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





A bug that manifests spectacularly ...





A bug that manifests spectacularly ...





A bug that manifests spectacularly ...





A bug that manifests spectacularly ...



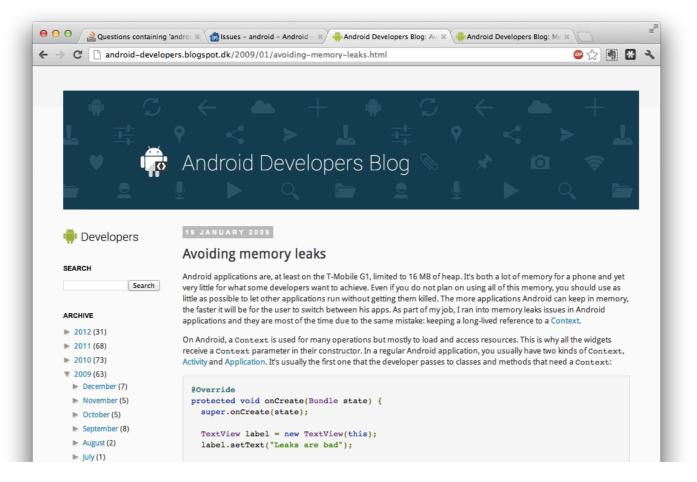


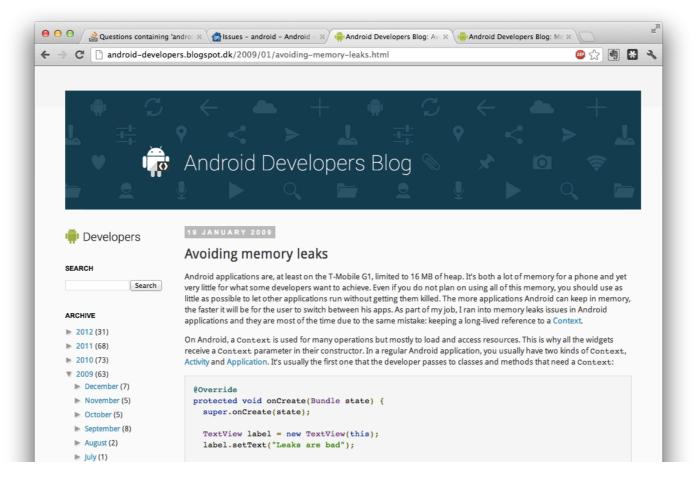
A bug that manifests spectacularly ...

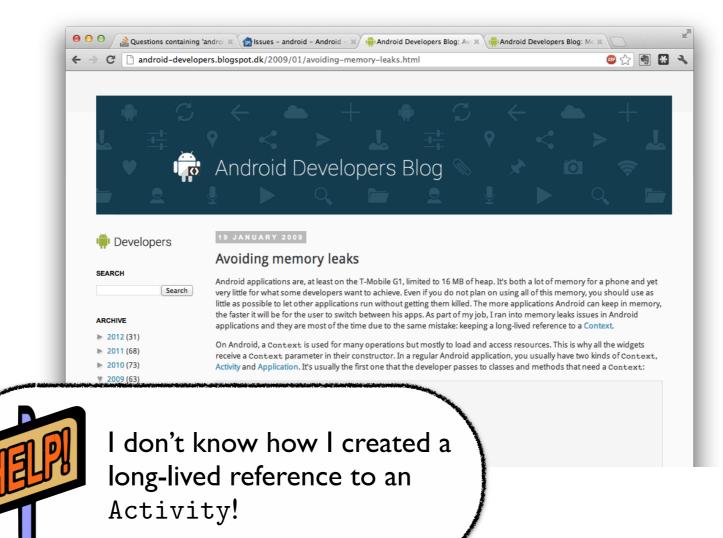




caused by an app-created memory leak











Imagining a post-MUSE scenario ...



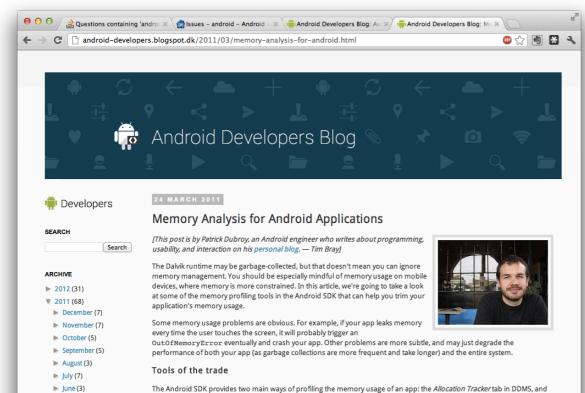
I don't know how I created a long-lived reference to an Activity!

for



▶ May (5)

