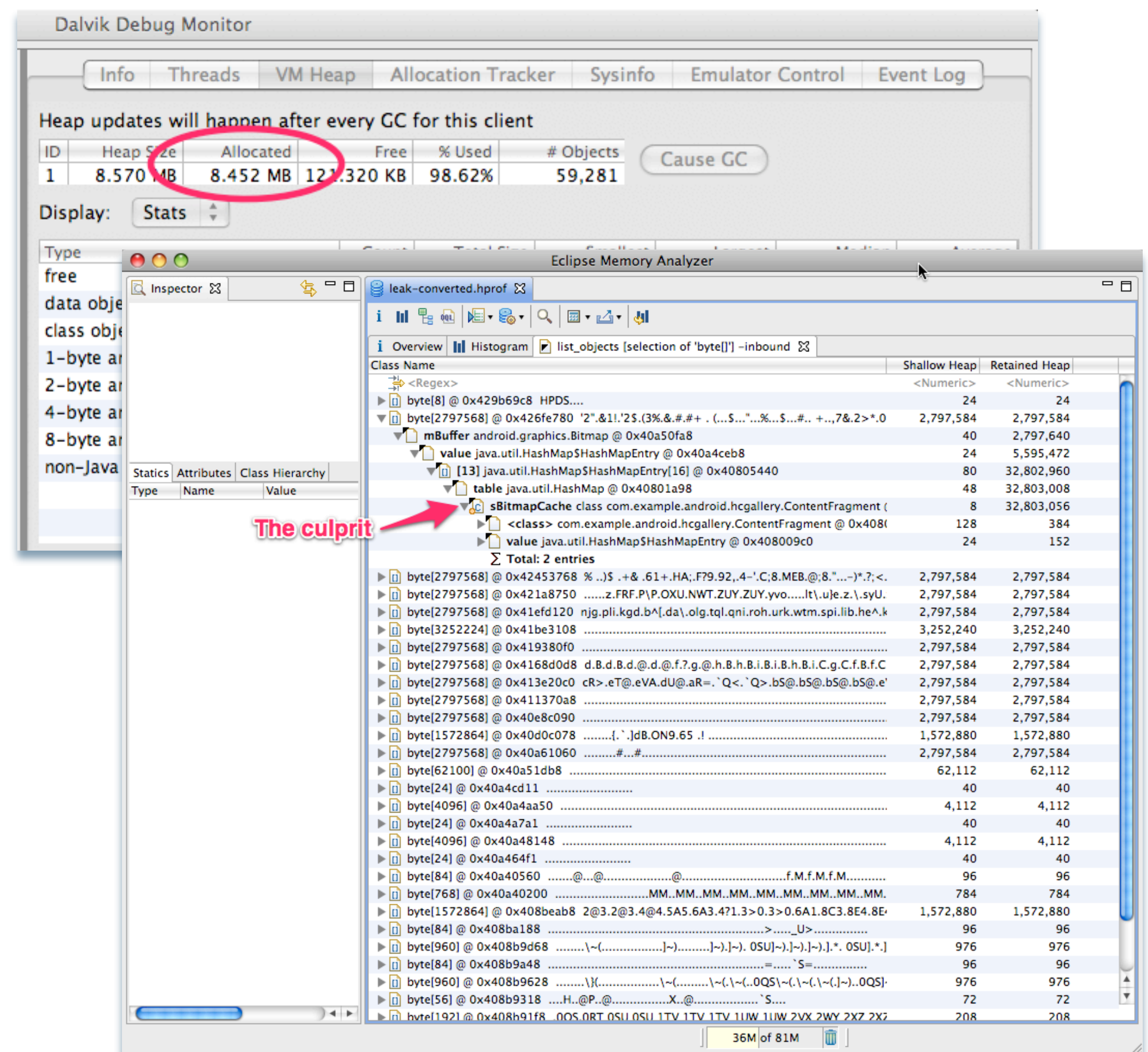
1. Run the app

Elsewhere, following the state of practice for debugging leaks ...



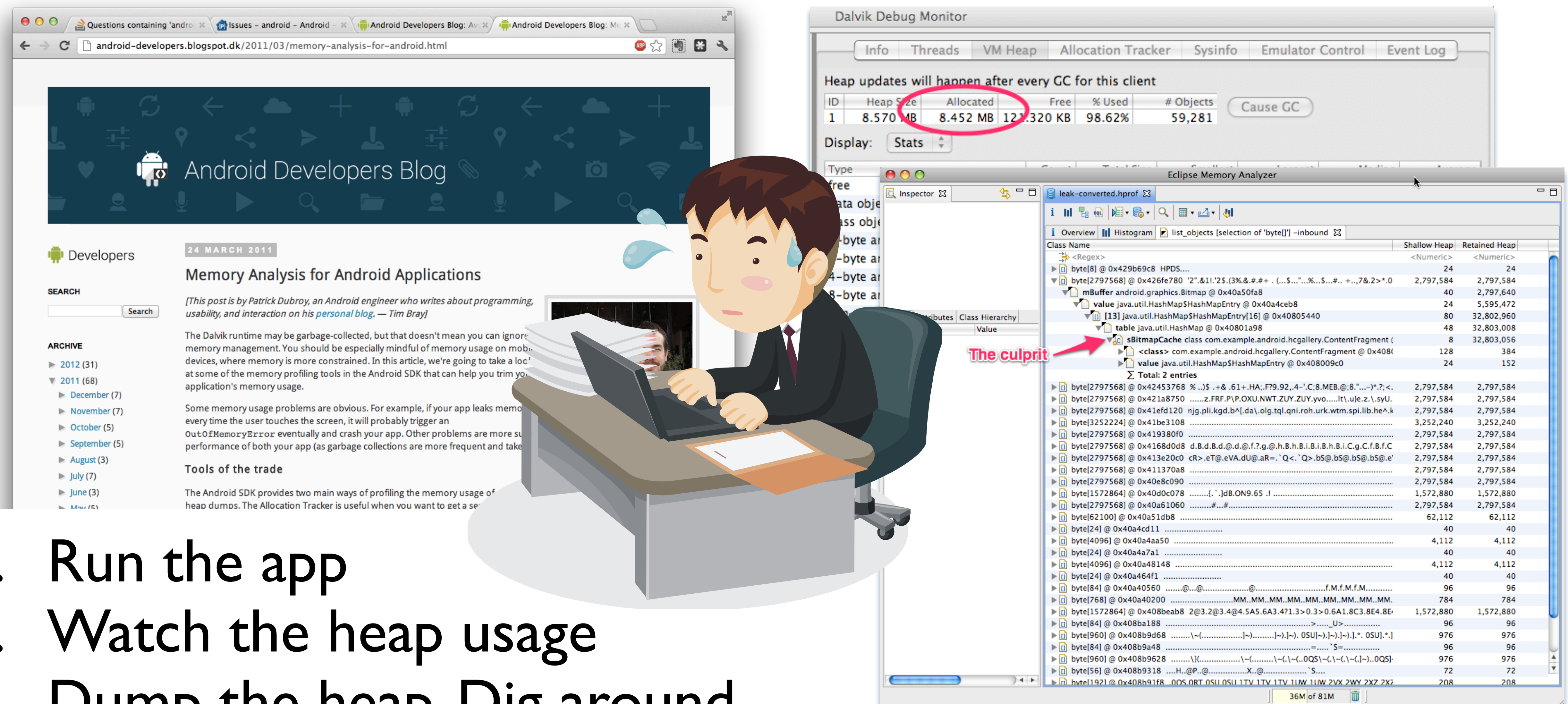
1. Run the app
2. Watch the heap usage

Elsewhere, following the state of practice for debugging leaks ...



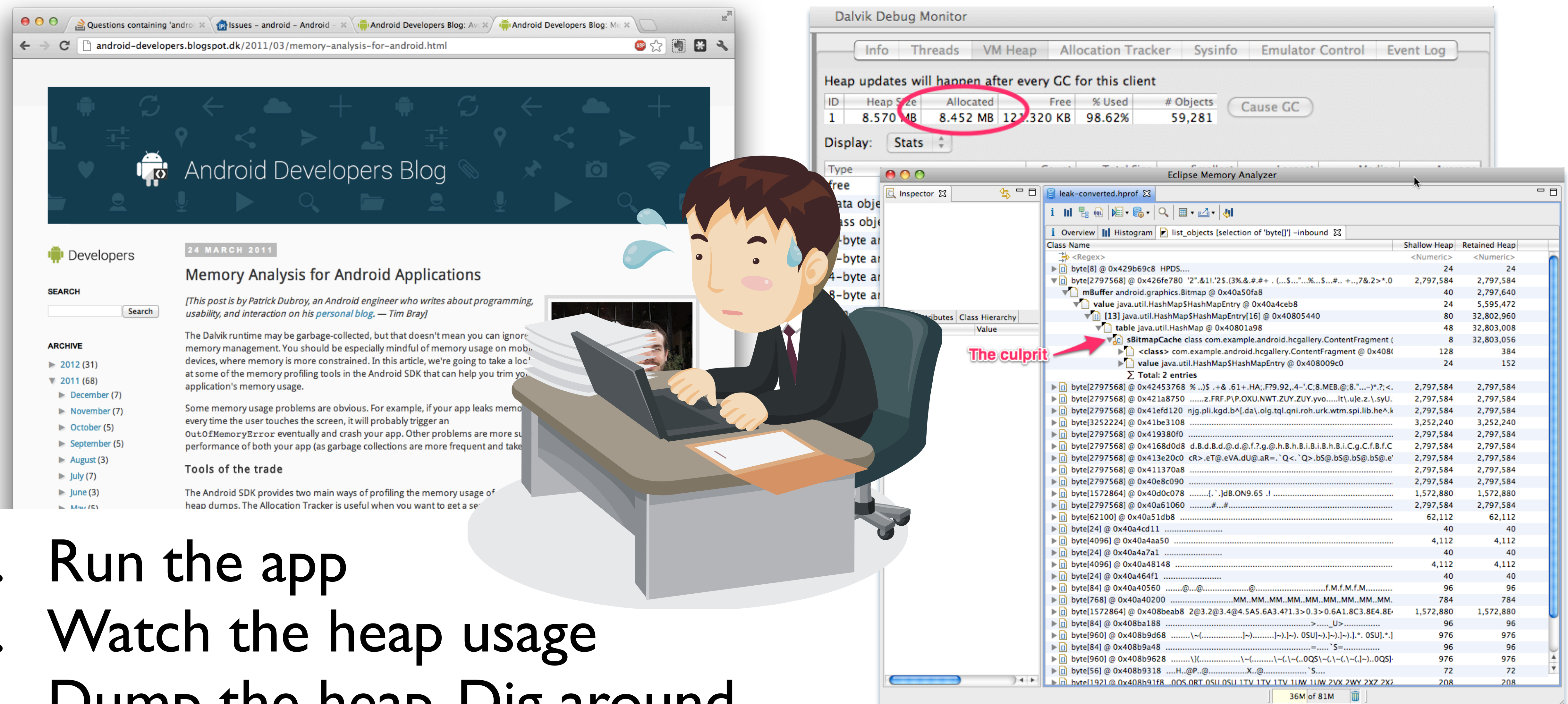
1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!

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!

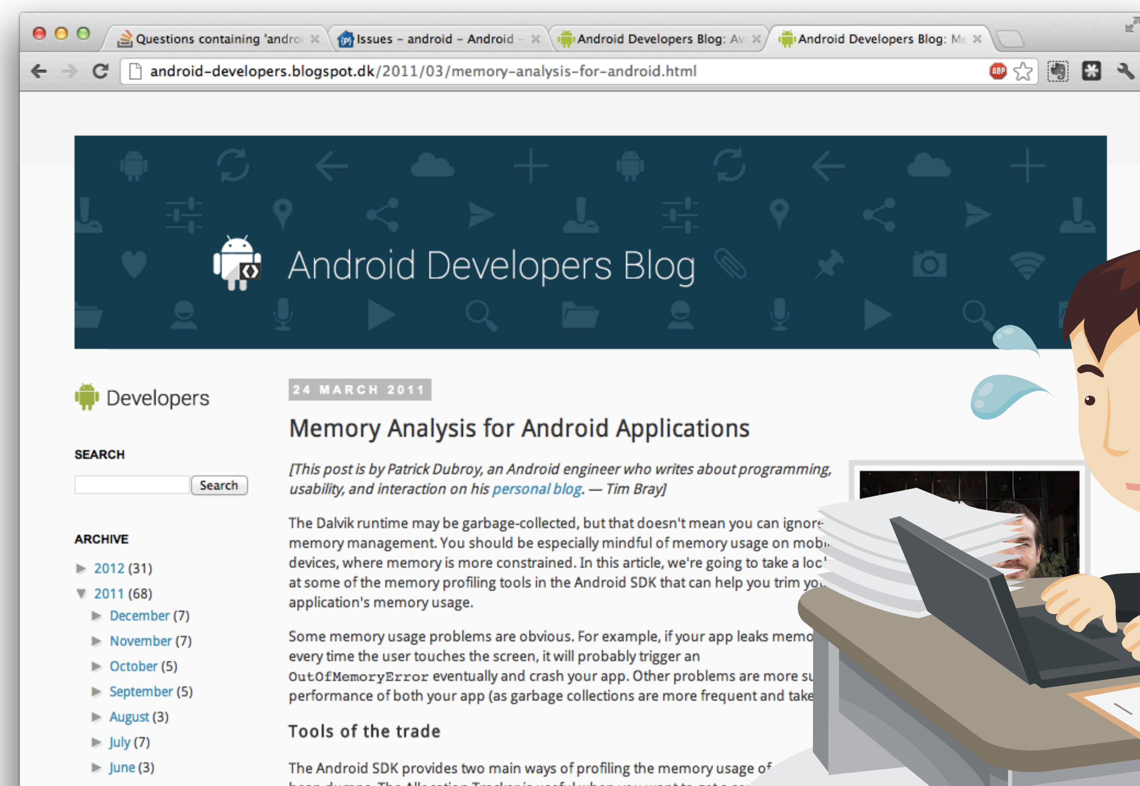
Elsewhere, following the state of practice for debugging leaks ...

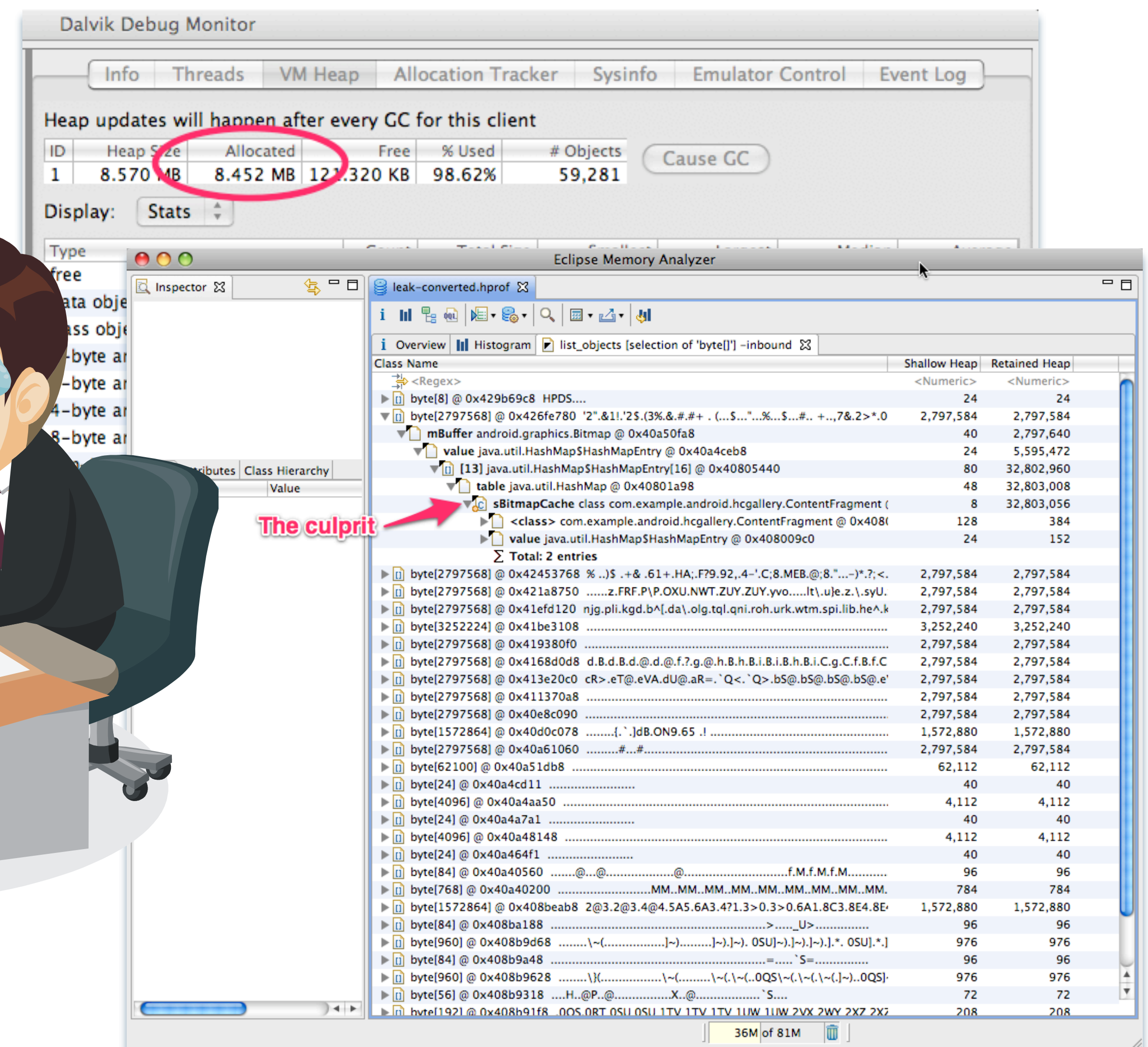


1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
4. Commit a bugfix

GitHub

Elsewhere, following the state of practice for debugging leaks ...





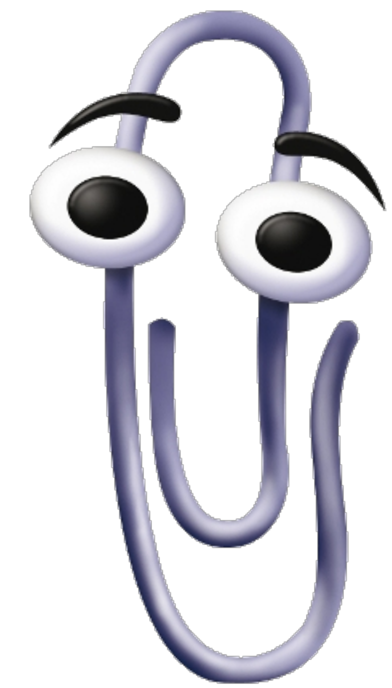
1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and finally find the culprit!
4. Commit a bugfix
5. Bugfix is picked up by **Fixr**

GitHub

Fixr

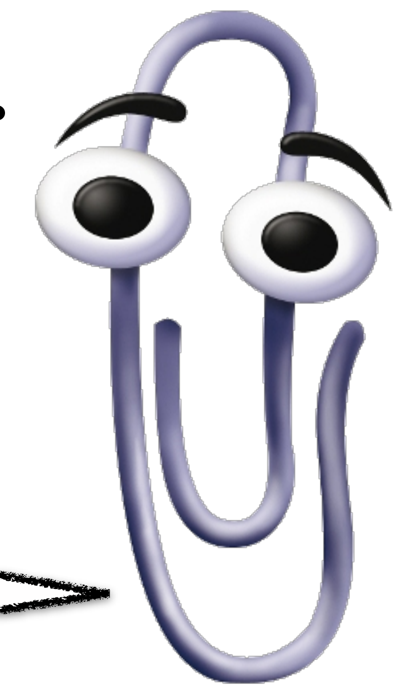
A **Fixr**-enabled IDE responds ...

A **Fixr**-enabled IDE responds ...



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

A **Fixr**-enabled IDE responds ...

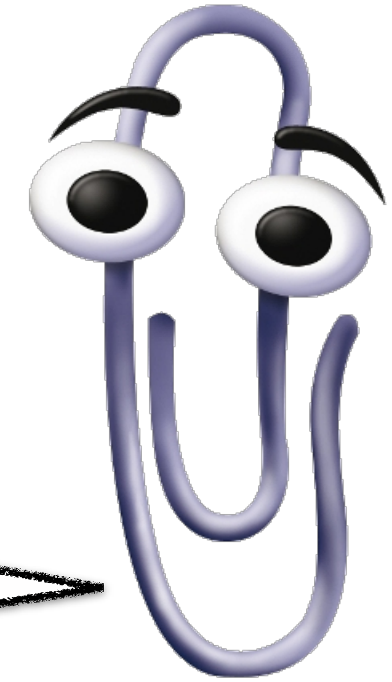


It looks like you've created a memory leak like  and 100,000 others. Would you like to apply 

?

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

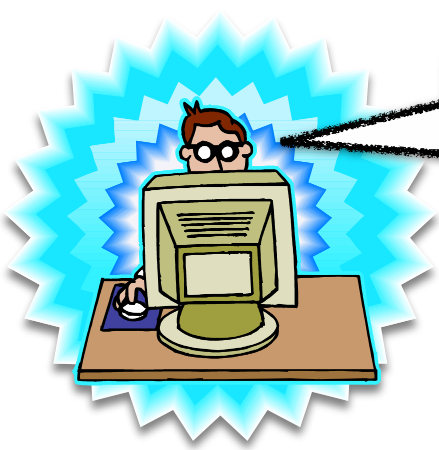
A **Fixr**-enabled IDE responds ...