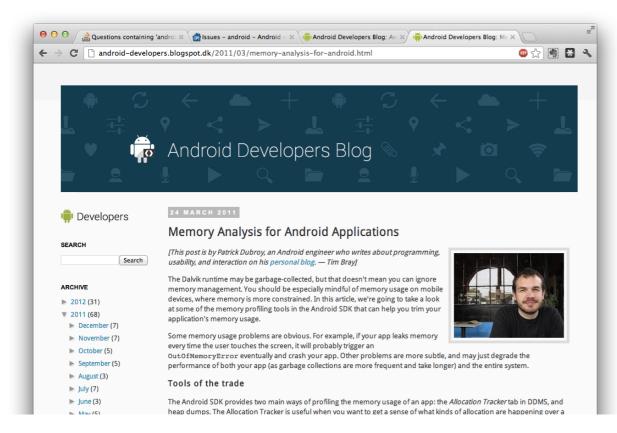
The Android SDK provides two main ways of profiling the memory usage of an app: the *Allocation Tracker* tab in DDMS, and heap dumps. The Allocation Tracker is useful when you want to get a sense of what kinds of allocation are happening over a



The Android SDK provides two main ways of profiling the memory usage of an app: the *Allocation Tracker* tab in DDMS, and heap dumps. The Allocation Tracker is useful when you want to get a sense of what kinds of allocation are happening over a

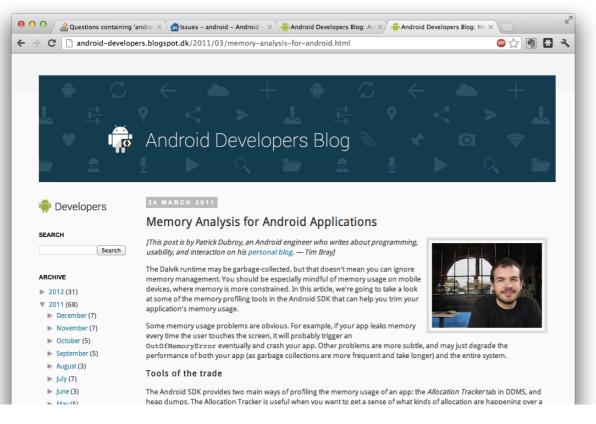
I. Run the app

► May (5)



Run the app Watch the heap usage

Info Threads VM Hea		ocation Track	,	Emulator		nt Log
Heap updates will happen after e	very GC	for this client				
ID Heap Size Allocated	Free	% Used		Cause GC		
1 8.570 MB 8.452 MB 121	.320 KB	98.62%	59,281			
Display: Stats 🗘						
••					-	
Туре	Count	Total Size	Smallest	Largest	Median	Average
free	1,772	107.312 KB	16 B	48.297 KB	24 B	62 B
data object	40,528	1.229 MB	16 B	1.047 KB	32 B	31 B
class object	2,187	637.234 KB	168 B	34.125 KB	168 B	298 B
1-byte array (byte[], boolean[])	2,247	5.654 MB	24 B	1.500 MB	48 B	2.576 KB
2-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
non-Java object	92	14.219 KB	16 B	8.023 KB	32 B	158 B

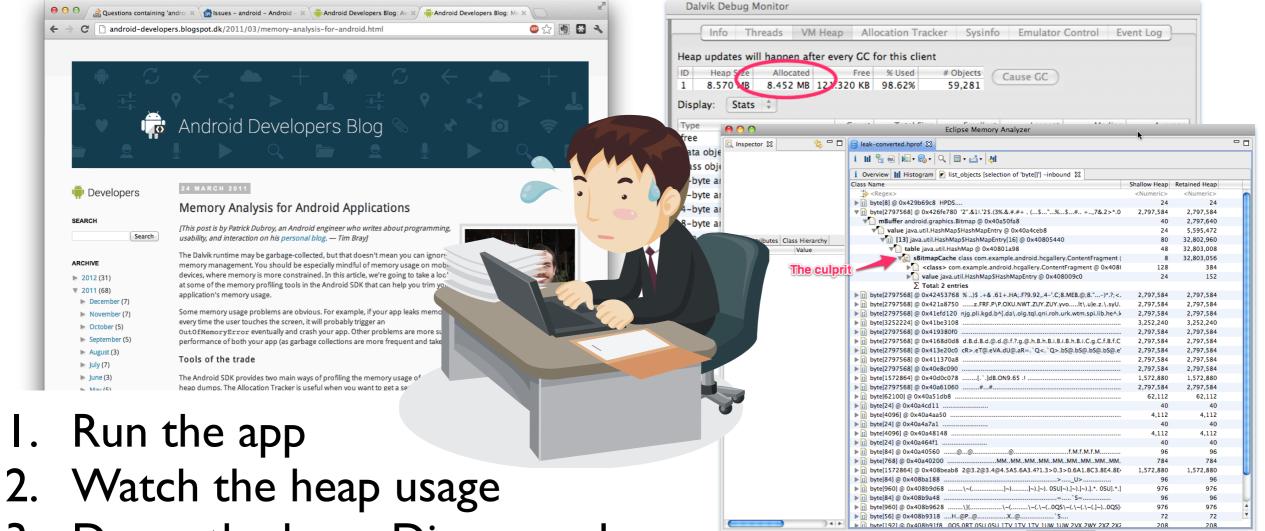


- I. Run the app
- 2. Watch the heap usage
- 3. Dump the heap. Dig around and finally find the culprit!