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

?

the bugfix is "transferred"



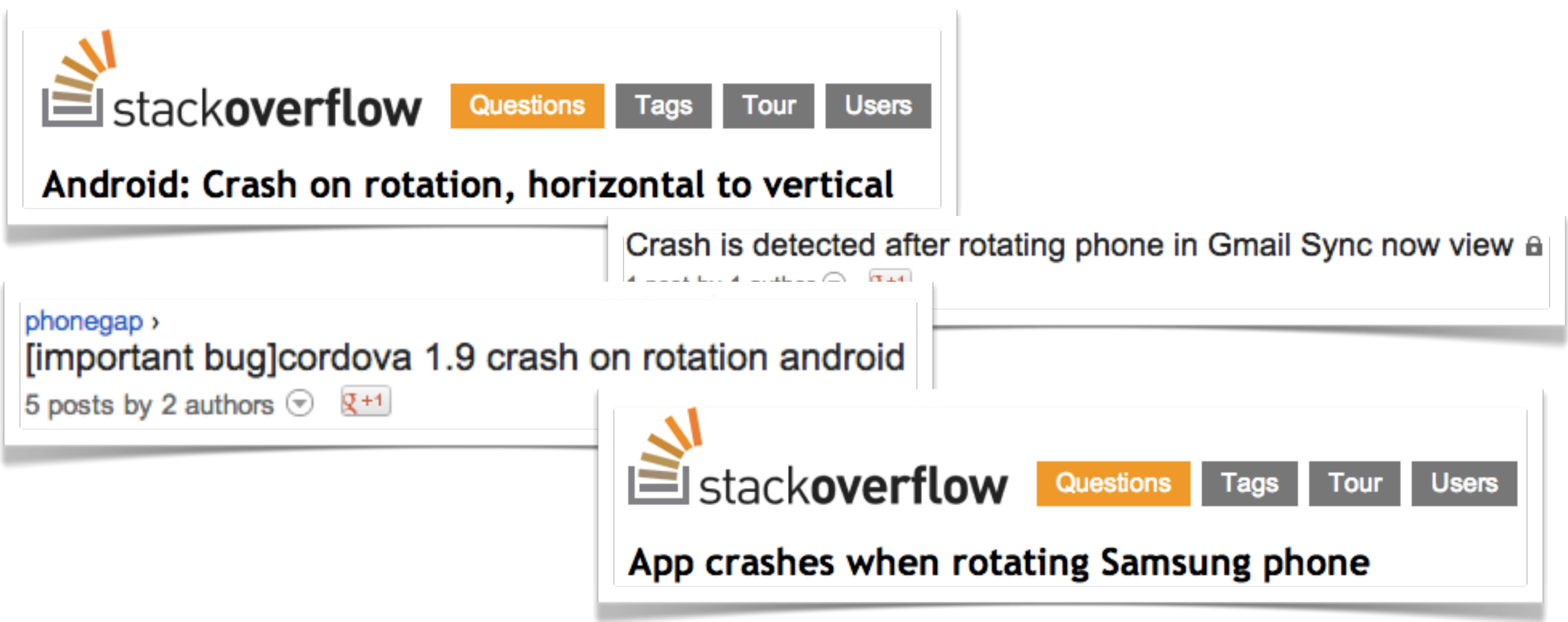
I don't know how I created a long-lived reference to an `Activity`!

One Sentence Summary: Mine specifications of framework rules (indirectly) from bugfixes

Leverage volume and variety of bugfixes made by the crowd of client app developers

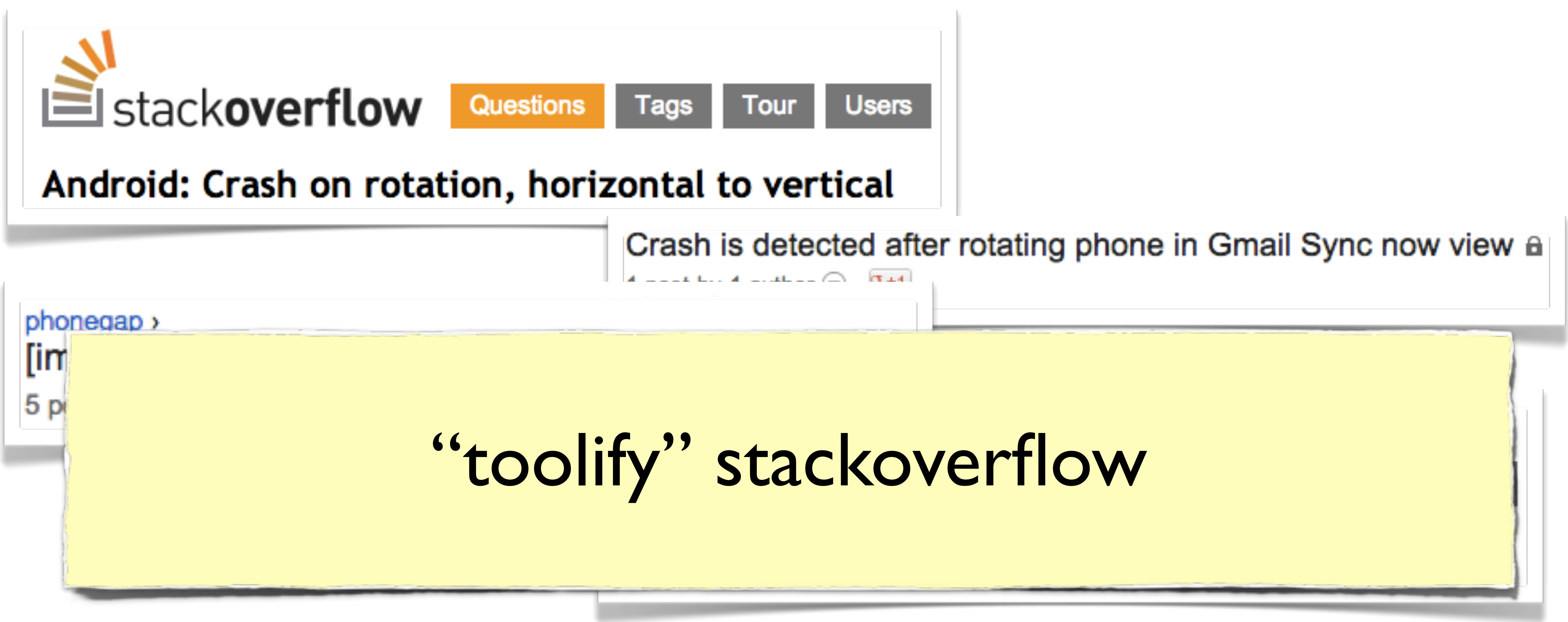
One Sentence Summary: Mine specifications of framework rules (indirectly) from bugfixes

Leverage volume and variety of bugfixes made by the crowd of client app developers



One Sentence Summary: Mine specifications of framework rules (indirectly) from bugfixes

Leverage volume and variety of bugfixes made by the crowd of client app developers



“toolify” stackoverflow

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)

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

~~aView.setTag(..., anObject)~~
aView.setTag(anObject)

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(..., anObject)~~

aView.setTag(anObject)

uses mTag instead

Bug
(on Android <4)

aView.setTag(..., anObject)

bug pre

if anObject can reach aView

Framework
Invariant

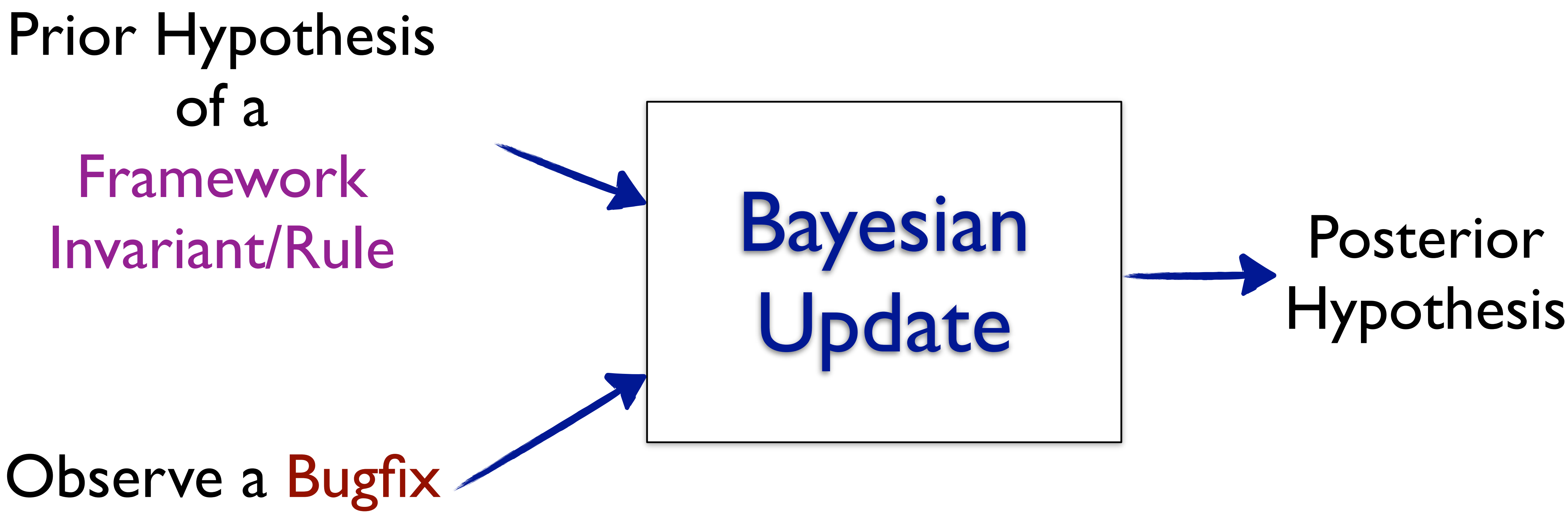
```
class View {
  static WeakHashMap<View, SparseArray<Object>> sTags;
  Object mTag;
}
```

invariant

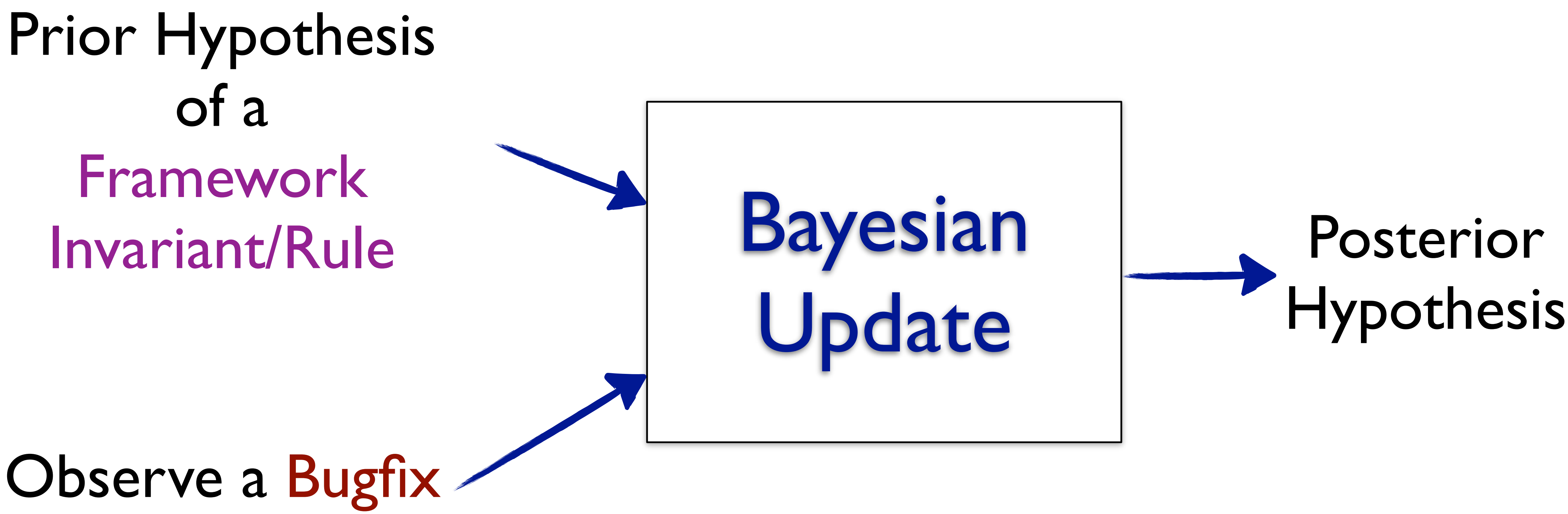
because of an unspecified class invariant: sTags' values (:Object) must not reach their keys (:View)

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

Mining framework specifications with bugfixes



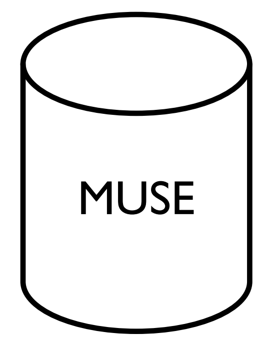
Mining framework specifications with bugfixes



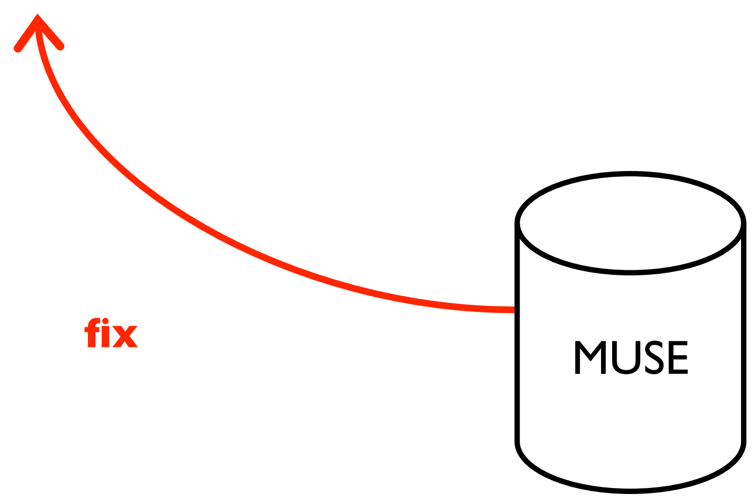
The **Fixr** Loop:
Create as many observations as possible

The **Fixr** Loop: Component by Component

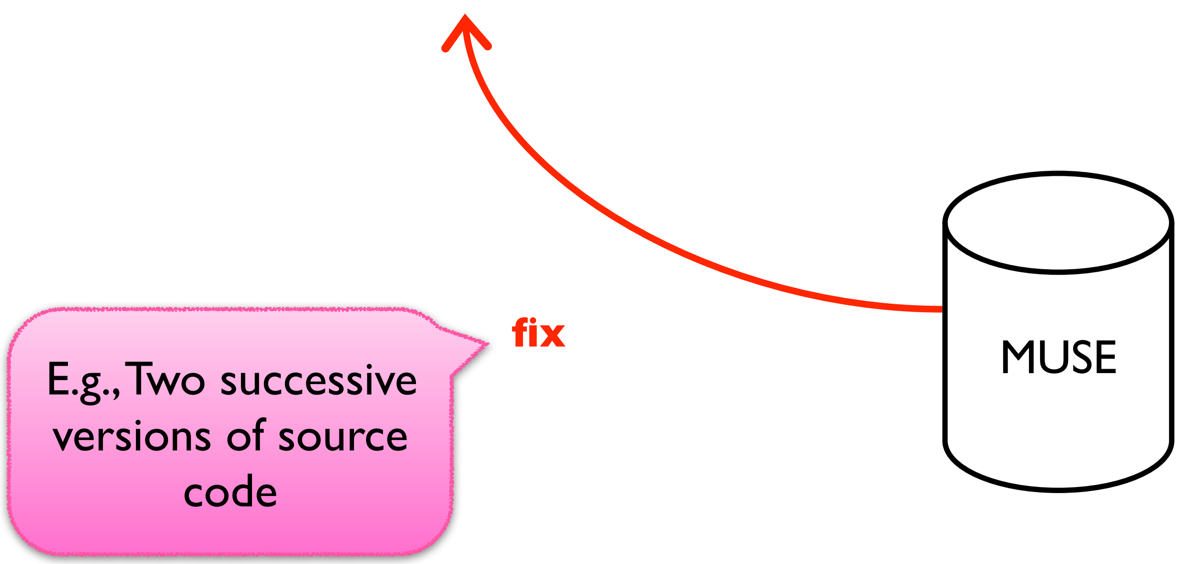
Fixr: Proposed System



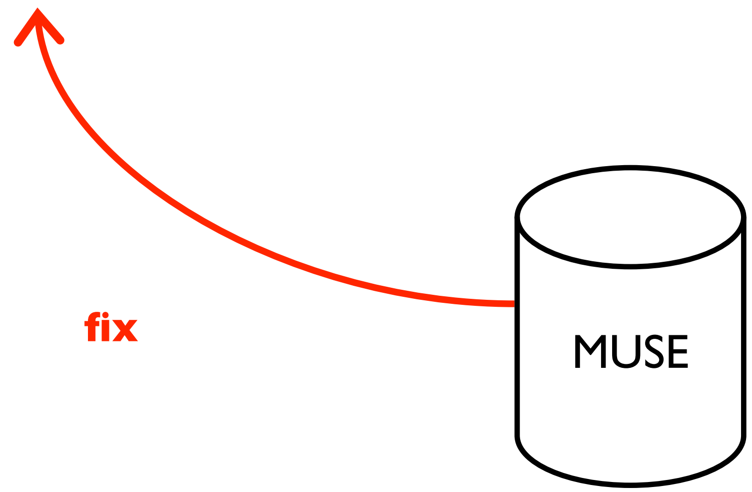
Fixr: Proposed System



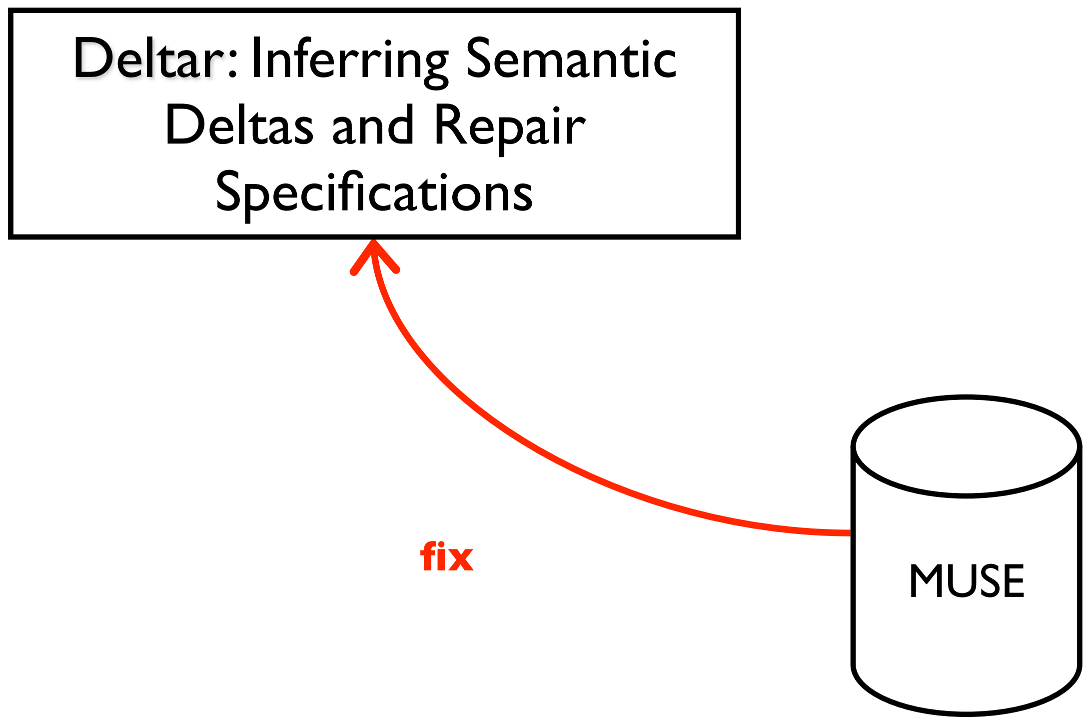
Fixr: Proposed System



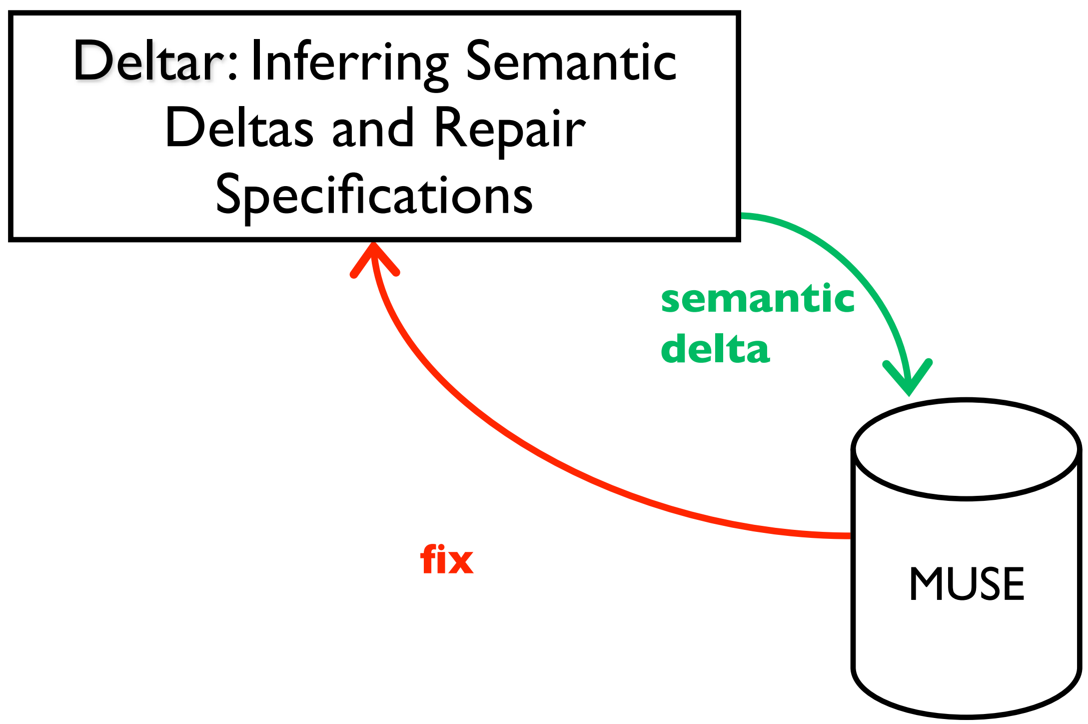
Fixr: Proposed System



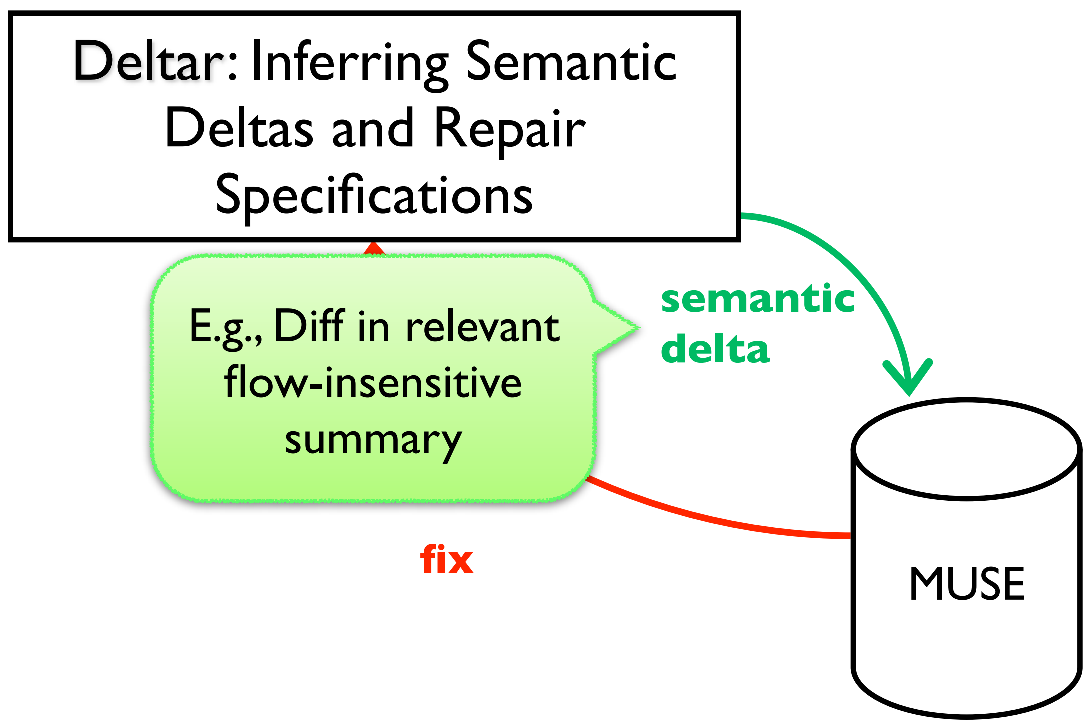
Fixr: Proposed System



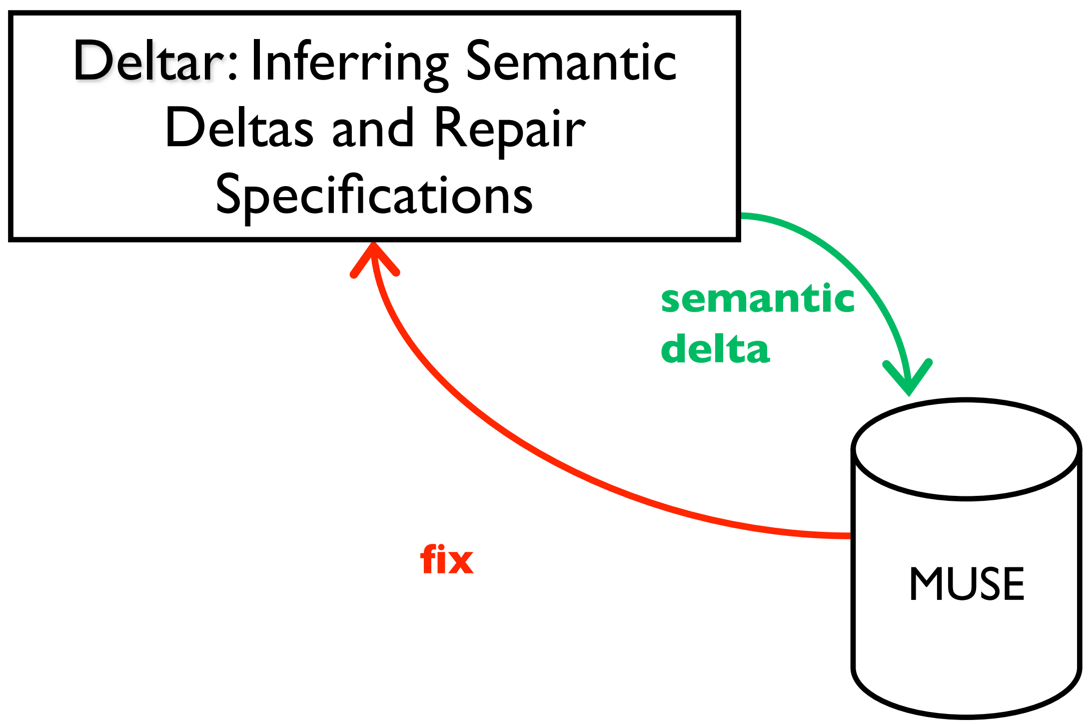
Fixr: Proposed System



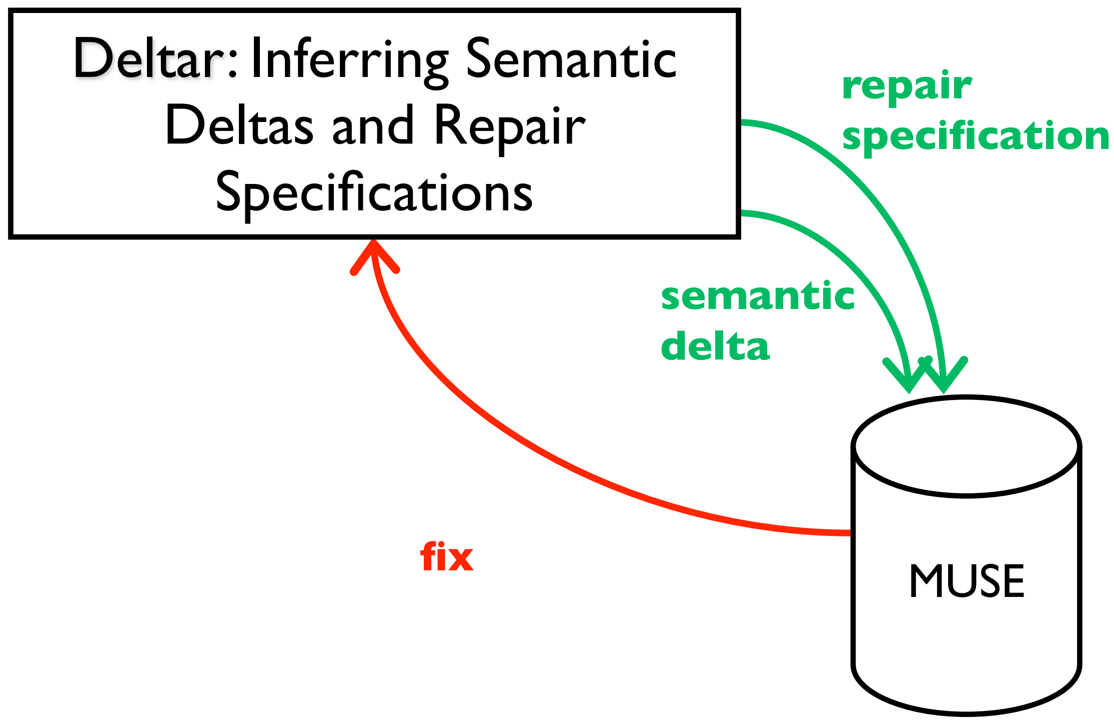
Fixr: Proposed System



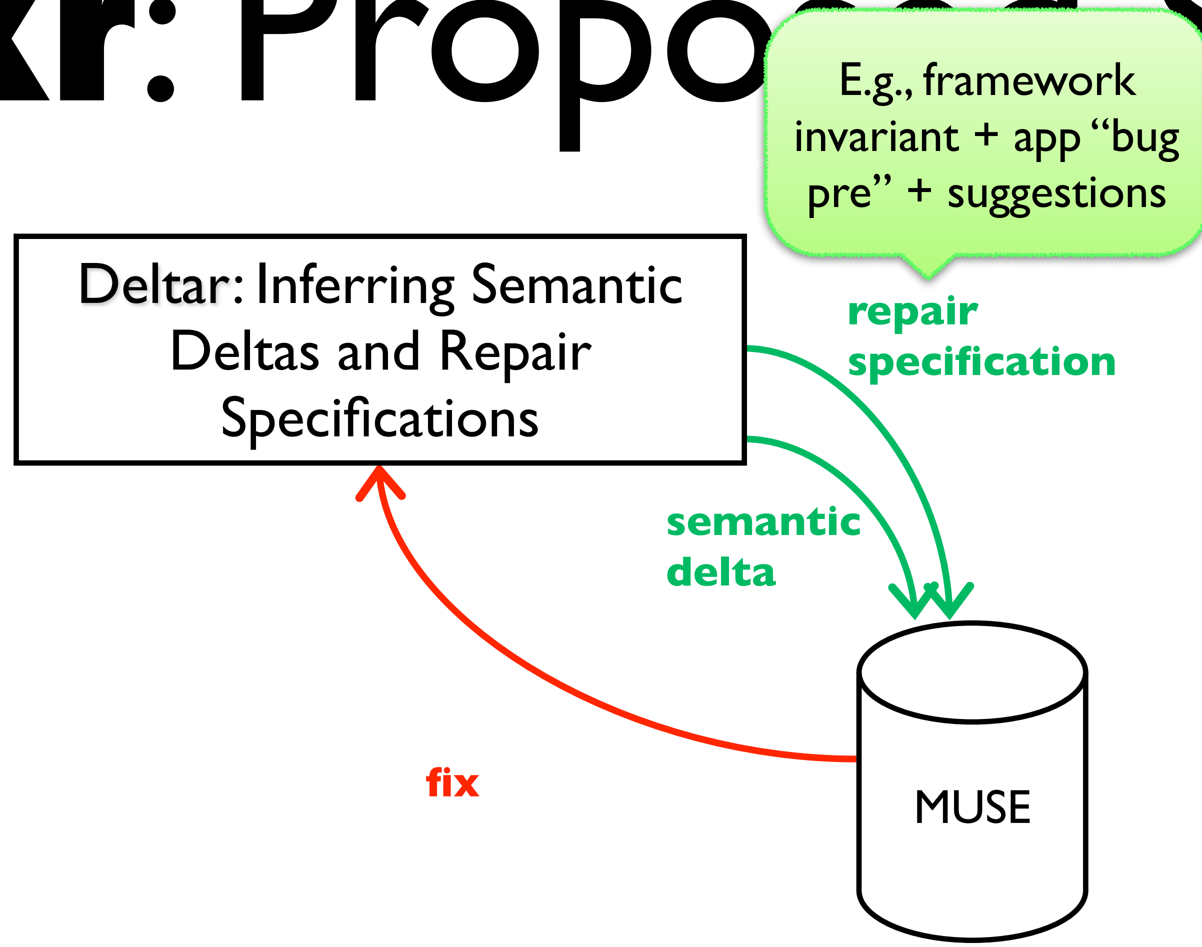
Fixr: Proposed System



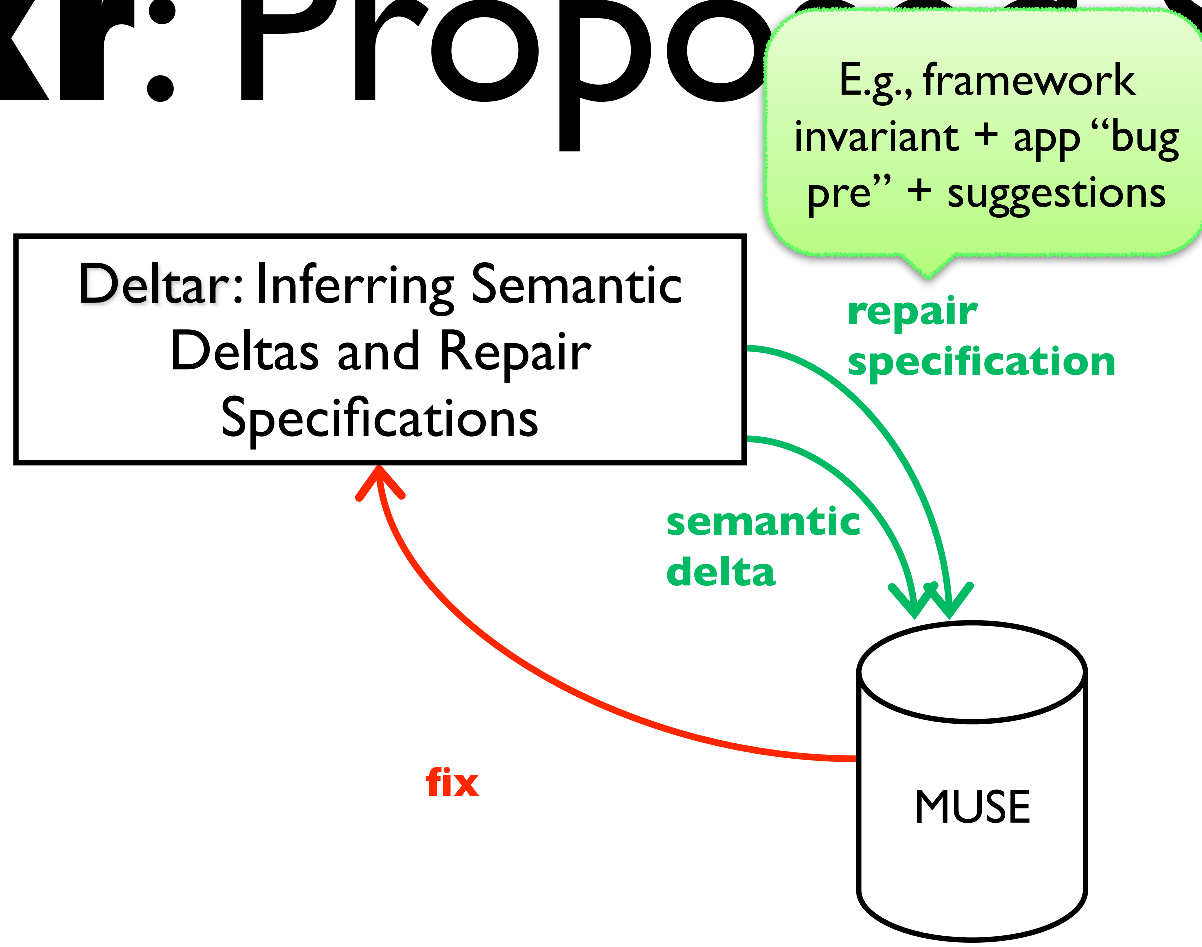
Fixr: Proposed System



Fixr: Proposed System



Fixr: Proposed System



Component: Deltar *maps* fixes to semantic difference summaries and candidate repair specifications

Deltar

Deltar

A Fix

~~aView.setTag(..., anObject)~~

aView.setTag(anObject)

Deltar

A Fix

~~aView.setTag(..., anObject)~~

aView.setTag(anObject)

Problem: Need to mine and check candidate framework invariants

Deltar

A Fix

~~aView.setTag(..., anObject)~~

aView.setTag(anObject)

Problem: Need to mine and check candidate framework invariants

Delta

~~WeakHashMap$Entry → MyView~~

Deltar

A Fix

~~aView.setTag(..., anObject)~~

aView.setTag(anObject)

Problem: Need to mine and check candidate framework invariants

Delta

~~WeakHashMap$Entry → MyView~~

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null**

Deltar

A Fix

~~aView.setTag(..., anObject)~~

aView.setTag(anObject)

Problem: Need to mine and check candidate framework invariants

Delta

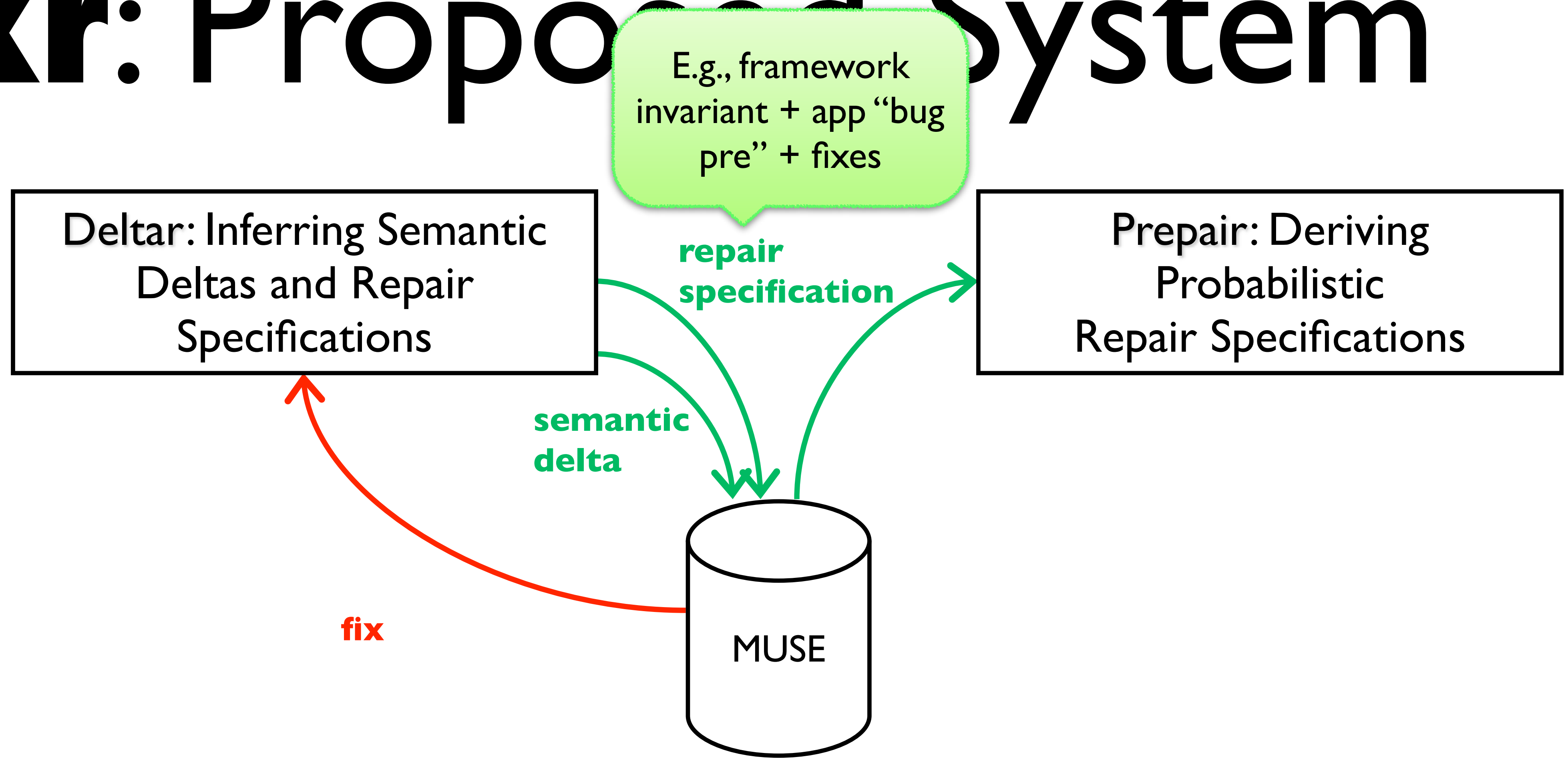
~~WeakHashMap$Entry → MyView~~

Candidate Invariant

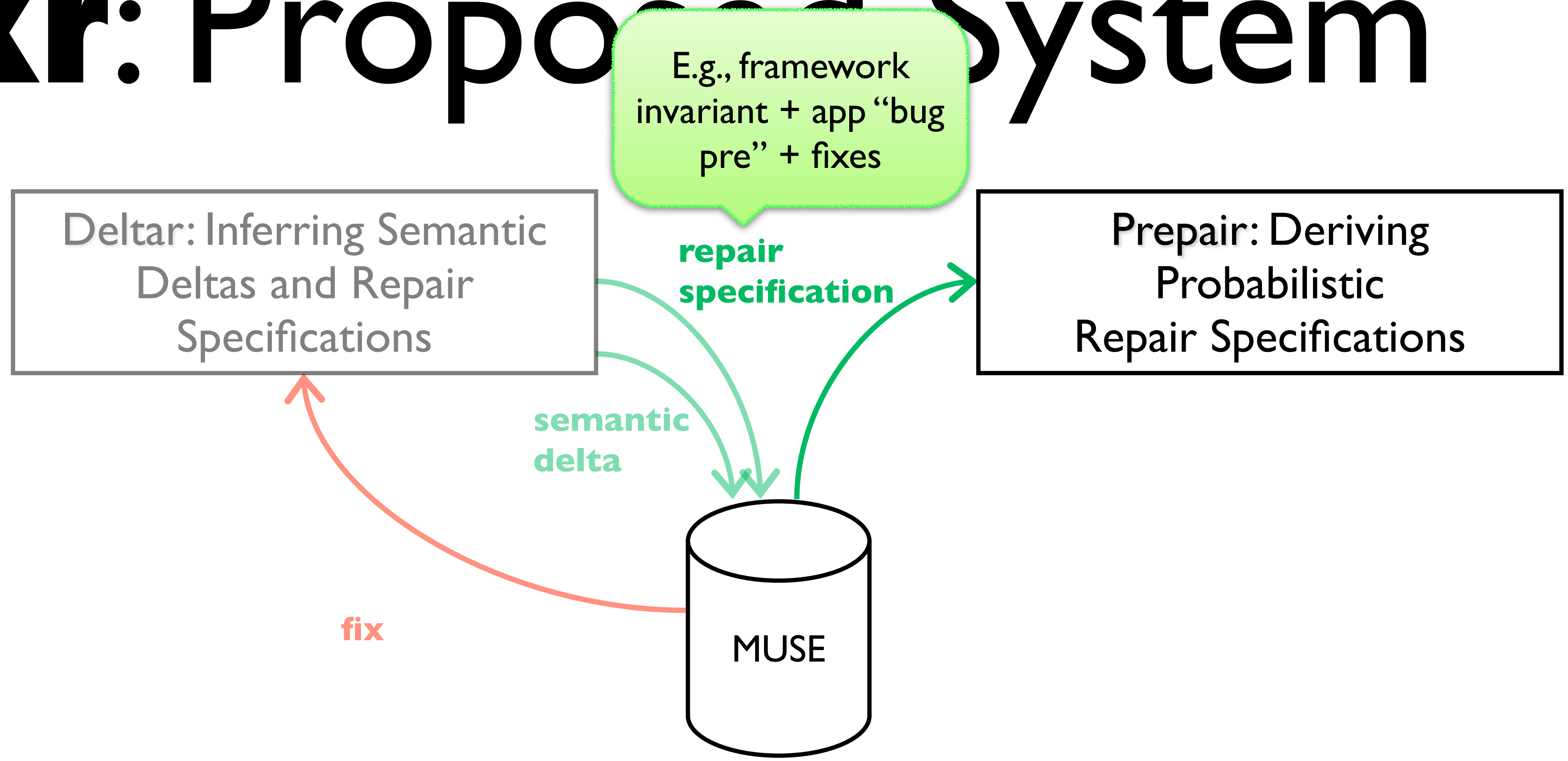
sTags == **null** ∧ mTag != **null**

Approach: Refine coarse, global summaries and verify candidate invariant on fixed version (scalably with "almost everywhere type analysis")