Inf	o Threads VM Heap	Allocation Tracker Sysinfo Emulator Control Ev	ent Log		
eap upd	ates will happen after ever	y GC for this client			
	ap Size Allocated	Free Millard # Objects			
	70 MB 8.452 MB 121.32				
isplay:	Stats 🗘				
ype	0.0.0	en en en le la	•		
ree	000	Eclipse Memory Analyzer	-		
	🖾 Inspector 🕱 🛛 🔄 🗖		~		- [
ata obje		i III 🖫 😡 🔎 • 🔍 💷 • 🖄 • 🔄			
lass obje					
-byte ar		i Overview III Histogram 🕑 list_objects [selection of 'byte[]'] -inbound 🔀	C 1 H H		
		Class Name	<pre>Shallow Heap <numeric></numeric></pre>	<pre>Retained Heap </pre> <pre></pre> <	
-byte ar		byte[8] @ 0x429b69c8 HPDS	24	24	- (
-byte ar		v byte[2797568] @ 0x426fe780 '2".&1!.'2\$.(3%.&.#.#+ . (\$"%\$# +,7&.2>*.0	2,797,584	2,797,584	
-byte ar		mBuffer android.graphics.Bitmap @ 0x40a50fa8	40	2,797,640	
		Value java.util.HashMapSHashMapEntry @ 0x40a4ceb8	24	5,595,472	
on-Java	Statics Attributes Class Hierarchy	🐨 👔 [13] java.util.HashMapSHashMapEntry[16] @ 0x40805440	80	32,802,960	
	Type Name Value	table java.util.HashMap @ 0x40801a98	48	32,803,008	
_		sBitmapCache class com.example.android.hcgallery.ContentFragment (8	32,803,056	
	The culpri	class> com.example.android.hcgallery.ContentFragment @ 0x408(128	384	
_		Value java.util.HashMapSHashMapEntry @ 0x408009c0 ∑ Total: 2 entries	24	152	
_		byte[2797568] @ 0x42453768 %)\$.+& .61+.HA;.F?9.92,.4-'.C;8.MEB.@;8.")*.?;<.	2,797,584	2,797,584	
		byte[2797568] @ 0x421a8750z.FRF.P\P.OXU.NWT.ZUY.ZUY.yvolt.u]e.z.\.syU.	2,797,584	2,797,584	
		bib 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	1,572,880	1,572,880	
		byte[2797568] @ 0x40a51060## byte[62100] @ 0x40a51db8##	2,797,584 62,112	2,797,584 62,112	
		byte[b2100] @ 0x40a510b8	62,112	62,112	
		b) byte[24] @ 0x40a4c011	4,112	4,112	
		b tte[24] @ 0x40a4a7a1	40	40	
		▶	4,112	4,112	
		▶ []] byte[24] @ 0x40a464f1	40	40	
		▶ 💼 byte[84] @ 0x40a40560@@@@@@@	96	96	
		▶ [] byte[768] @ 0x40a40200MMMMMMMMMMMMMMMM.	784	784	
		▶ [] byte[1572864] @ 0x408beab8 2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3.8E4.8E4	1,572,880	1,572,880	
		▶ [] byte[84] @ 0x408ba188	96	96	
		▶ [] byte[960] @ 0x408b9d68\~(]~)]~)]~). 0SU]~).]~)]*. 0SU].*.]	976	976	
		▶ [] byte[84] @ 0x408b9a48	96 976	96 976	
		byte[56] @ 0x408b9318H@P@X@	72	72	