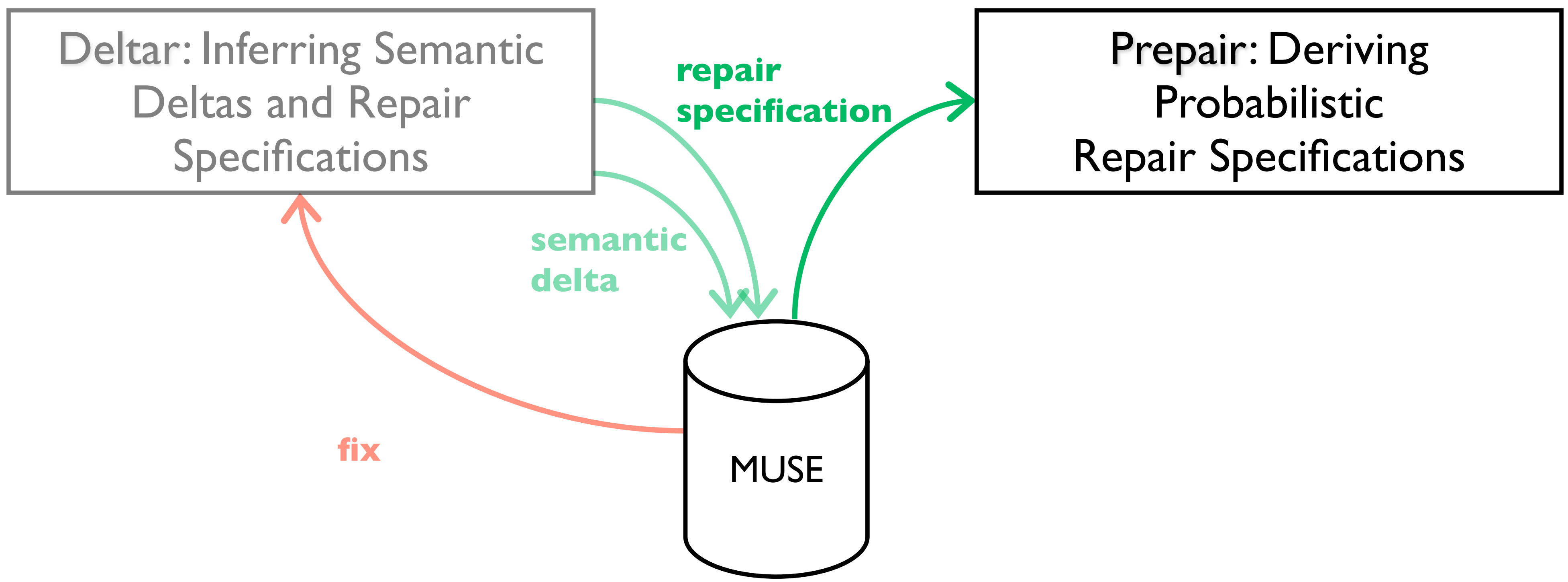
Fixr: Proposed System



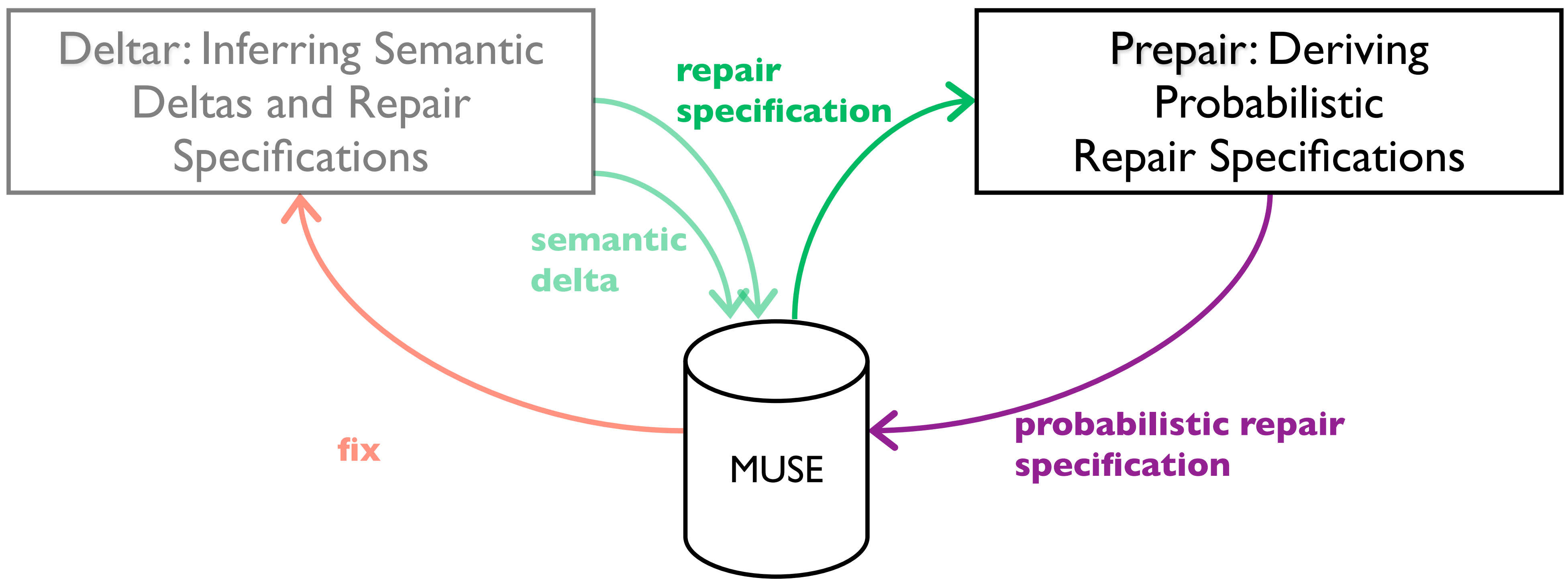
Fixr: Proposed System



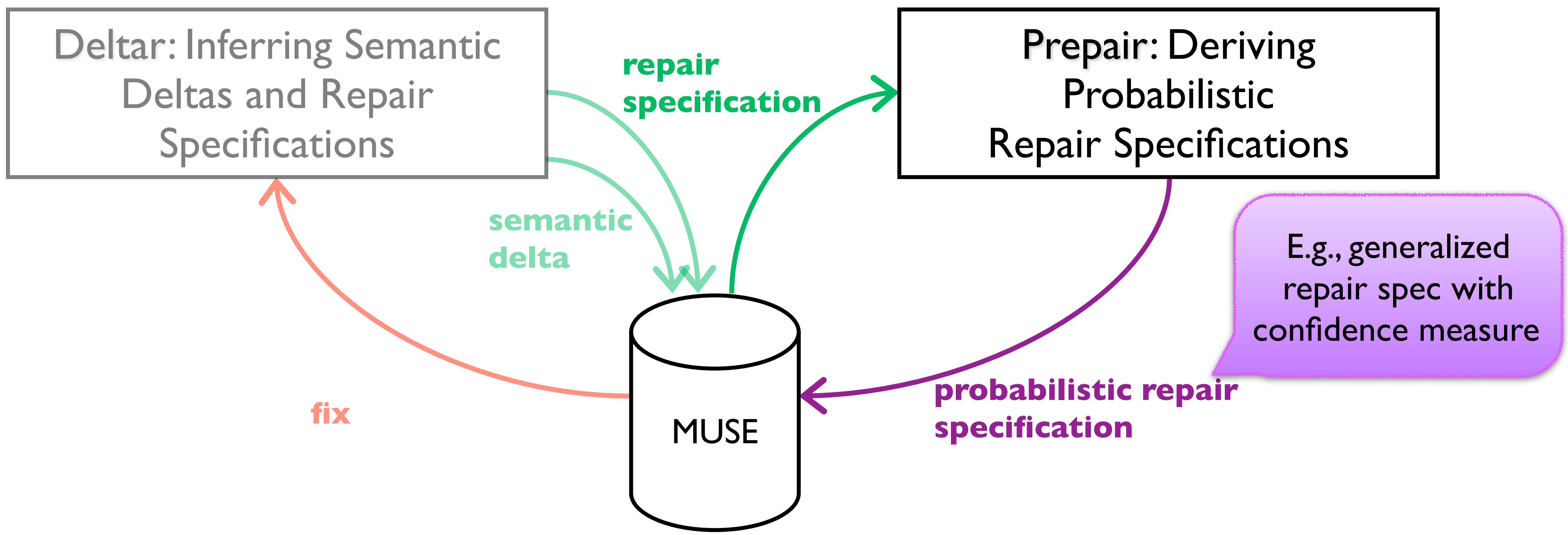
Fixr: Proposed System



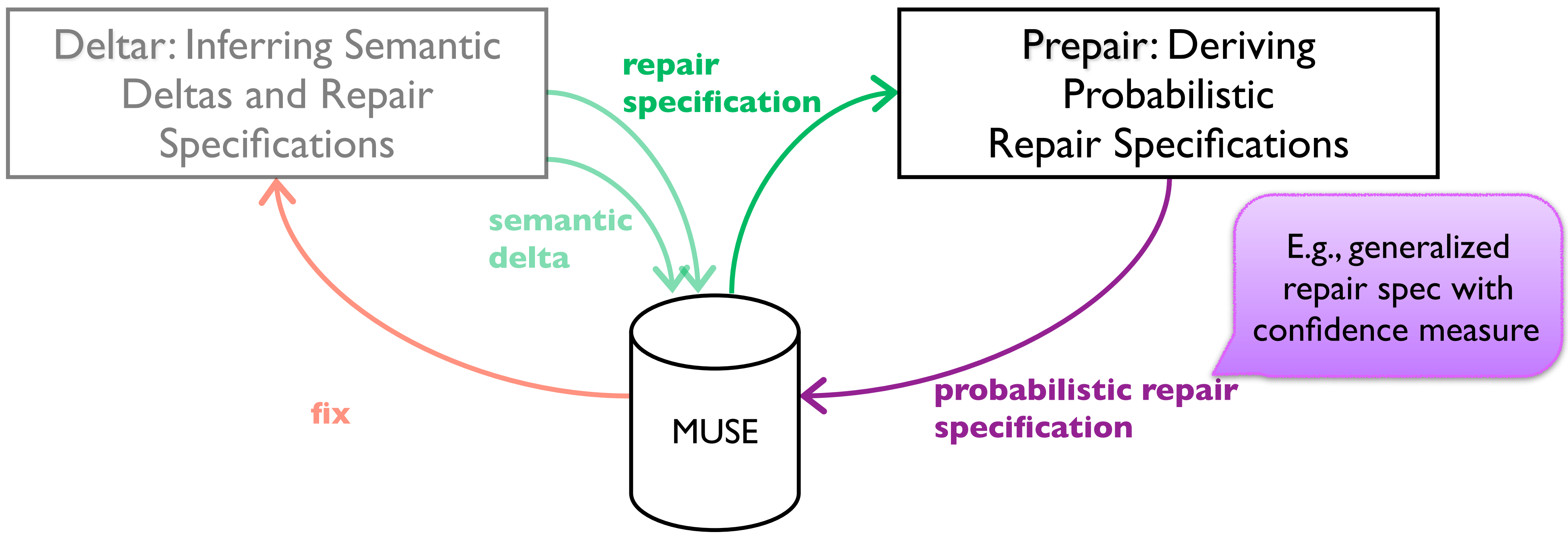
Fixr: Proposed System



Fixr: Proposed System



Fixr: Proposed System



Component: Prepair *reduces* candidate repair specifications to generalized probabilistic repair specifications

Prepair

Prepair

Candidate Invariant

sTags == **null** ∧ mTag != **null**

Prepair

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null**

$\forall i.$ sTags$[v][i] \nrightarrow v$

Prepair

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null**

$\forall i$. sTags[v][i] $\not\rightarrow$ v

$\forall v$. sTags[v][0] $\not\rightarrow^*$ v

Prepair

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null**

$\forall i$. sTags[v][i] $\nrightarrow v$

$\forall v$. sTags[v][0] $\nrightarrow^{*} v$

Problem: Multiple (overly-specific or under-specified) candidate repair specifications

Prepair

Candidate Invariant

sTags == **null** ∧ mTag != **null**

$\forall i.$ sTags$[v][i] \nrightarrow v$

$\forall v.$ sTags$[v][0] \nrightarrow^{*} v$

Problem: Multiple (overly-specific or under-specified) candidate repair specifications

Approach: Static analysis as a form of Bayesian updating of priors to derive posteriors. Prevalence of fixes in MUSE database provides priors.

Prepair

Candidate Invariant

$\text{sTags} == \textbf{null} \wedge \text{mTag} \mathrel{!=} \textbf{null}$ *0.9*

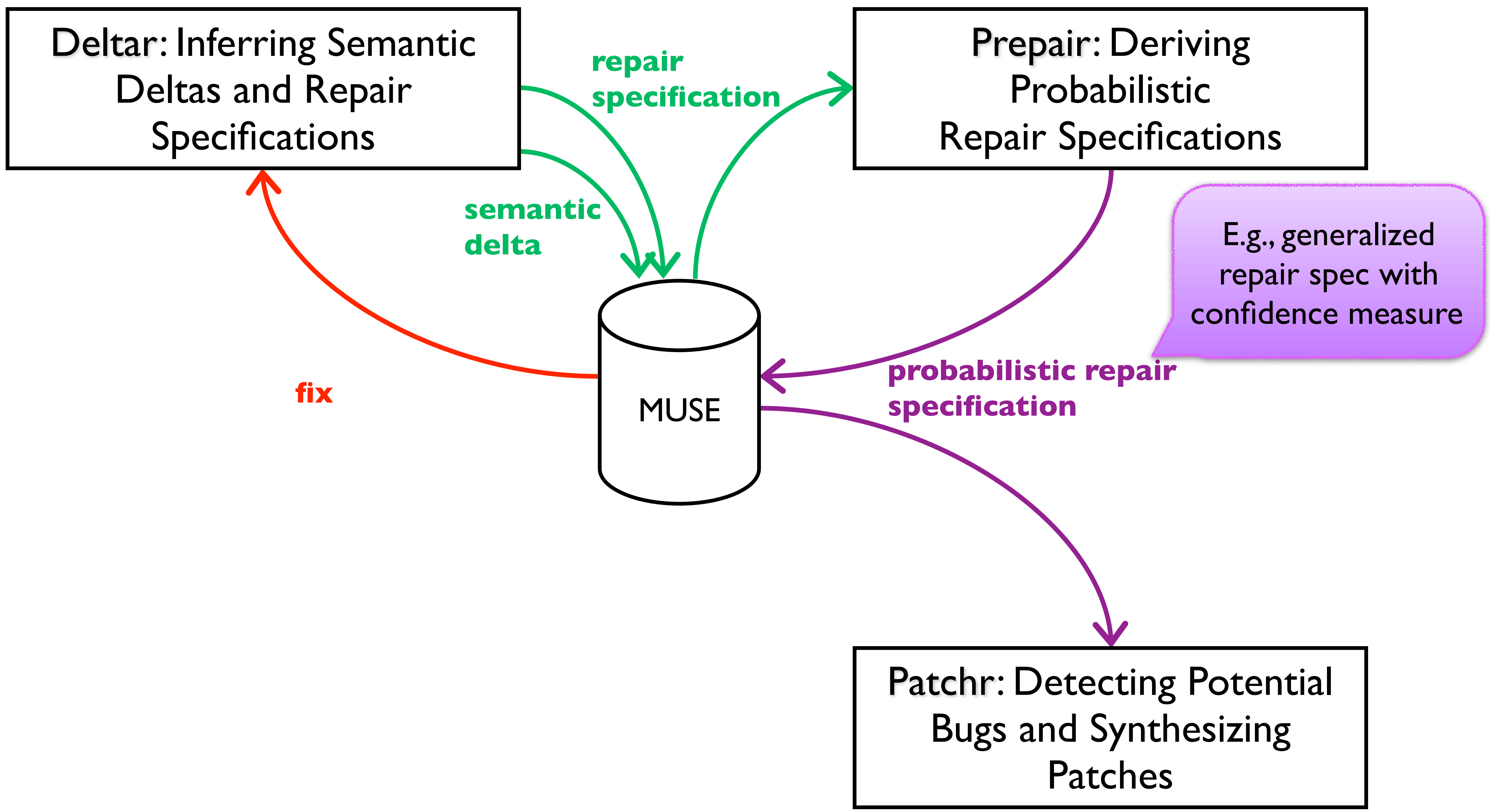
$\forall i.\ \text{sTags}[v][i] \nrightarrow v$ $\sqcup$

$\forall v.\ \text{sTags}[v][0] \nrightarrow^{*} v$

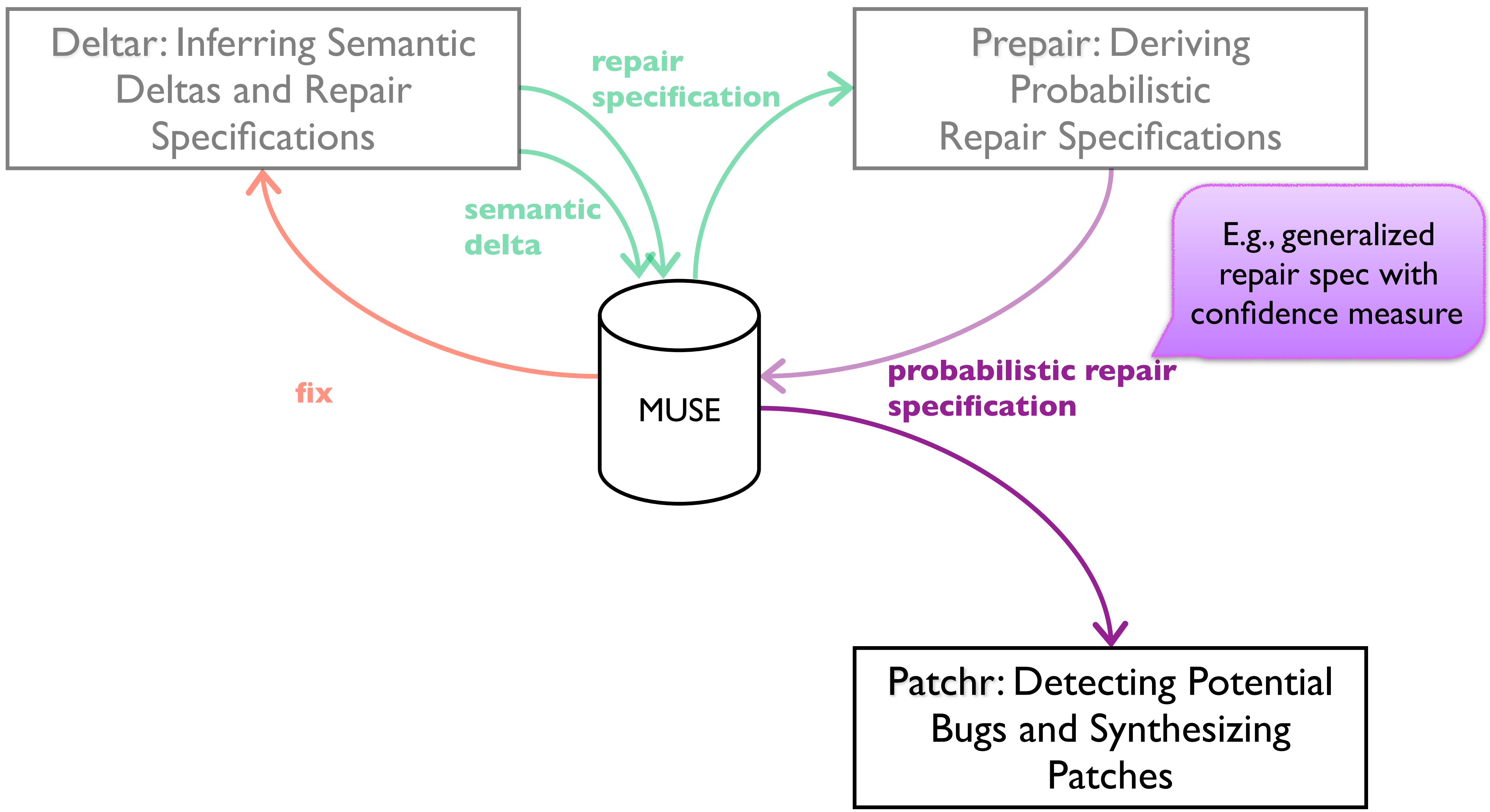
Problem: Multiple (overly-specific or under-specified) candidate repair specifications

Approach: Static analysis as a form of Bayesian updating of priors to derive posteriors. Prevalence of fixes in MUSE database provides priors.

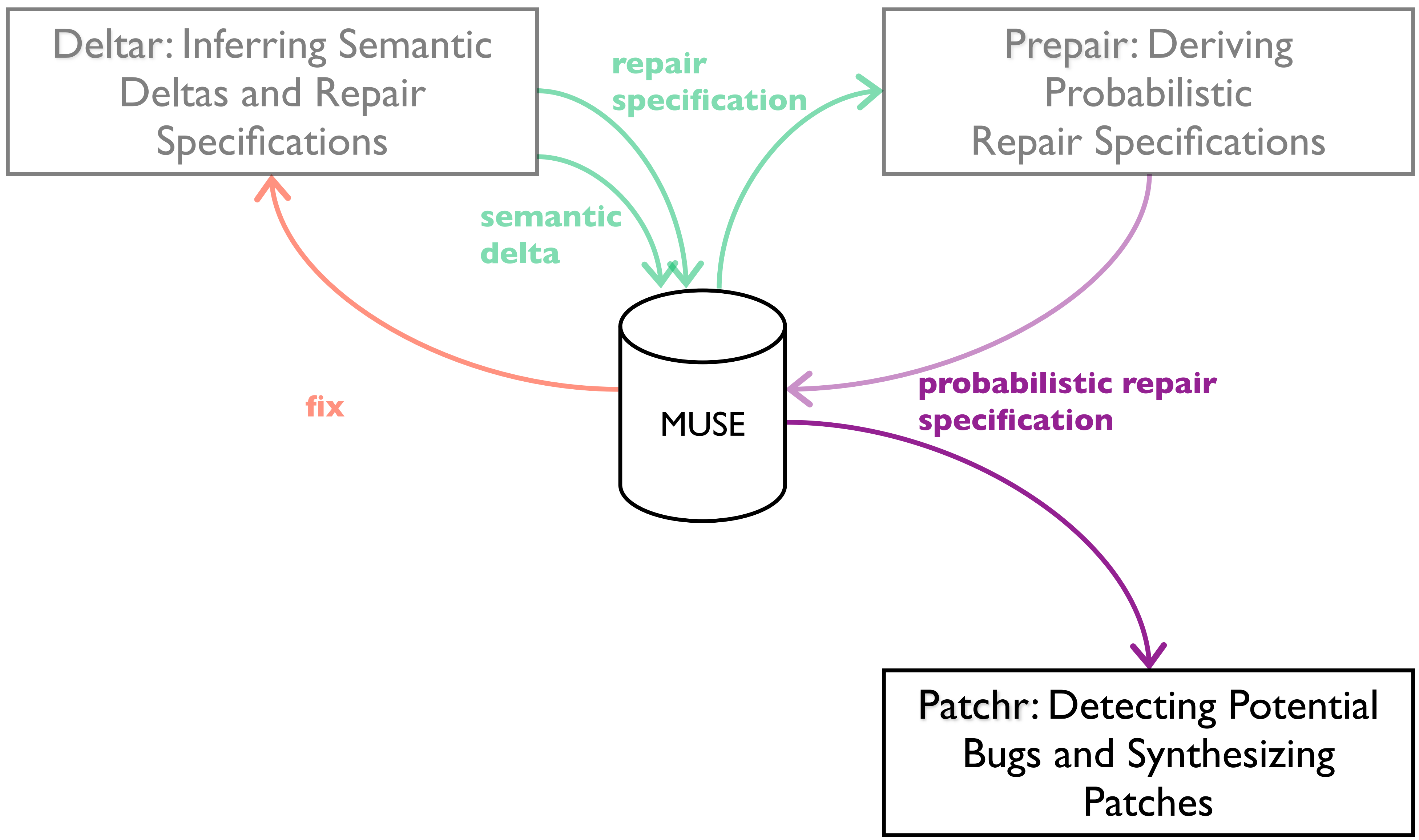
Fixr: Proposed System



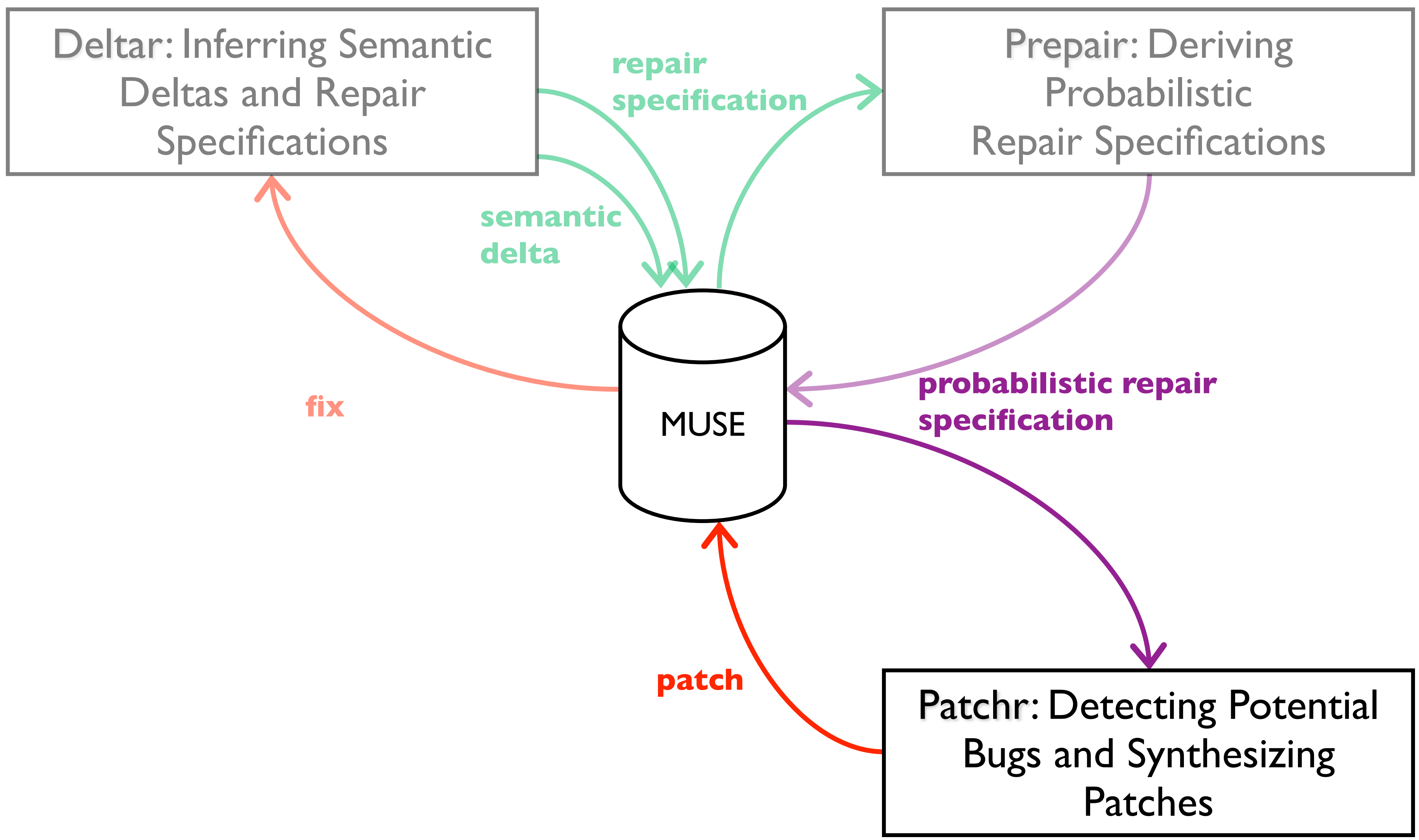
Fixr: Proposed System



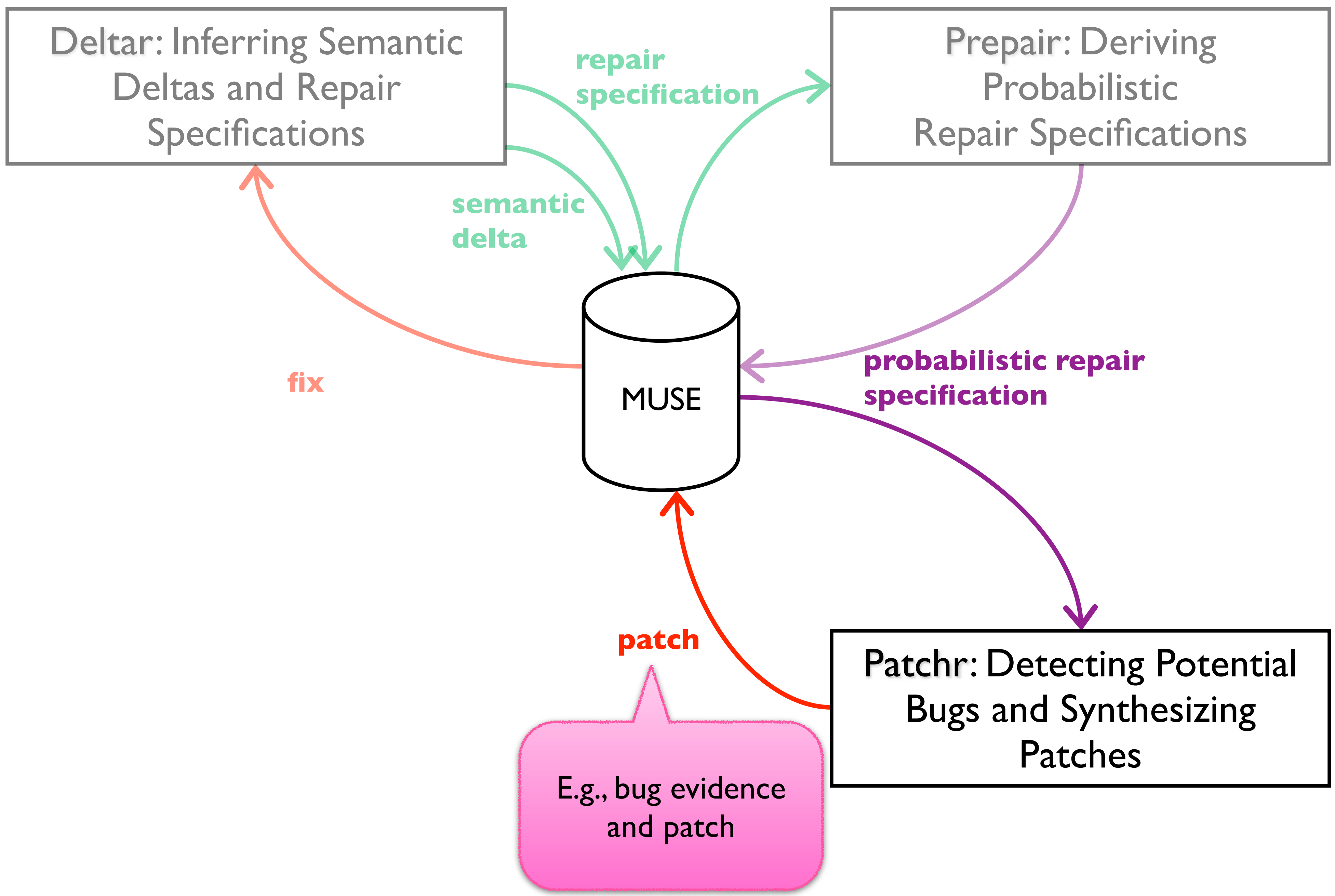
Fixr: Proposed System



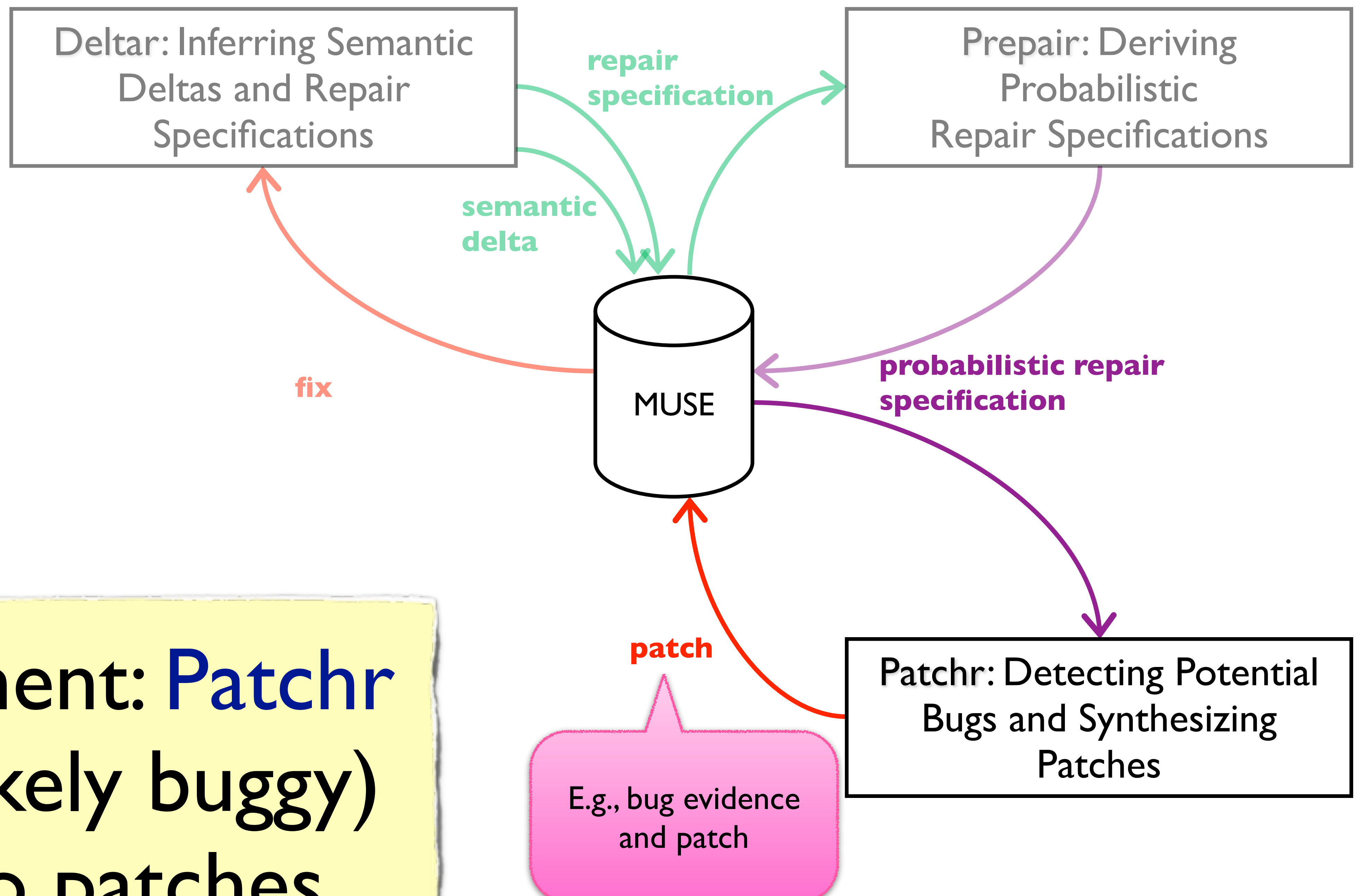
Fixr: Proposed System



Fixr: Proposed System



Fixr: Proposed System



Component: Patchr *maps* (likely buggy) apps to patches

Patchr

Patchr

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null** *0.9*

Patchr

Candidate Invariant

sTags == **null** ∧ mTag != **null** *0.9*

Problem: How do we validate repair specifications?

Patchr

Candidate Invariant

sTags == **null** ∧ mTag != **null**

0.9

Problem: How do we validate repair specifications?

Approach: Synthesize patches for human validation (easier to understand and immediately useful)

Patchr

Candidate Invariant

sTags == **null** ∧ mTag != **null**

0.9

Problem: How do we validate repair specifications?

Approach: Synthesize patches for human validation (easier to understand and immediately useful)

A Patch

~~otherView.setTag(..., o)~~

otherView.setTag(o)

Patchr

Candidate Invariant

sTags == **null** $\wedge$ mTag != **null** *0.9*

Problem: How do we validate repair specifications?

Approach: Synthesize patches for human validation (easier to understand and immediately useful)

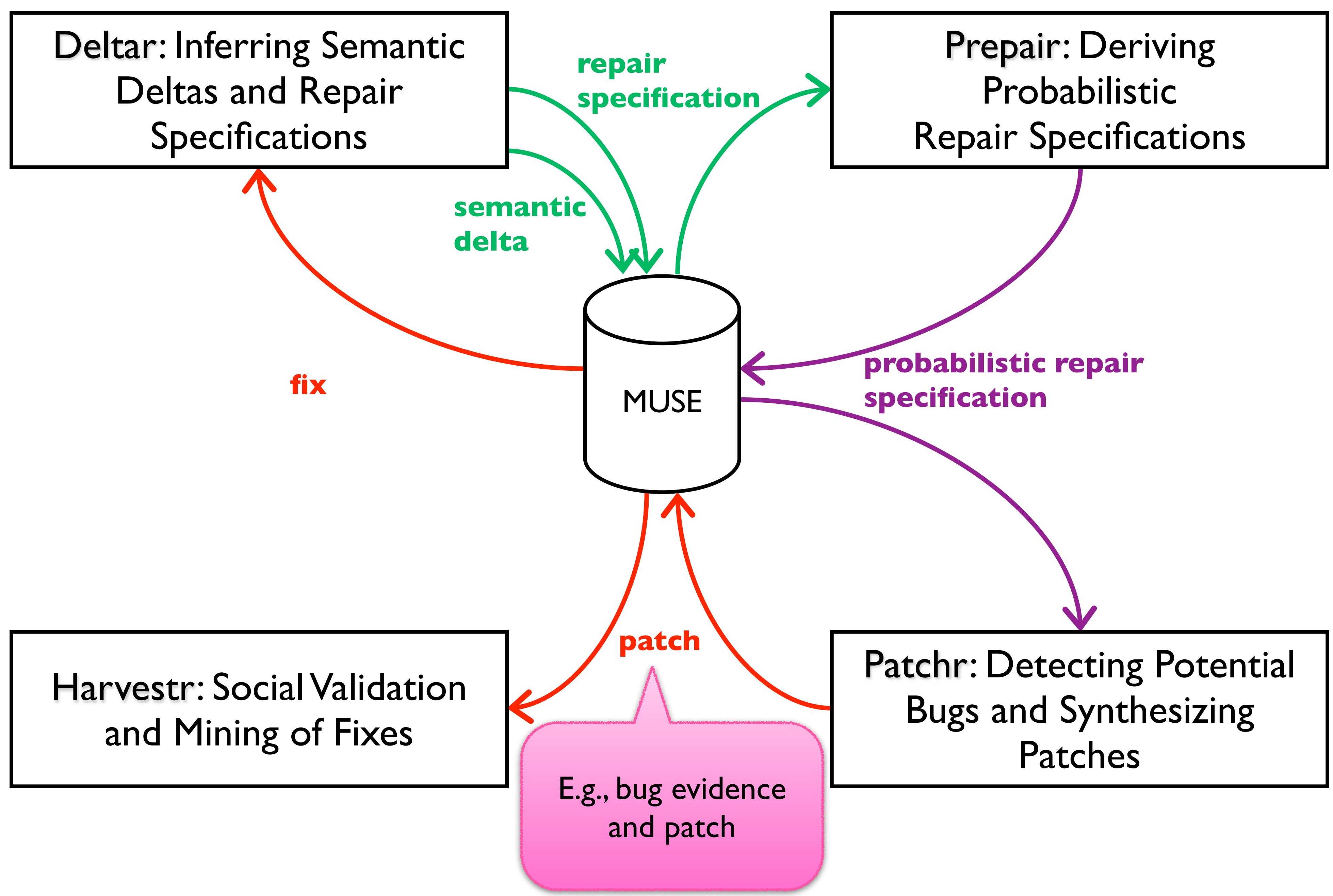
A Patch

~~otherView.setTag(..., o)~~

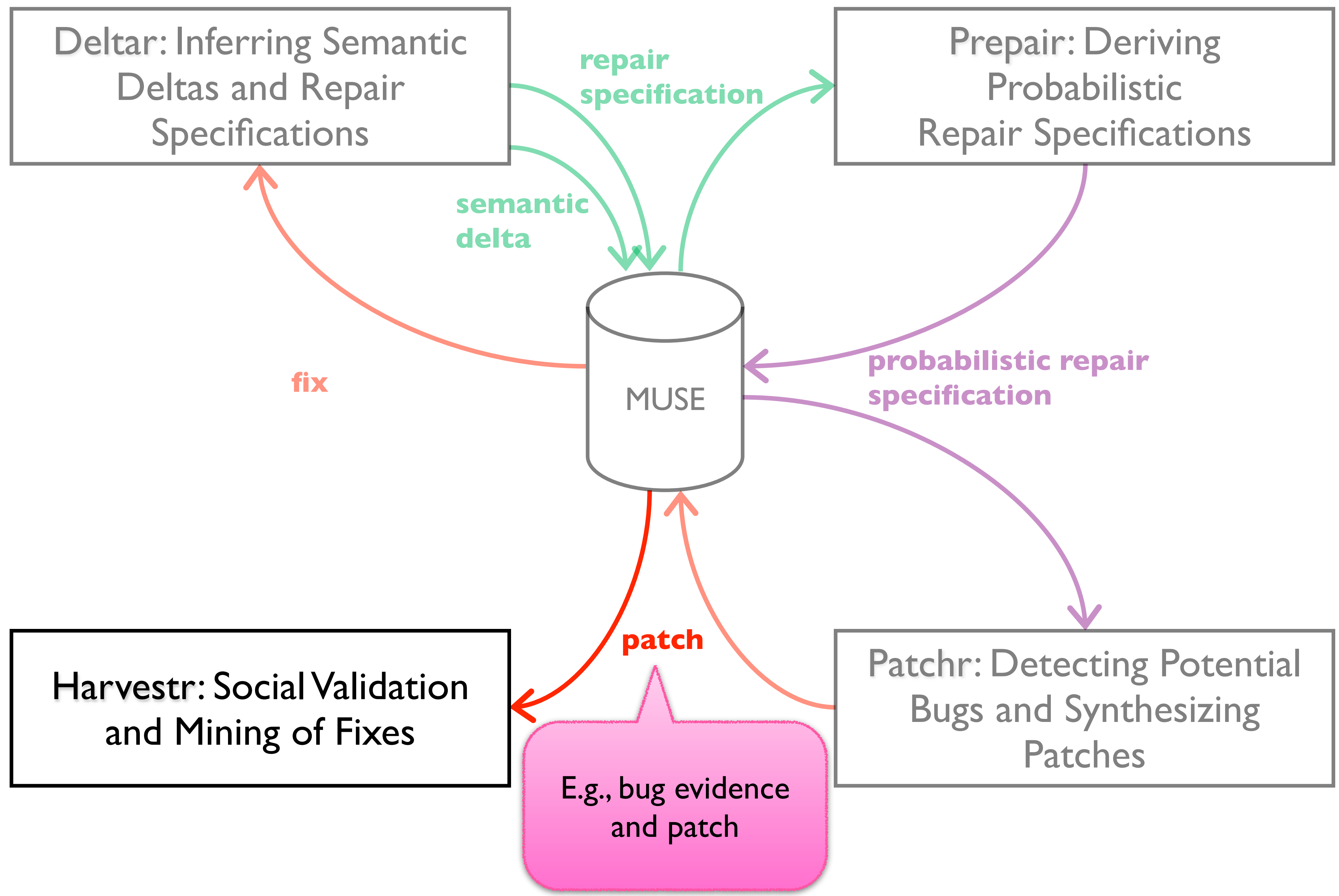
otherView.setTag(o)

need to find apps satisfying “bug pre”