Developers	Android Developers Blog 24 MARCH 2011 Memory Analysis for Android Applications This post is by Patrick Dubroy, an Android engineer who writes about programming.	Heap updates will happen after every CC for this client D Heap 5/20 Allocated 12 Free % Used # Objects Cause GC 1 8.570 VB 8.452 MB 123.320 KB 98.62% 59,281 Display: Stats Type Eclipse Memory Analyzer free ata obje ata obje ass obje -byte ar -byte ar -byte ar		
Developers	24 MARCH 2011 Memory Analysis for Android Applications	Type Image: Construction of the construc	<u>k</u>	
Developers	24 MARCH 2011 Memory Analysis for Android Applications	free ata obje ass obje -byte at -byte at	••••••	
SEARCH	Memory Analysis for Android Applications	i ii 🦉 🖗 🗟 · 🔍 🖾 · 🖄 🤞 i Overview iii Histogram 🖻 list_objects [selection of 'byte[]'] –inbound 🕸 Class Name -byte at		
SEARCH	Memory Analysis for Android Applications	-byte an -byte		
SEARCH	Memory Analysis for Android Applications	- byte at B - byte at		
SEARCH	Memory Analysis for Android Applications		Shallow Heap	Retained Heap
SEARCH		▶ n byte(8) @ 0x429b69c8 HPDS	<numeric></numeric>	<numeric></numeric>
			24	
	This post is by Patrick Dubroy, an Android engineer who writes about programming.	▼ Dyte[2797568] @ 0x426fe780 '2".&11.'25.(3%&#.#+. (5"%\$# +,7</td><td></td><td></td></tr><tr><td>Sear</td><td>ITTIS DUSLIS DV PAUTICK DUDFOV, AN ANOFOIO ENGINEER WNO WRITES ADOUT DROGRAMMING.</td><td>S-byte al The mBuffer android.graphics.Bitmap @ 0x40a50fa8</td><td>40</td><td>-,,</td></tr><tr><td>Sear</td><td></td><td>Value java.util.HashMapSHashMapEntry @ 0x40a4ceb8</td><td>24</td><td></td></tr><tr><td></td><td>usability, and interaction on his personal blog. — Tim Bray]</td><td>vibutes Class Hierarchy V [] [13] java.util.HashMapSHashMapEntry[16] @ 0x40805440</td><td>80</td><td></td></tr><tr><td></td><td>The Dalvik runtime may be garbage-collected, but that doesn't mean you can ignore</td><td>Value Value value</td><td>48</td><td>,,</td></tr><tr><td>ARCHIVE</td><td>memory management. You should be especially mindful of memory usage on mobile</td><td>StitmapCache class com.example.android.hcgallery.ContentFro</td><td></td><td></td></tr><tr><td></td><td>devices, where memory is more constrained. In this article, we're going to take a loc'</td><td>The culprit - Casas com.example.android.hegallery.ContentFragment</td><td></td><td></td></tr><tr><td>2012 (31)</td><td></td><td>P Nalue java.util.HashMapSHashMapEntry @ 0x408009c0</td><td>24</td><td>152</td></tr><tr><td>2011 (68)</td><td>at some of the memory profiling tools in the Android SDK that can help you trim you</td><td></td><td>18 2 4 2 707 501</td><td>3 707 661</td></tr><tr><td>December (7)</td><td>application's memory usage.</td><td>▶ [] byte[2797568] @ 0x42453768 %]5 -+& .51+.14,.F9.92,.4-1(-8,.ME@,8,*)</td><td></td><td></td></tr><tr><td></td><td>Some memory usage problems are obvious. For example, if your app leaks memo</td><td>▶</td><td></td><td></td></tr><tr><td>November (7)</td><td>every time the user touches the screen, it will probably trigger an</td><td>bit byte[2797568] @ 0x41efd120 njg.pli.kgd.b4[.da\.olg.tql.qni.roh.urk.wtm.spi. bit byte[3252224] @ 0x41be3108 </td><td>he^.k 2,797,584 3,252,240</td><td></td></tr><tr><td>October (5)</td><td>OutOfMemoryError eventually and crash your app. Other problems are more su</td><td>Dyte:sc/sc/241@UX410e3108 Dyte:sc/sc/241@UX410e3108 Dyte:sc/sc/241@UX410e3108 Dyte:sc/sc/24108000 Dyte:sc/sc/24108000 Dyte:sc/sc/sc/sc/sc/sc/sc/sc/sc/sc/sc/sc/sc/s</td><td></td><td></td></tr><tr><td>September (5)</td><td>performance of both your app (as garbage collections are more frequent and take</td><td>▶ □ byte(2797568) © 0X41930070 ▶ □ byte(2797568) © 0X4168d048 d.B.d.B.d.@.d.@.f.?g.@.h.B.h.B.I.B.h.B.L.C.g.</td><td></td><td></td></tr><tr><td></td><td>performance of both your app (as garbage collections are more frequent and take</td><td>■ Uyte(2797568) @ 0X41362000 G.8L3(@,004716)(B,158156)(G,2) ■ Uyte(2797568) @ 0X41362000 G.8L3(@,04A104)(B,2)(G,2)(B,56)(B,5) ■ Uyte(2797568) @ 0X41362000 G.8L3(@,04A104)(B,2)(G,2)(B,56)(B,5)(B,5)(B,5)(B,5)(B,5)(B,5)(B,5)(B,5</td><td></td><td></td></tr><tr><td>August (3)</td><td>Tools of the trade</td><td>→ D yrte[2797568] @ 0x411370a8</td><td>2,797,584</td><td></td></tr><tr><td>July (7)</td><td></td><td>▶ 0 yte[275756] @ 0x405709</td><td>2,797,584</td><td></td></tr><tr><td>June (3)</td><td>The Android SDK provides two main ways of profiling the memory usage of</td><td>by byte[1572864] @ 0x40d0c078</td><td></td><td>1 1 1 1 1</td></tr><tr><td>May (5)</td><td>heap dumps. The Allocation Tracker is useful when you want to get a se</td><td>► byte[2797568] © 0x40a61060##</td><td>2,797,584</td><td></td></tr><tr><td></td><td></td><td>▶ ⓑ byte[62100] @ 0x40a51db8</td><td></td><td></td></tr><tr><td></td><td></td><td>▶ ⓑ byte[24] @ 0x40a4cd11</td><td>40</td><td></td></tr><tr><td></td><td></td><td>▶ D byte[4096] @ 0x40a4aa50</td><td></td><td>4,112</td></tr><tr><td></td><td></td><td>▶ ⓑ byte[24] @ 0x40a4a7a1</td><td>40</td><td>40</td></tr><tr><td>RIIN</td><td>the app</td><td>▶ 🔯 byte[4096] @ 0x40a48148</td><td></td><td></td></tr><tr><td>INMII</td><td></td><td>▶ □ byte[24] @ 0×40a464f1</td><td>40</td><td></td></tr><tr><td></td><td></td><td>▶ □ byte[84] @ 0x40a40560@@</td><td></td><td></td></tr><tr><td></td><td></td><td>▶</td><td></td><td></td></tr><tr><td></td><td></td><td>▶ D byte[1572864] @ 0x408beab8 2@3.2@3.4@4.5A5.6A3.4?1.3>0.3>0.6A1.8C3</td><td></td><td></td></tr><tr><td>\/\/~4</td><td>\mathbf{t}</td><td>▶ <u>ii</u> byte[84] @ 0x408ba188</td><td> 96</td><td>96</td></tr><tr><td></td><td></td><td>▶ [] byte[960] @ 0×408b9468</td><td></td><td></td></tr><tr><td></td><td>tch the heap usage</td><td>▶ [] byte[84] @ 0x40859a48</td><td> 96</td><td></td></tr><tr><td></td><td></td><td>▶ D byte[960] @ 0x408b9528\/</td><td></td><td></td></tr><tr><td>_</td><td></td><td>> ></td><td>72</td><td></td></tr><tr><td></td><td>np the heap. Dig arou</td><td></td><td>208</td><td>208</td></tr></tbody></table>		

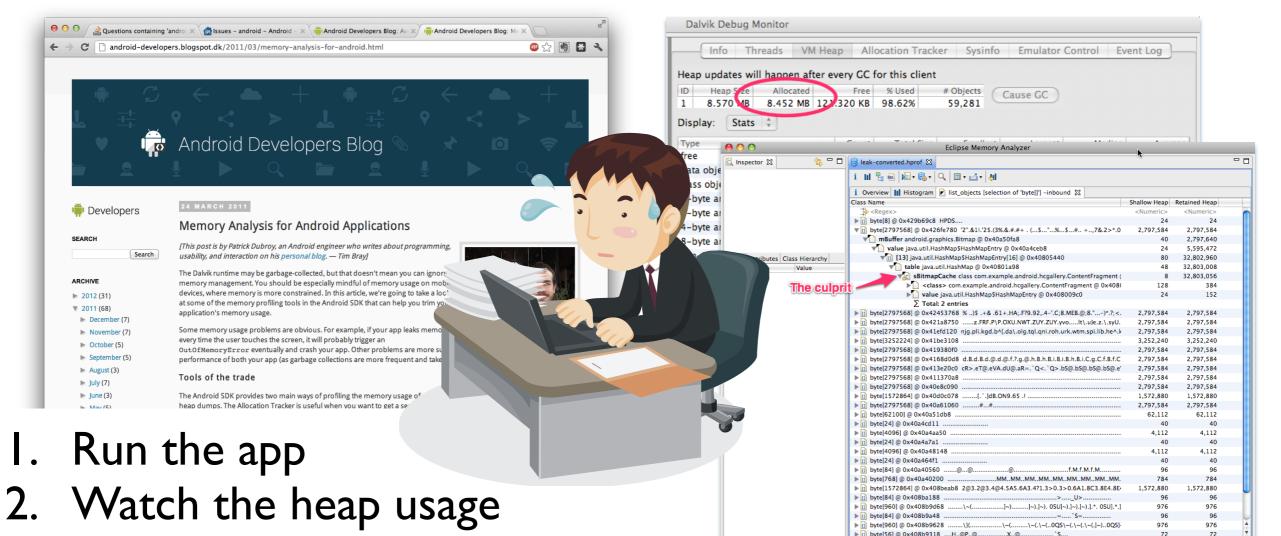
Cap. Dig and finally find the culprit!