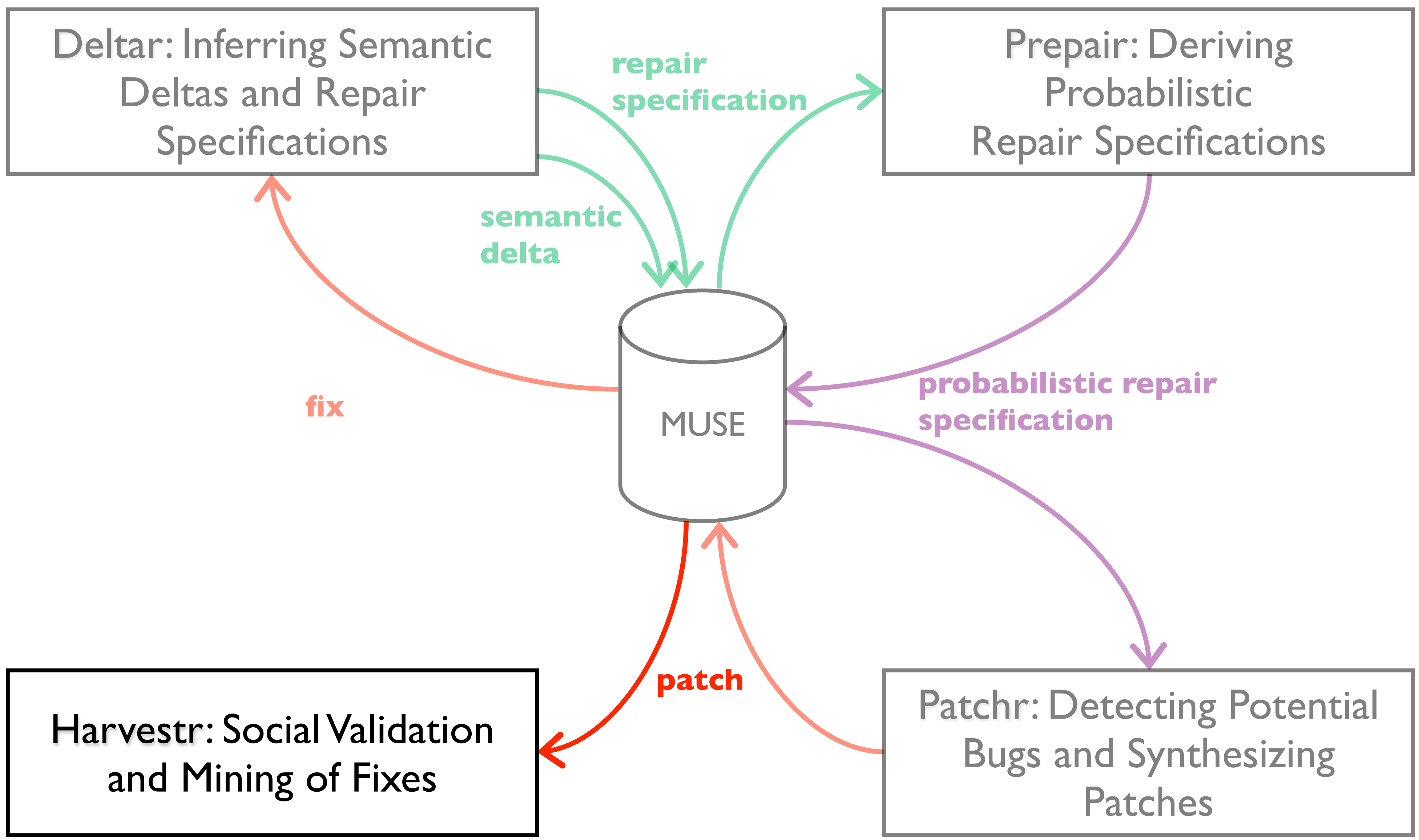
Fixr: Proposed System



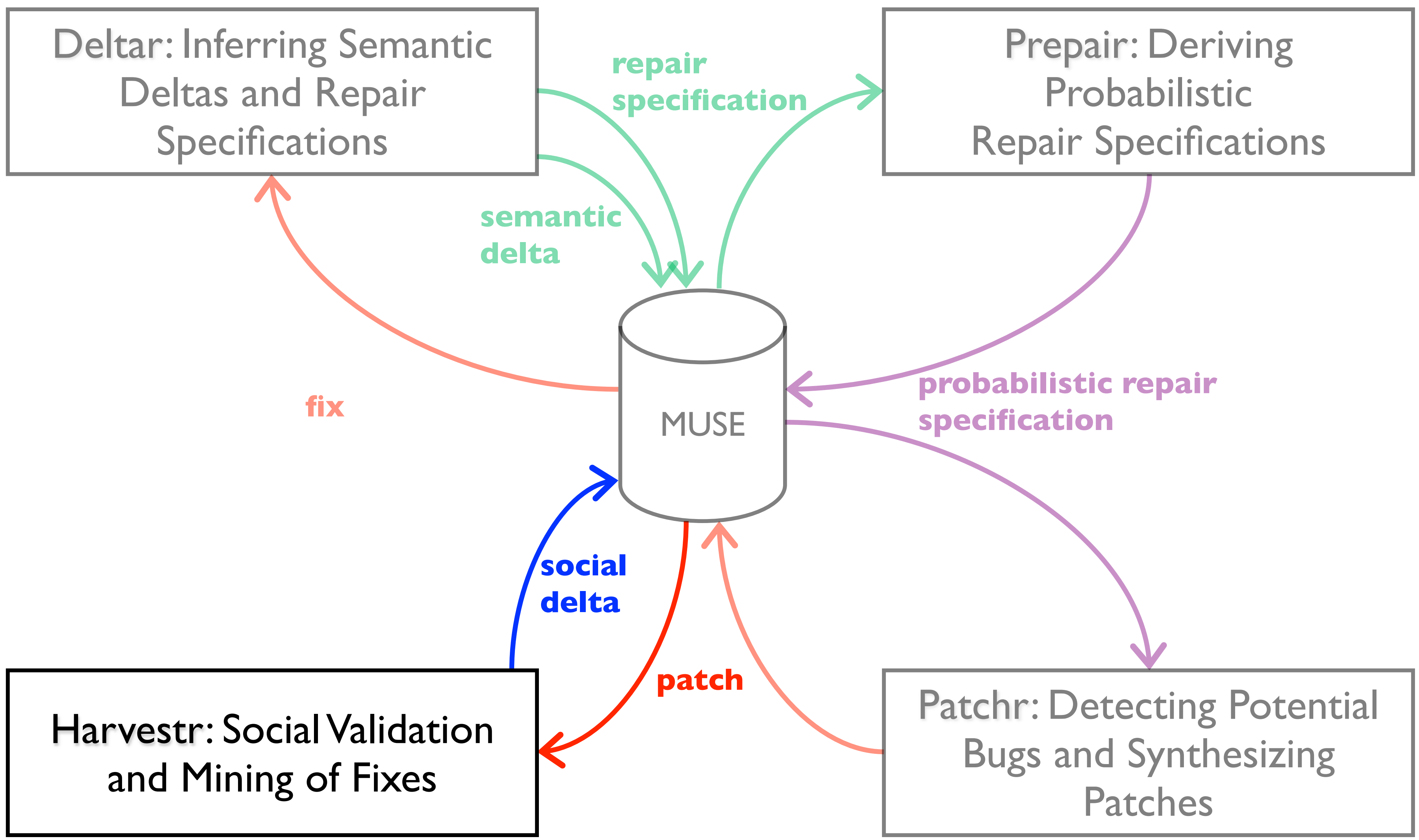
Fixr: Proposed System



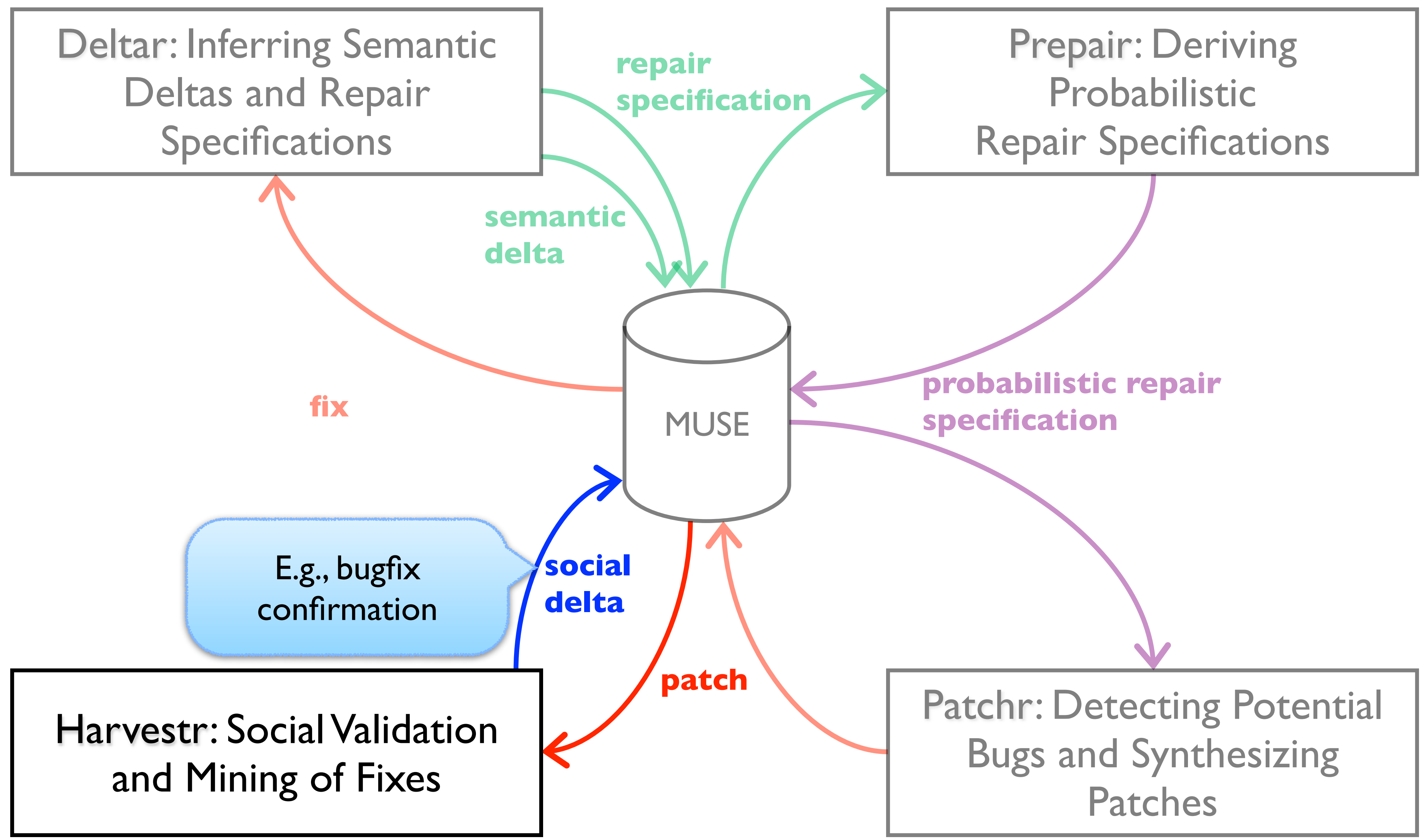
Fixr: Proposed System



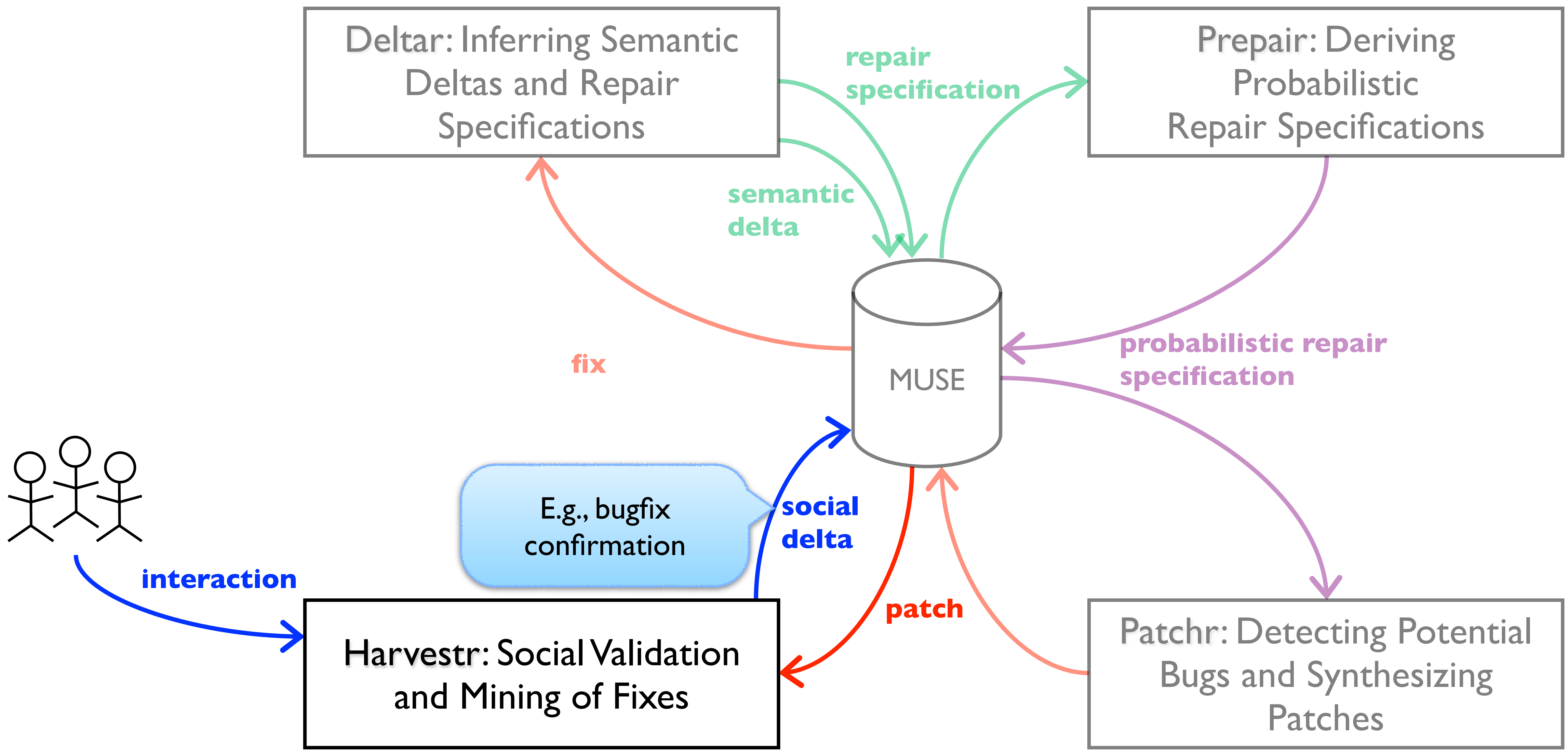
Fixr: Proposed System



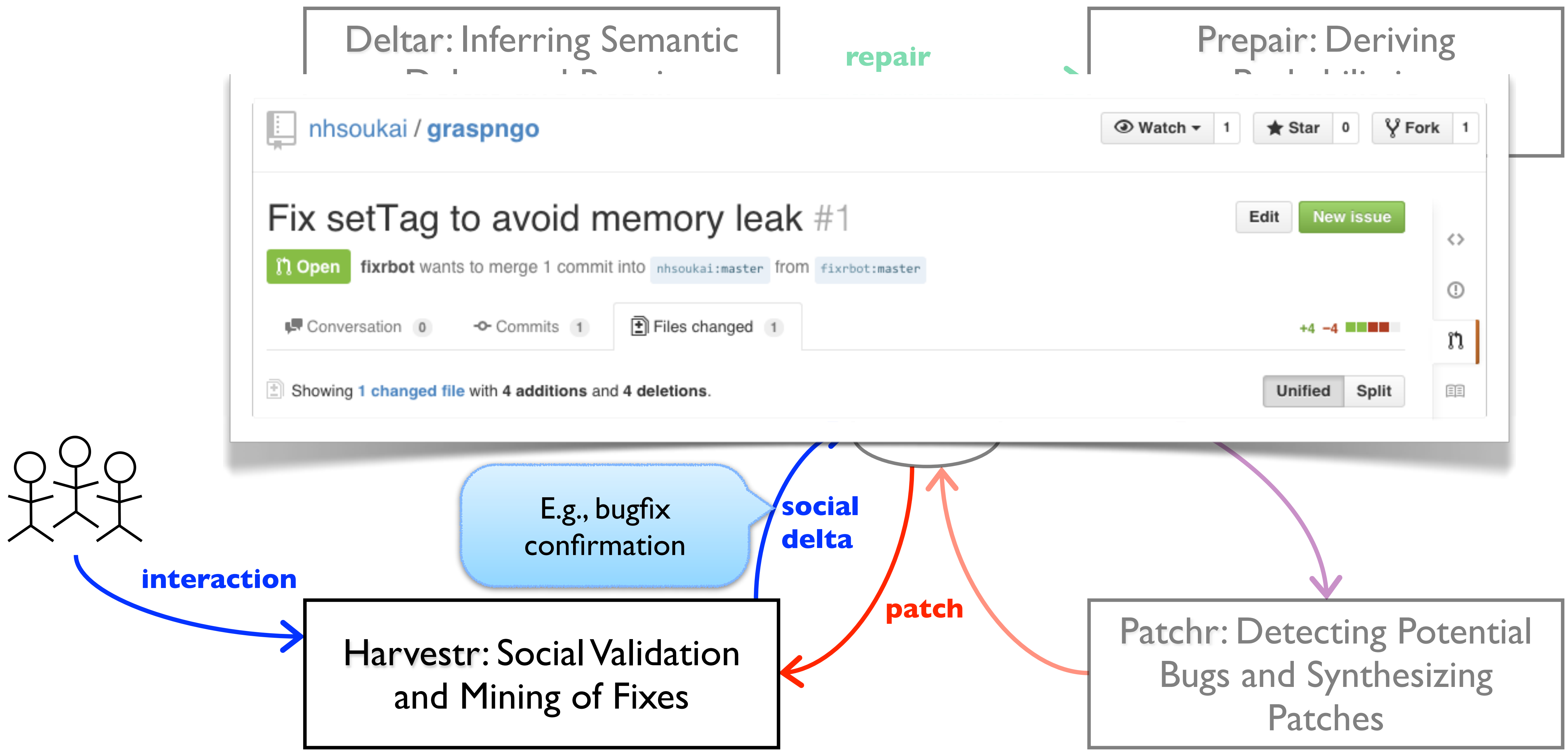
Fixr: Proposed System



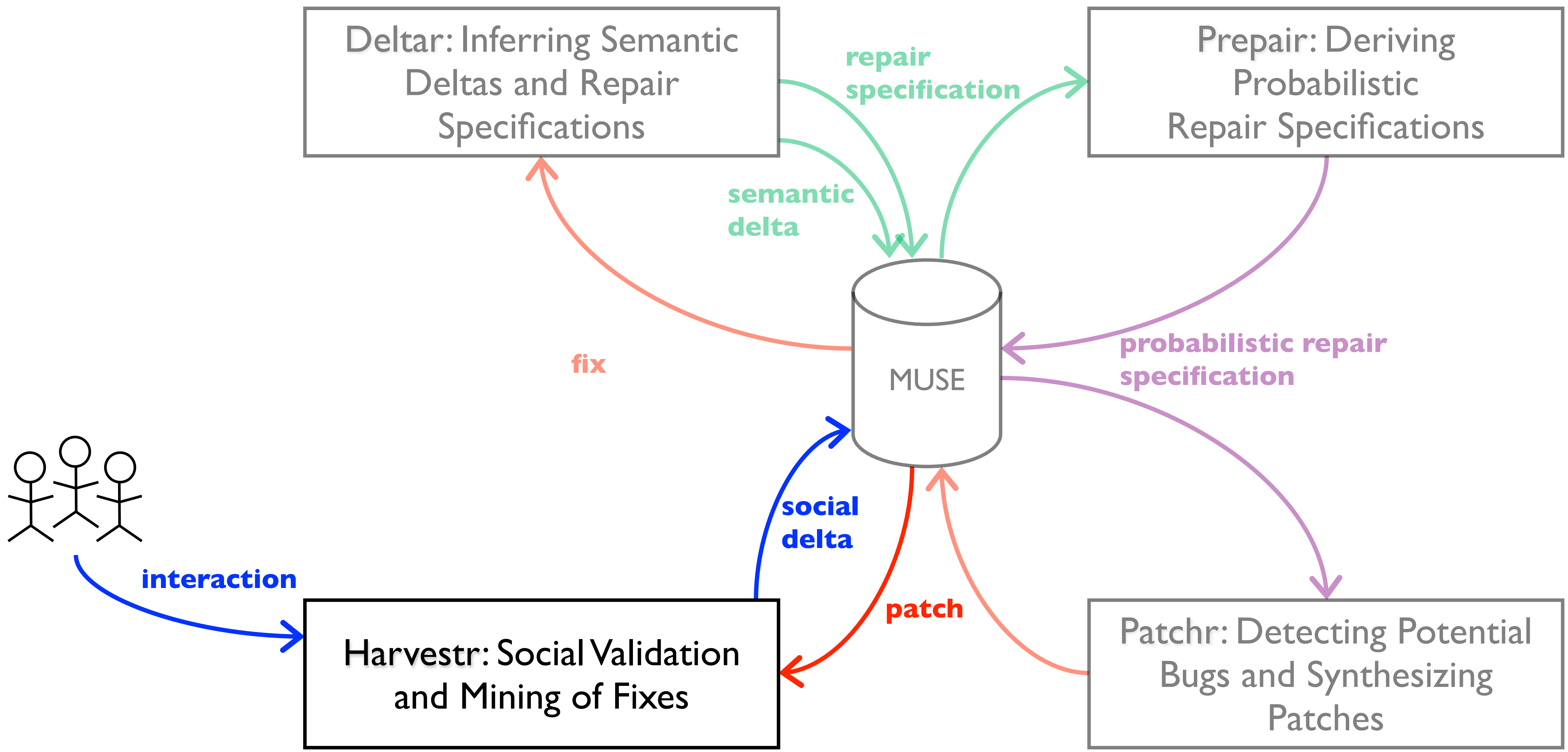
Fixr: Proposed System



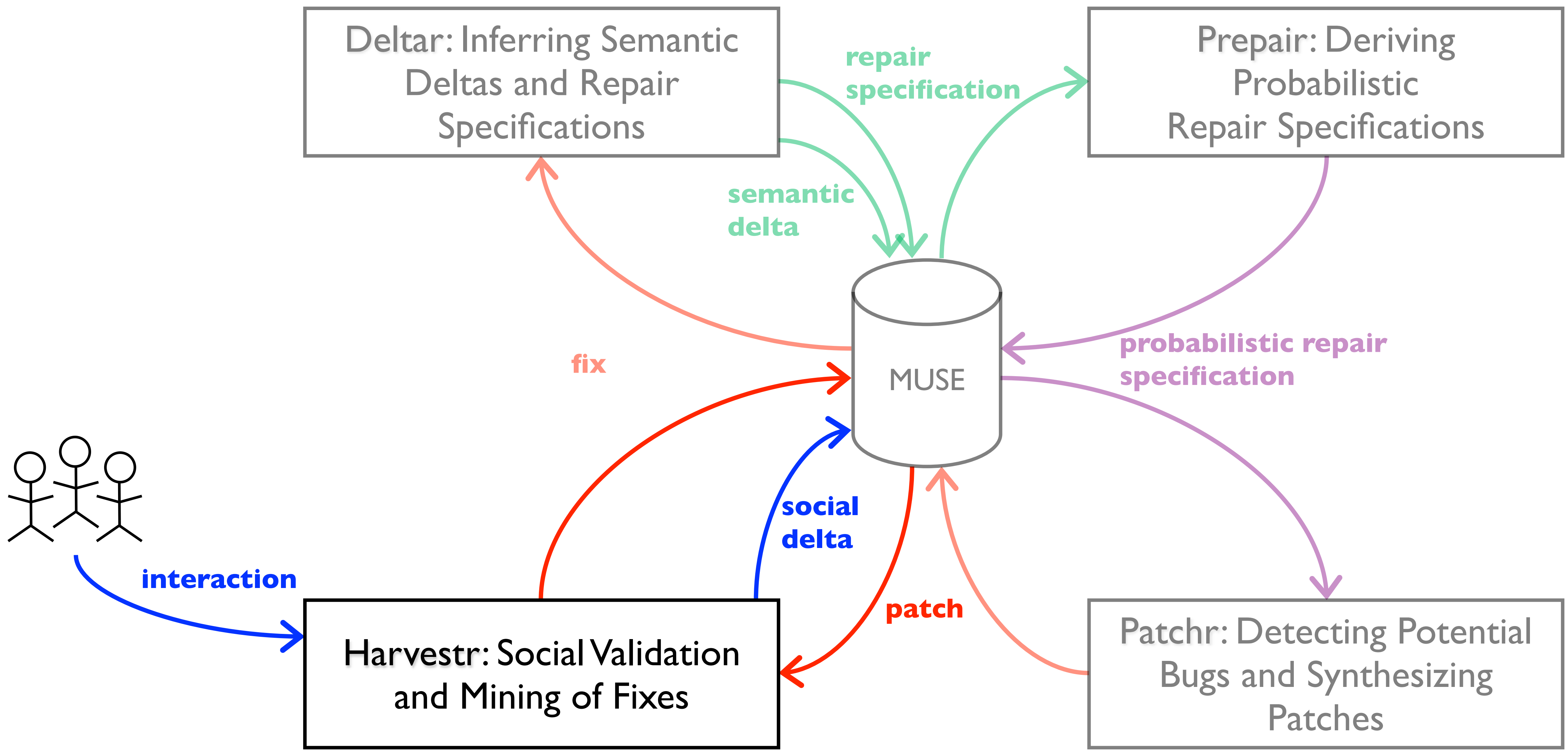
Fixr: Proposed System



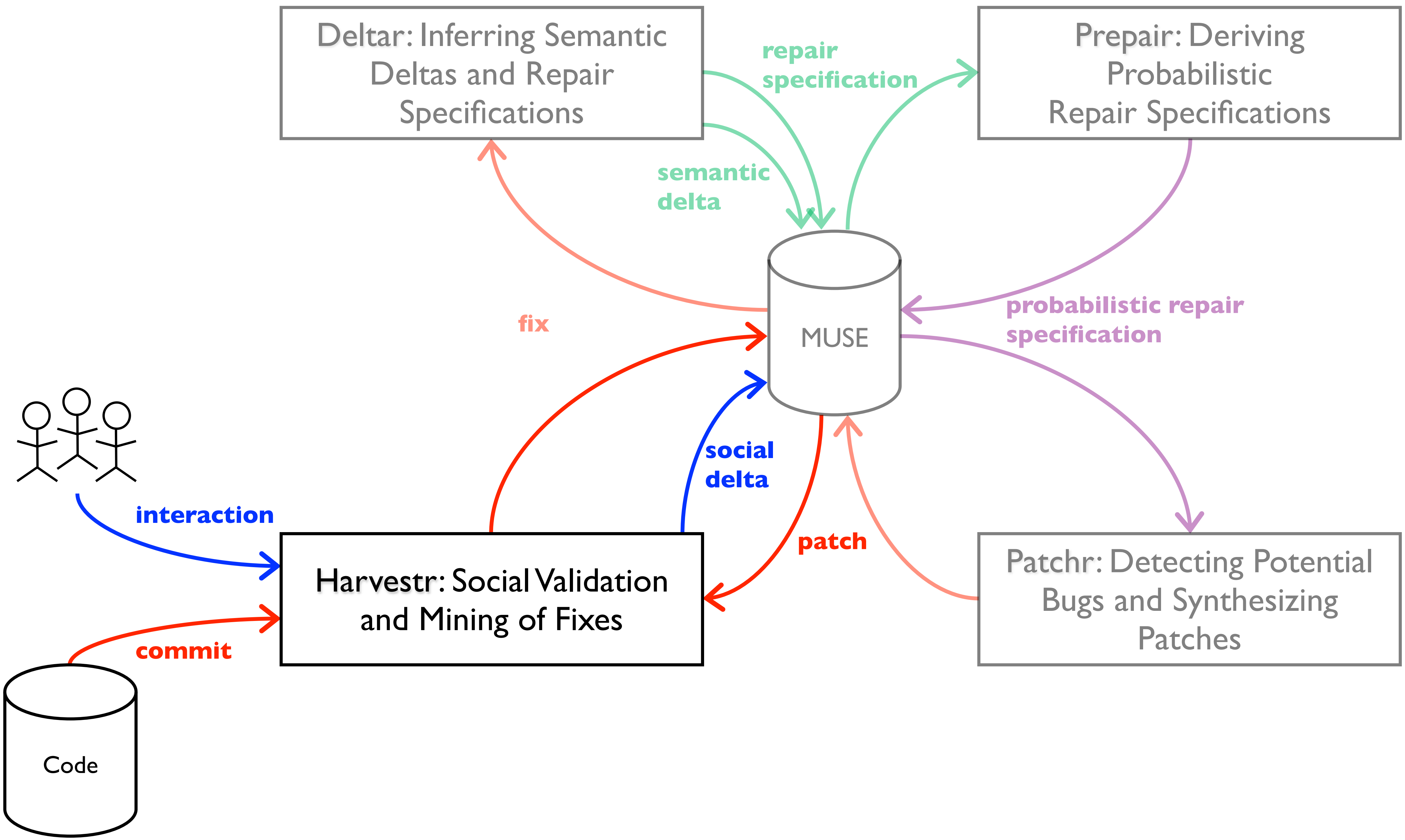
Fixr: Proposed System



Fixr: Proposed System

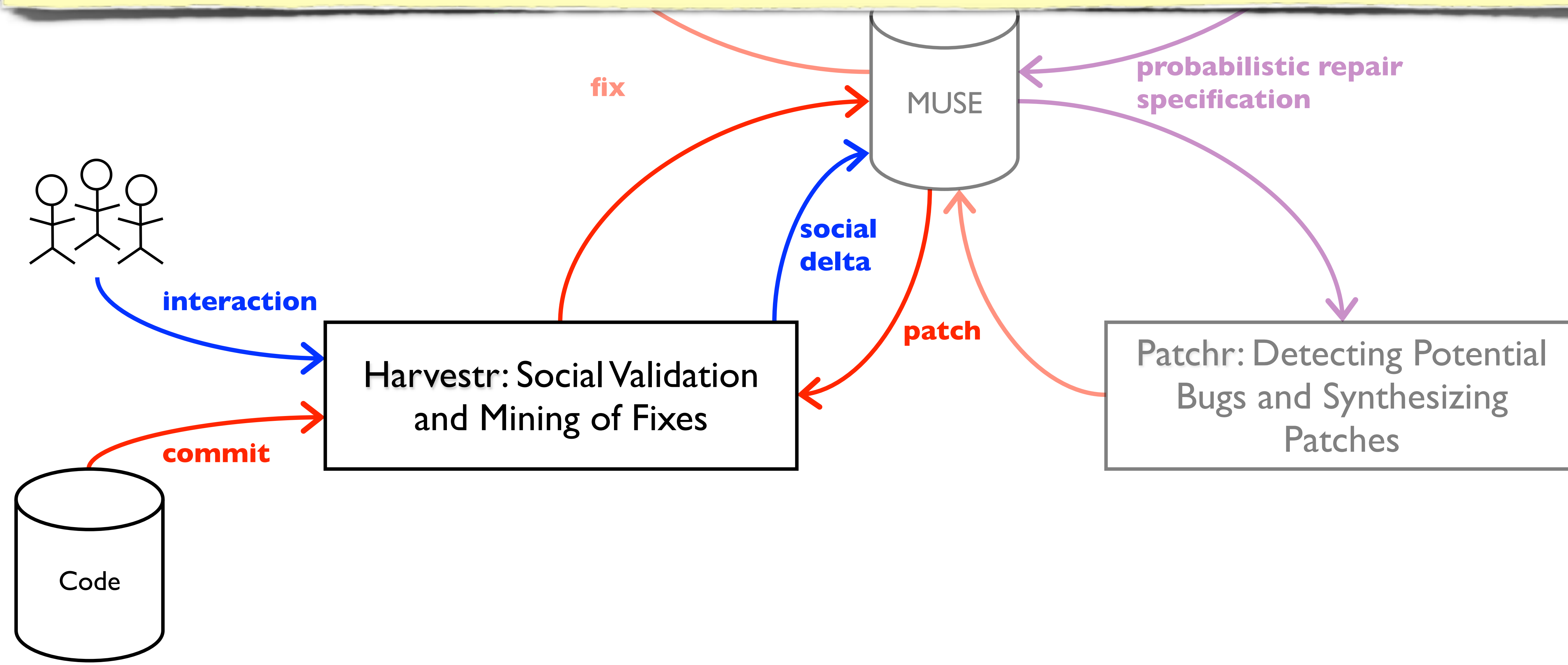


Fixr: Proposed System



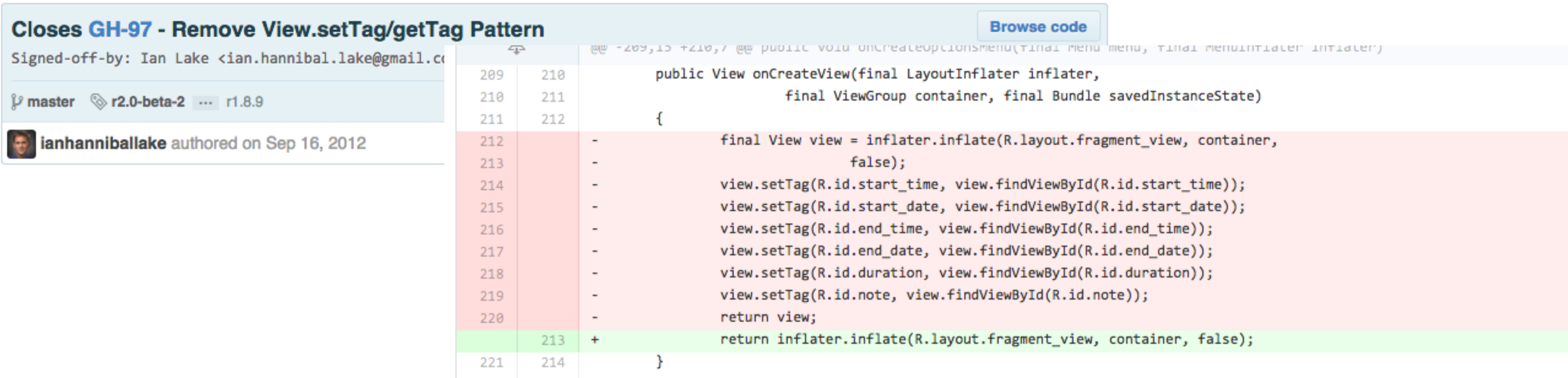
Fixr: Proposed System

Component: Harvestr *maps* commits and patches to candidate fixes



Harvestr

Harvestr



Closes GH-97 - Remove View.setTag/getTag Pattern Browse code

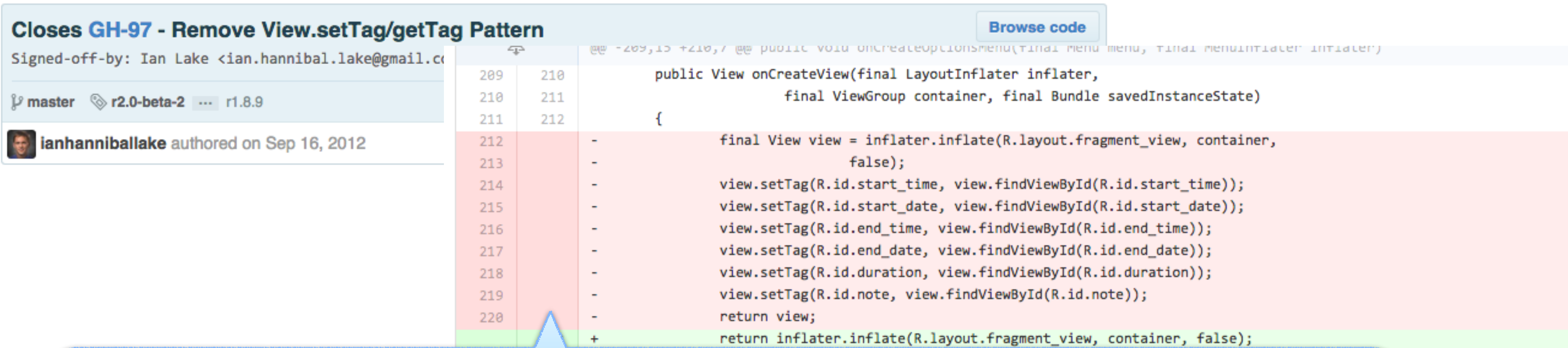
Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.c

master r2.0-beta-2 ... r1.8.9

ianhanniballake authored on Sep 16, 2012

```
@@ -209,15 +210,7 @@ public void onCreateOptionsMenu(final Menu menu, final MenuInflater inflater)
         public View onCreateView(final LayoutInflater inflater,
                     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;
+            return inflater.inflate(R.layout.fragment_view, container, false);
         }
```

Harvestr



Problem: How do we find relevant bugfixes?

Harvestr

Closes GH-97 - Remove View.setTag/getTag Pattern Browse code

`Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.c`

master r2.0-beta-2 ... r1.8.9

 **ianhanniballake** authored on Sep 16, 2012

```
         public View onCreateView(final LayoutInflater inflater,
                 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;
+            return inflater.inflate(R.layout.fragment_view, container, false);
```

Problem: How do we find relevant bugfixes?

ianhanniballake commented on Sep 16, 2012 Owner