- 3. Dump the heap. Dig around and finally find the culprit!
- 4. Commit a bugfix



36M of 81M



- 3. Dump the heap. Dig around and finally find the culprit!
- 4. Commit a bugfix
- 5. Bugfix is picked up by Fixr



▶ n byte[192] @ 0x408b91f8 .005.0RT 05U 05U 1TV 1TV 1TV 1UW 1UW 2VX 2WY 2X7 2X

36M of 81M



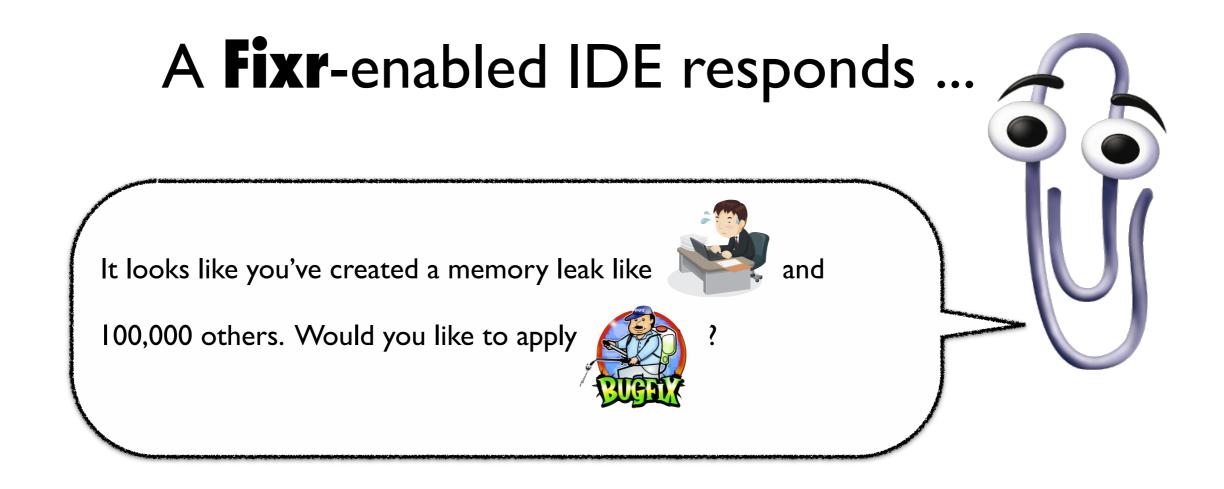
208

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

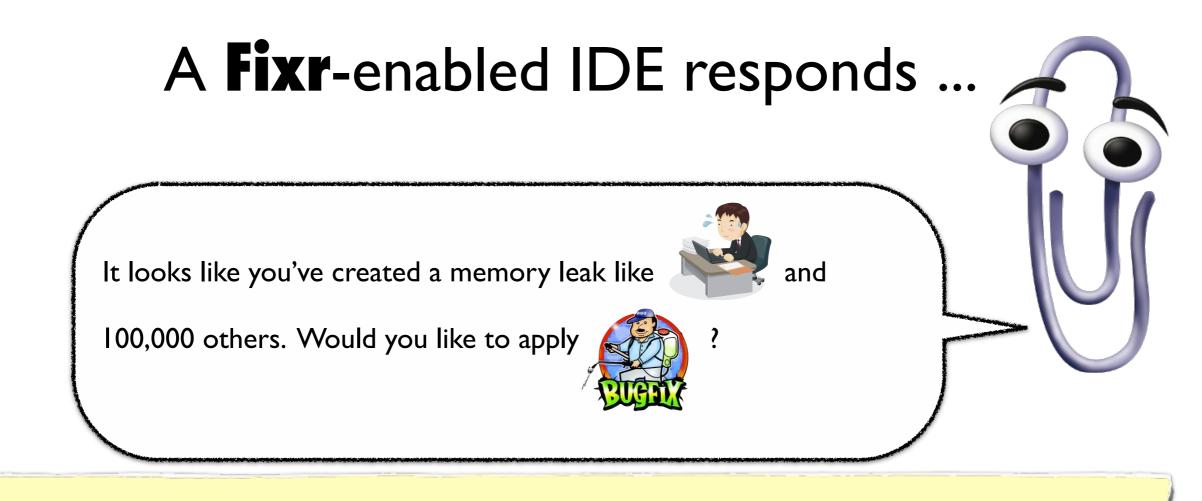


A **Fixr**-enabled IDE responds ...









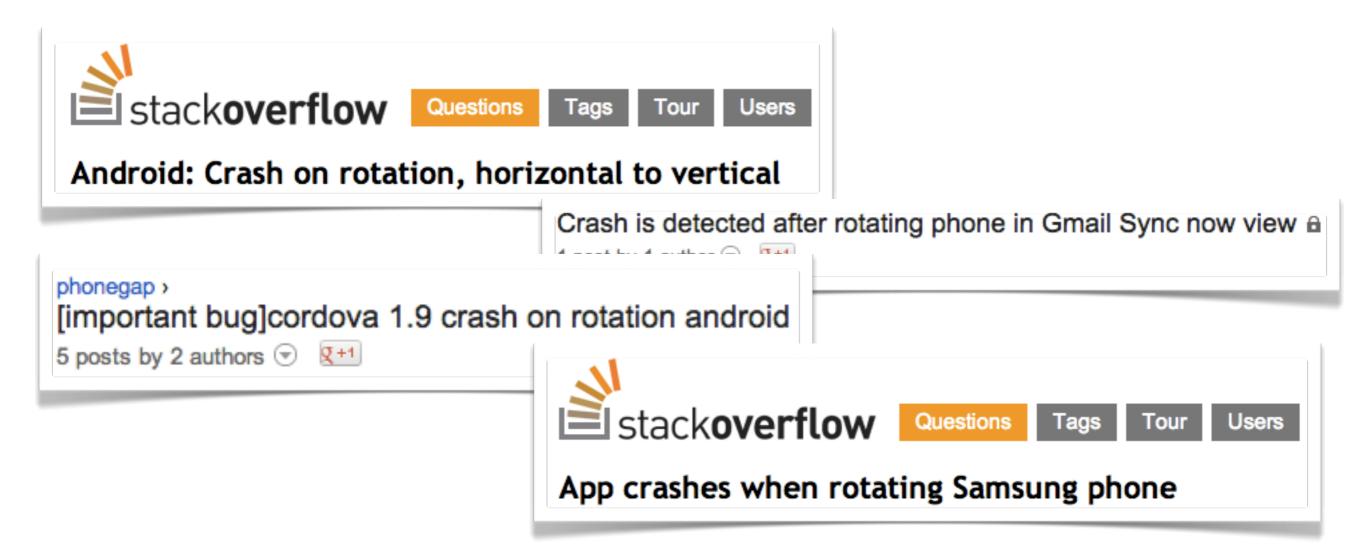
the bugfix is "transferred"