Remove View.setTag/getTag pattern to prevent crashes due to out of memory error as per Lint error:

"Prior to Android 4.0, the implementation of View.setTag(int, Object) would store the objects in a static map, where the values were strongly referenced. This means that if the object contains any references pointing back to the context, the context (which points to pretty much everything else) will leak. If you pass a view, the view provides a reference to the context that created it. Similarly, view holders typically contain a view, and cursors are sometimes also associated with views."

Harvestr

Closes GH-97 - Remove View.setTag/getTag Pattern Browse code

Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.c

master r2.0-beta-2 … r1.8.9

 **ianhanniballake** authored on Sep 16, 2012

```
@@ -209,15 +210,7 @@ public void onCreateOptionsMenu(final Menu menu, final MenuInflater inflater)
      public View onCreateView(final LayoutInflater inflater,
                      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;
+         return inflater.inflate(R.layout.fragment_view, container, false);
```

Problem: How do we find relevant bugfixes?

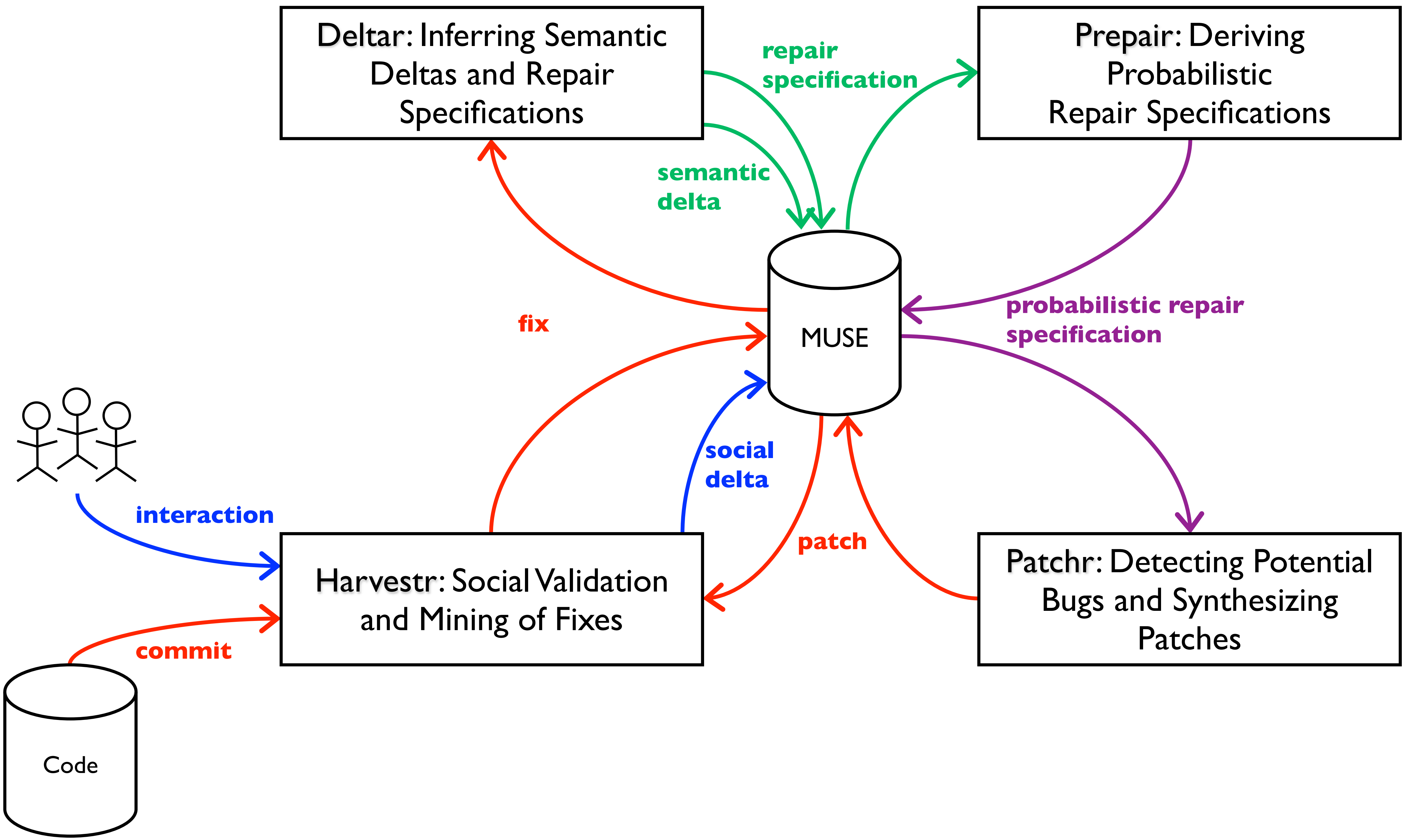
ianhanniballake commented on Sep 16, 2012 Owner

Remove View.setTag/getTag pattern to prevent crashes due to out of memory error as per Lint error:

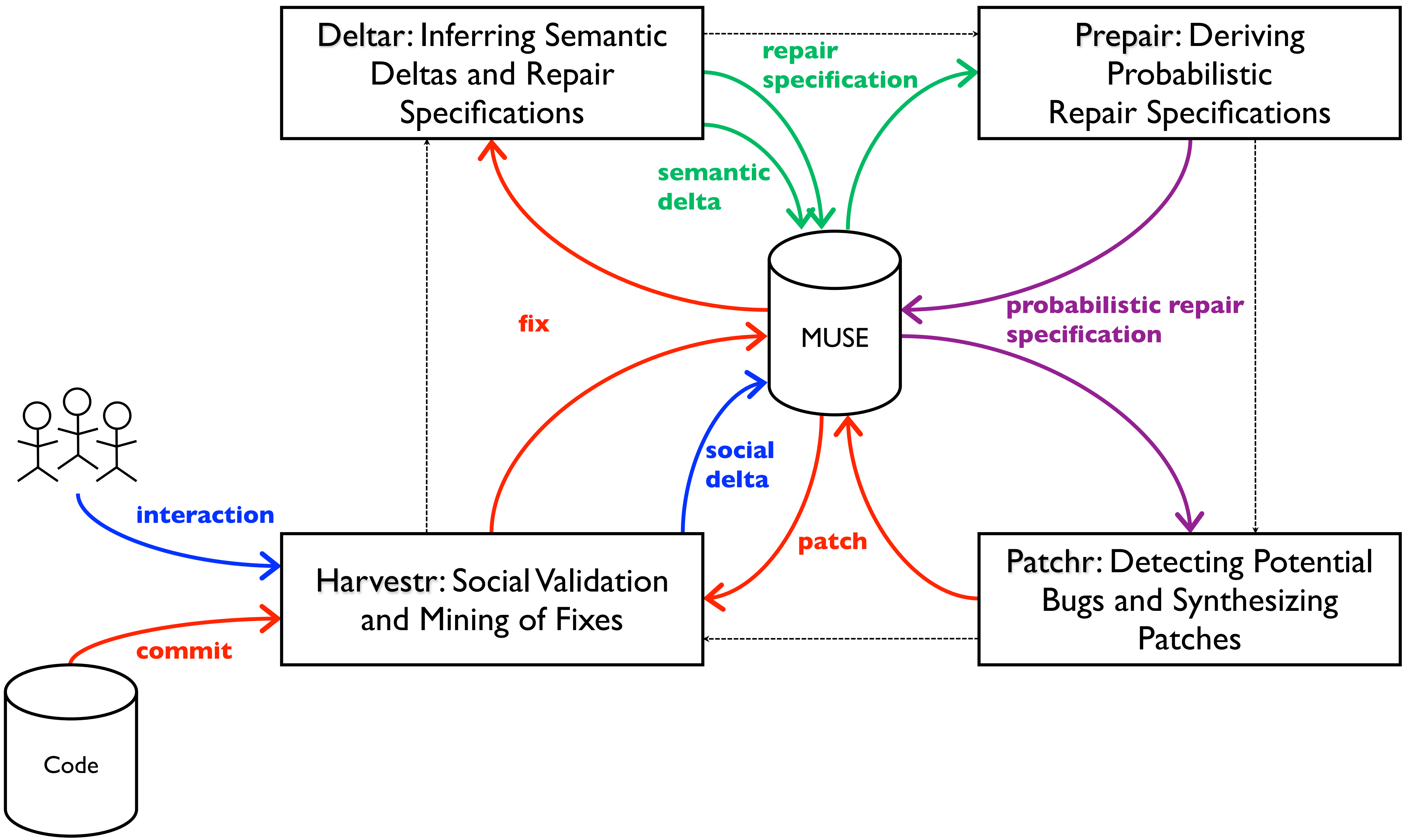
"Prior to Android 4.0, the implementation of View.setTag(int, Object) would store the objects in a static map, where the values were strongly referenced. This means that if the object contains any references pointing back to the context, the context (which points to pretty much everything else) will leak. If you pass a view, the view provides a reference to the context that created it. Similarly, view holders typically contain a view, and cursors are sometimes also associated with views."