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

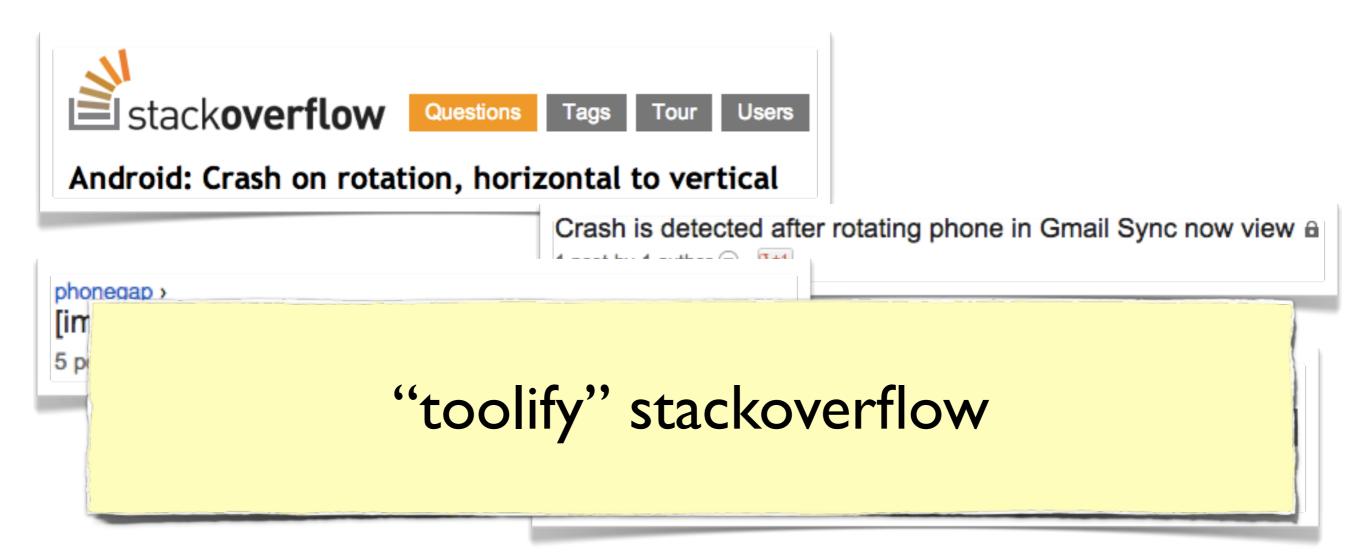
Leverage volume and variety of bugfixes made by the crowd of client app developers One Sentence Summary: Mine specifications of framework rules (indirectly) from bugfixes