Approach: Mine meta-data artifacts

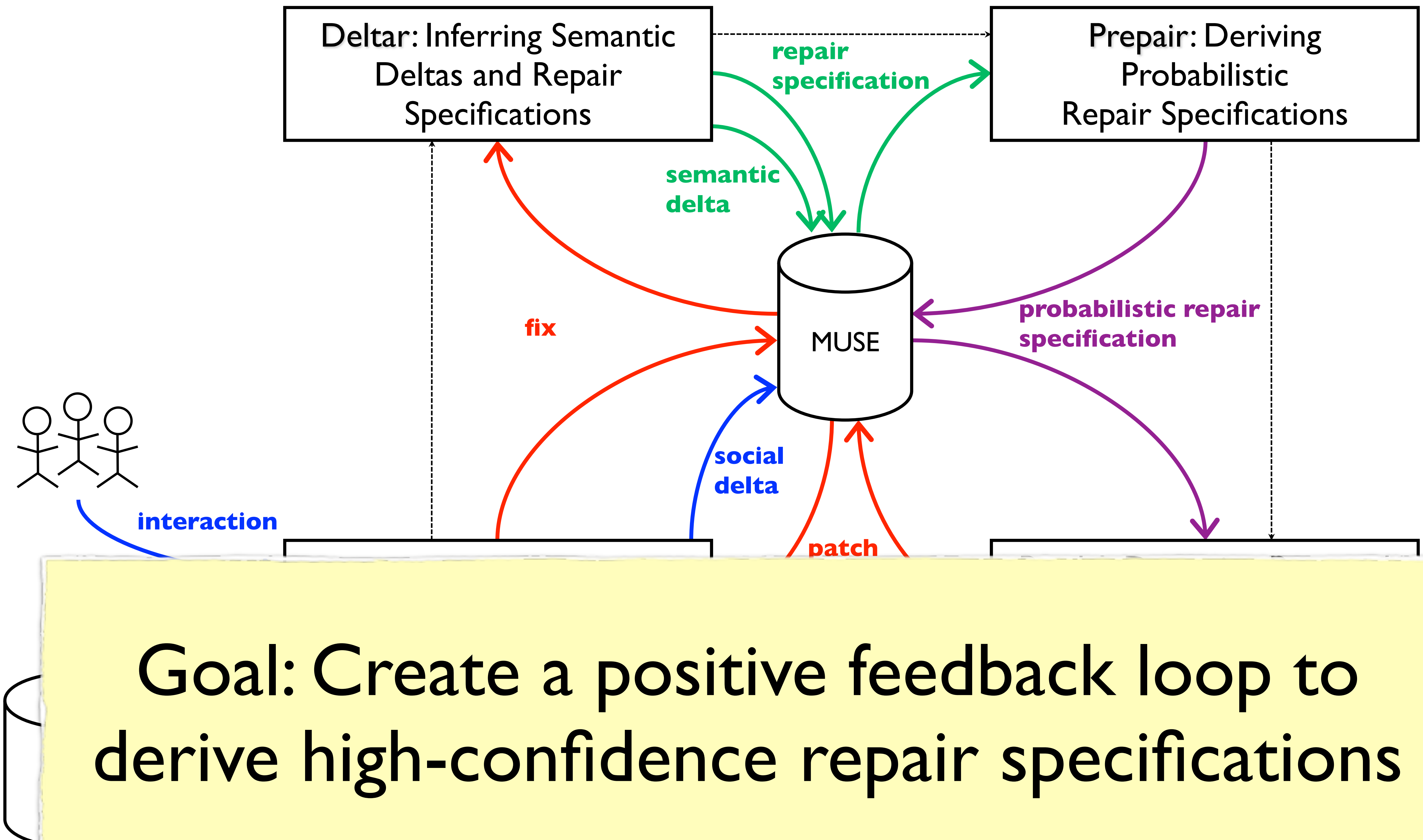
Fixr: Proposed System



Fixr: Proposed System

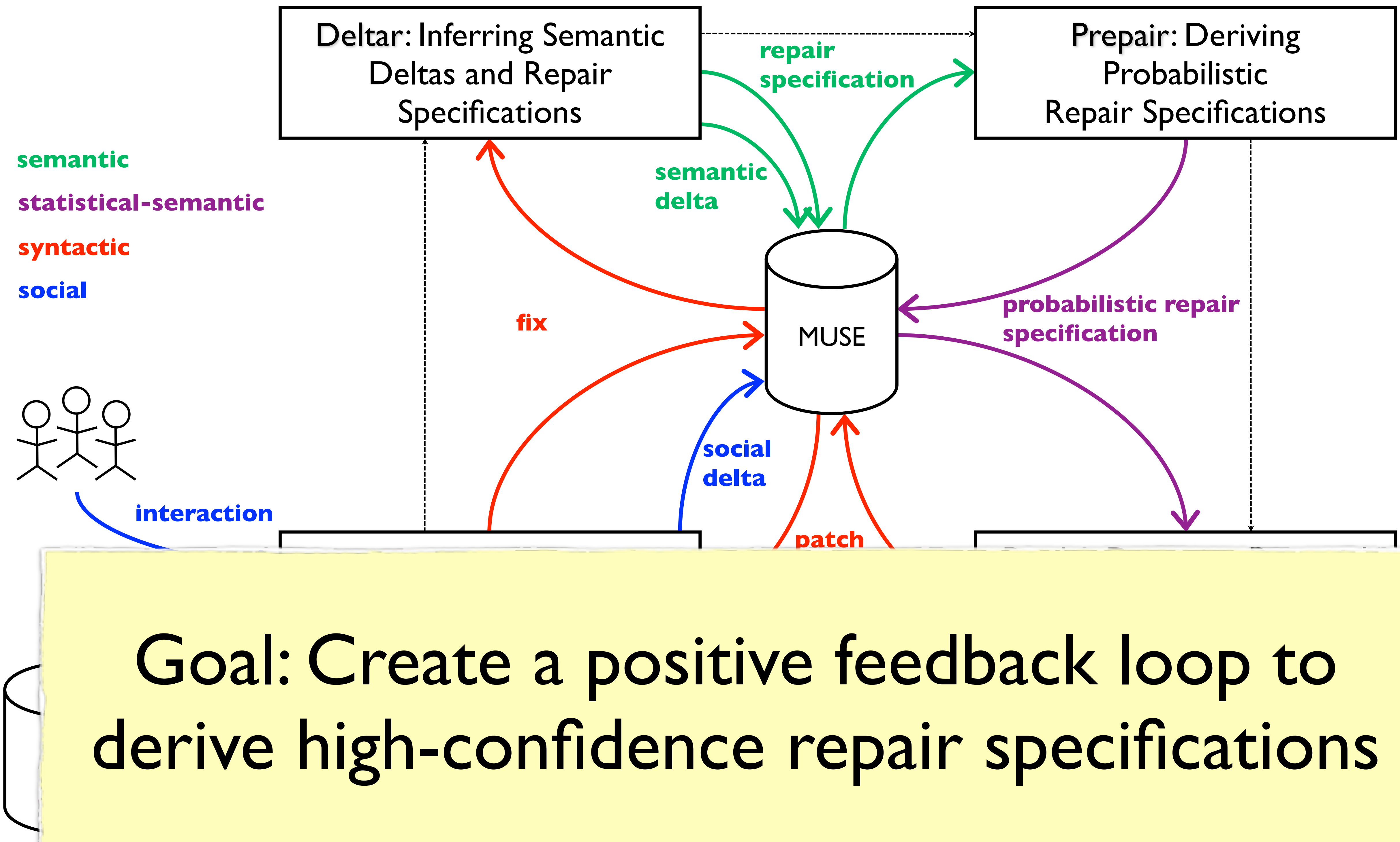


Fixr: Proposed System

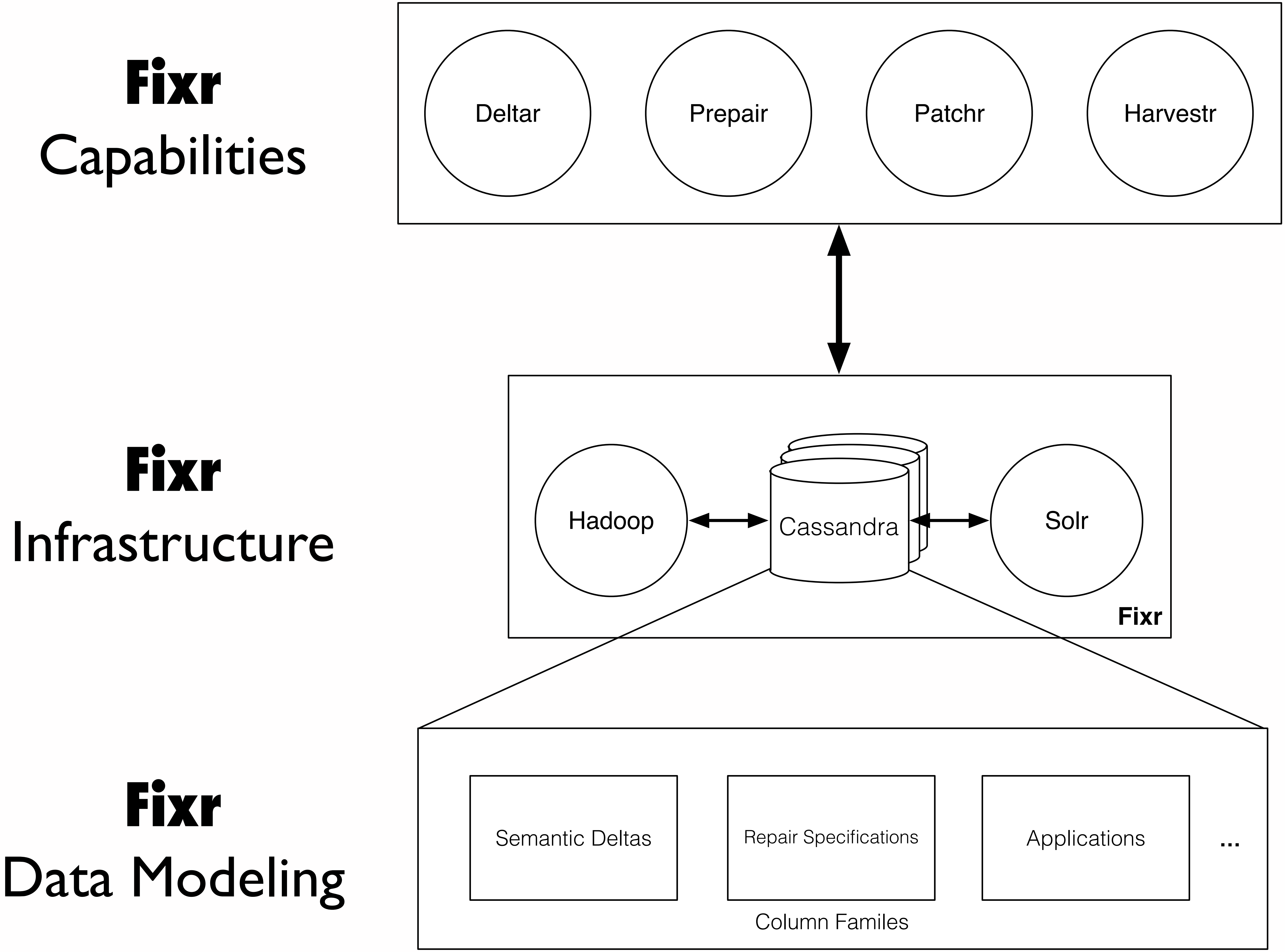


Goal: Create a positive feedback loop to derive high-confidence repair specifications

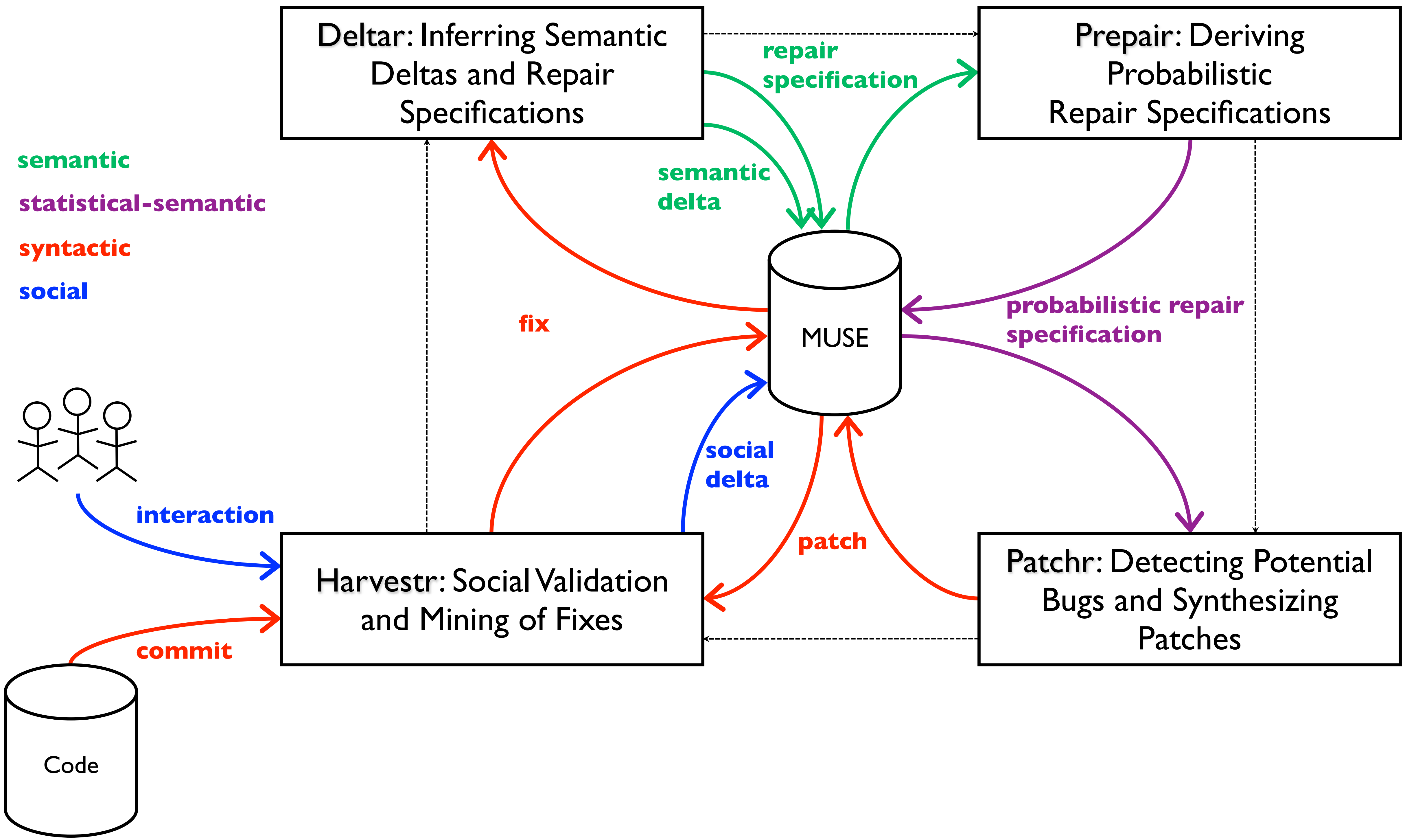
Fixr: Proposed System



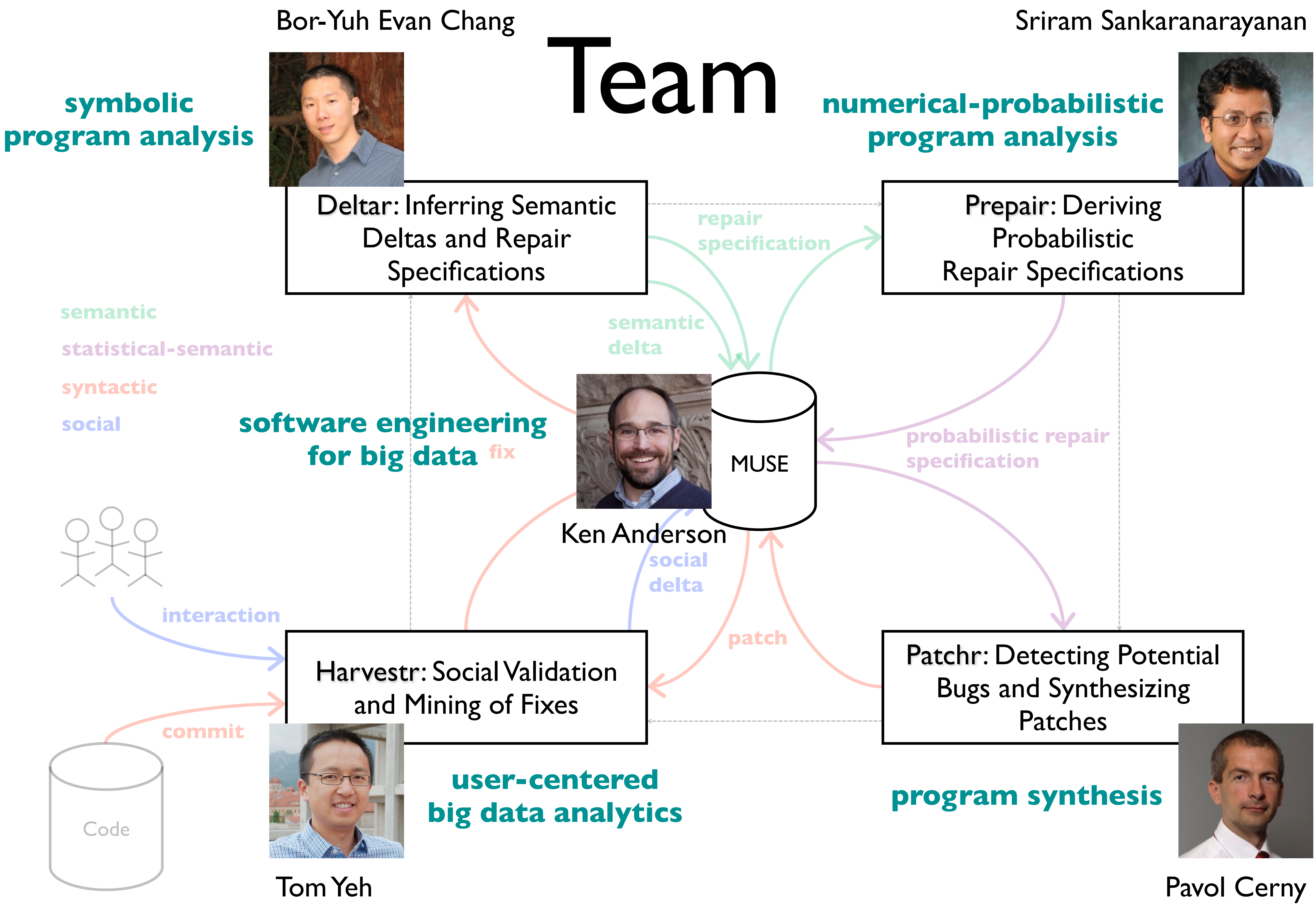
Goal: Create a positive feedback loop to derive high-confidence repair specifications



Fixr
Capabilities
Deltar
Prepair
Patchr
Harvestr
Fixr
Infrastructure
Hadoop
Cassandra
Solr
Fixr
Fixr
Data Modeling
Semantic Deltas
Repair Specifications
Applications
...
Column Familes


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

Team



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

Evaluation Questions

Evaluation Questions

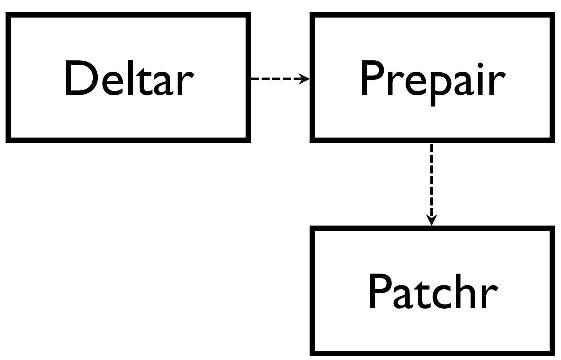
- Iterative and incremental design and evaluation of the **Fixr** loop

Evaluation Questions

- Iterative and incremental design and evaluation of the **Fixr** loop
  - Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?

Evaluation Questions

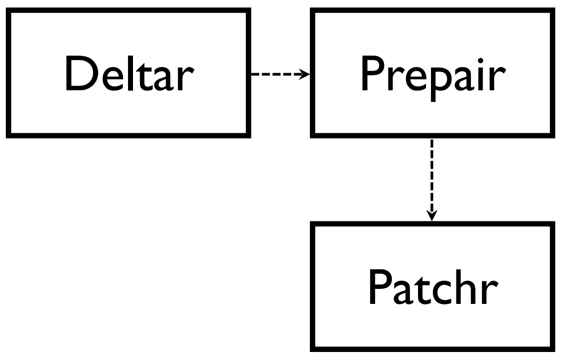
- Iterative and incremental design and evaluation of the **Fixr** loop



- Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?

Evaluation Questions

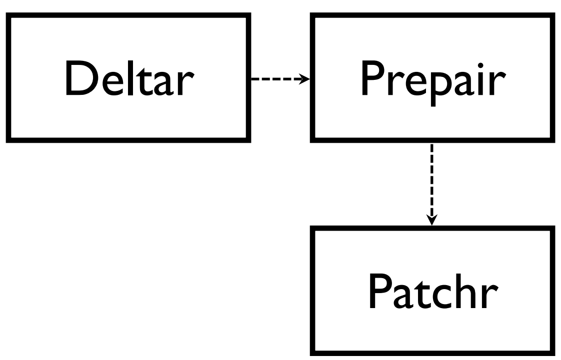
- Iterative and incremental design and evaluation of the **Fixr** loop



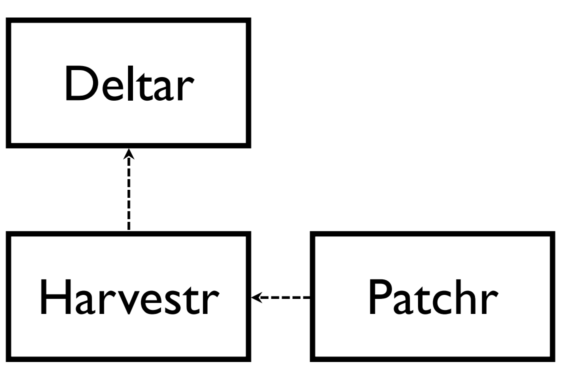
- Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?
- Effectiveness of Bugfix Seeding: Can we isolate likely bugfixes from source repositories?

Evaluation Questions

- Iterative and incremental design and evaluation of the **Fixr** loop



- Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?



- Effectiveness of Bugfix Seeding: Can we isolate likely bugfixes from source repositories?

Prepair for **Fixr**

www.cs.colorado.edu/~bec
pl.cs.colorado.edu

PLV

Most Relevant References

Deltar

Blackshear, Sam, Bor-Yuh Evan Chang, and Manu Sridharan. 2013. "Thresher: Precise Refutations for Heap Reachability." In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 275–86. PLDI '13. ACM. doi:10.1145/2491956.2462186.

Coughlin, Devin, and Bor-Yuh Evan Chang. 2014. "Fissile Type Analysis: Modular Checking of Almost Everywhere Invariants." In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 73–85. POPL '14. ACM. doi:10.1145/2535838.2535855.

Prepair

Ivancic, F., G. Balakrishnan, A. Gupta, S. Sankaranarayanan, N. Maeda, H. Tokuoka, T. Imoto, and Y. Miyazaki. 2011. "DC2: A Framework for Scalable, Scope-Bounded Software Verification." In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 133–42. doi:10.1109/ASE.2011.6100046.

Sankaranarayanan, Sriram, Aleksandar Chakarov, and Sumit Gulwani. 2013. "Static Analysis for Probabilistic Programs: Inferring Whole Program Properties from Finitely Many Paths." In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 447–58. PLDI '13. ACM. doi:10.1145/2491956.2462179.

Sankaranarayanan, Sriram, Swarat Chaudhuri, Franjo Ivančić, and Aarti Gupta. 2008. "Dynamic Inference of Likely Data Preconditions over Predicates by Tree Learning." In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, 295–306. ISSTA '08. ACM. doi:10.1145/1390630.1390666.

Sankaranarayanan, Sriram, Franjo Ivančić, and Aarti Gupta. 2008. "Mining Library Specifications Using Inductive Logic Programming." In *Proceedings of the 30th International Conference on Software Engineering*, 131–40. ICSE '08. ACM. doi:10.1145/1368088.1368107.

Patchr

Černý, Pavol, Krishnendu Chatterjee, Thomas A. Henzinger, Arjun Radhakrishna, and Rohit Singh. 2011. "Quantitative Synthesis for Concurrent Programs." In *Computer Aided Verification*, 243–59. Lecture Notes in Computer Science 6806. Springer Berlin Heidelberg.

Černý, Pavol, Thomas A. Henzinger, Arjun Radhakrishna, Leonid Ryzhyk, and Thorsten Tarrach. 2013. "Efficient Synthesis for Concurrency by Semantics-Preserving Transformations." In *Computer Aided Verification*, 951–67. Lecture Notes in Computer Science 8044. Springer Berlin Heidelberg.

Harvestr

Alharbi, Khalid, Sam Blackshear, Emily Kowalczyk, Atif M. Memon, Bor-Yuh Evan Chang, and Tom Yeh. 2014. "Android Apps Consistency Scrutinized." In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, 2347–52. CHI EA '14. ACM. doi:10.1145/2559206.2581352.

Yeh, Tom, Brandyn White, Jose San Pedro, Boriz Katz, and Larry S. Davis. 2011. "A Case for Query by Image and Text Content: Searching Computer Help Using Screenshots and Keywords." In *Proceedings of the 20th International Conference on World Wide Web*, 775–84. WWW '11. ACM. doi:10.1145/1963405.1963513.

Fixr Database

Anderson, Kenneth Mark, Aaron Schram, Ali Alzabarah, and Leysia Palen. 2013. "Architectural Implications of Social Media Analytics in Support of Crisis Informatics Research." *IEEE Data Eng. Bull.* 36 (3): 13–20.

Schram, Aaron, and Kenneth M. Anderson. 2012. "MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability." In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, 191–202. SPLASH '12. ACM. doi:10.1145/2384716.2384773.