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



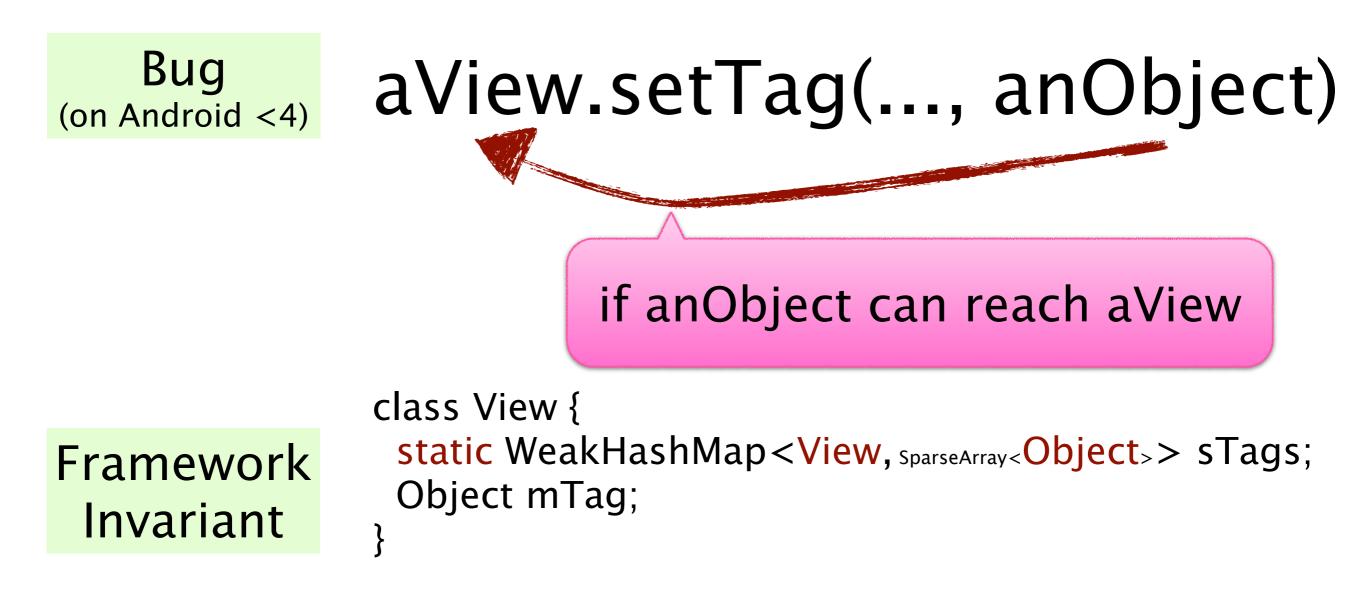
Simple motivating example: A well-understood Android bug

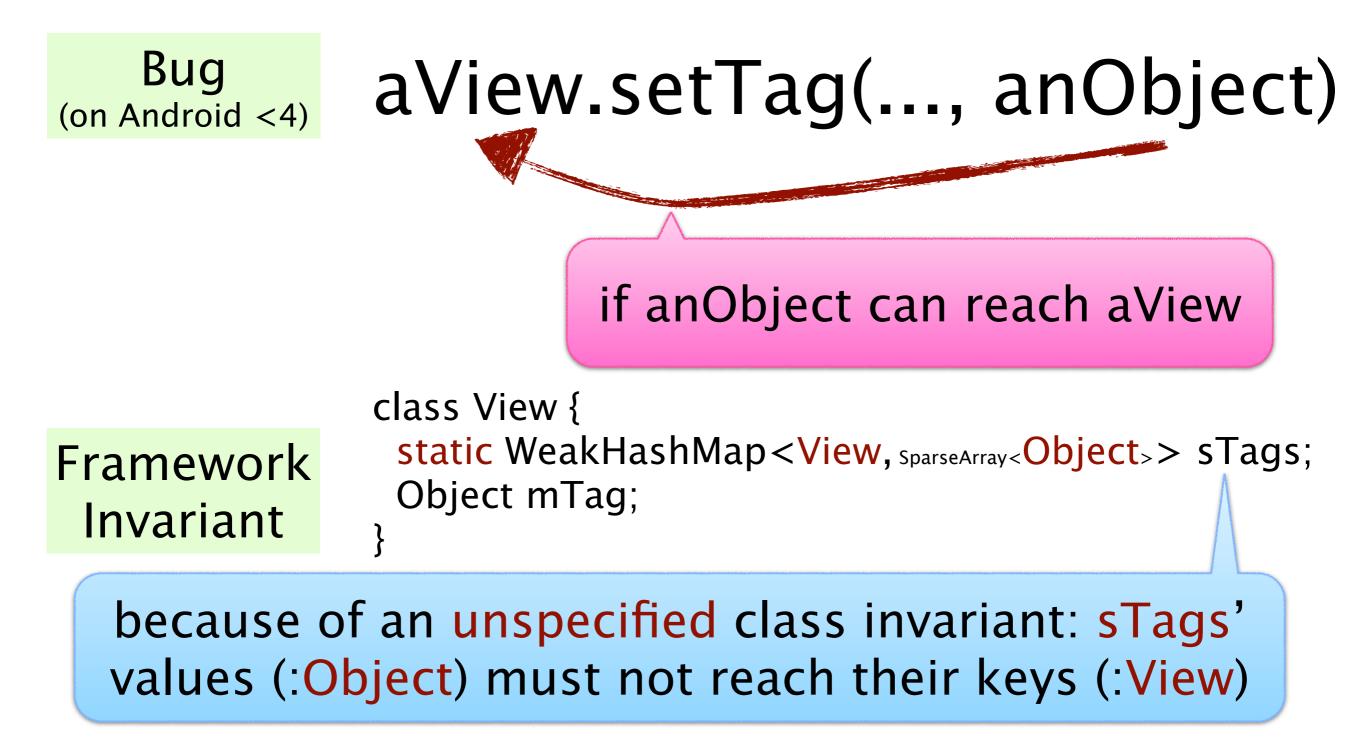
Simple motivating example: A well-understood Android bug

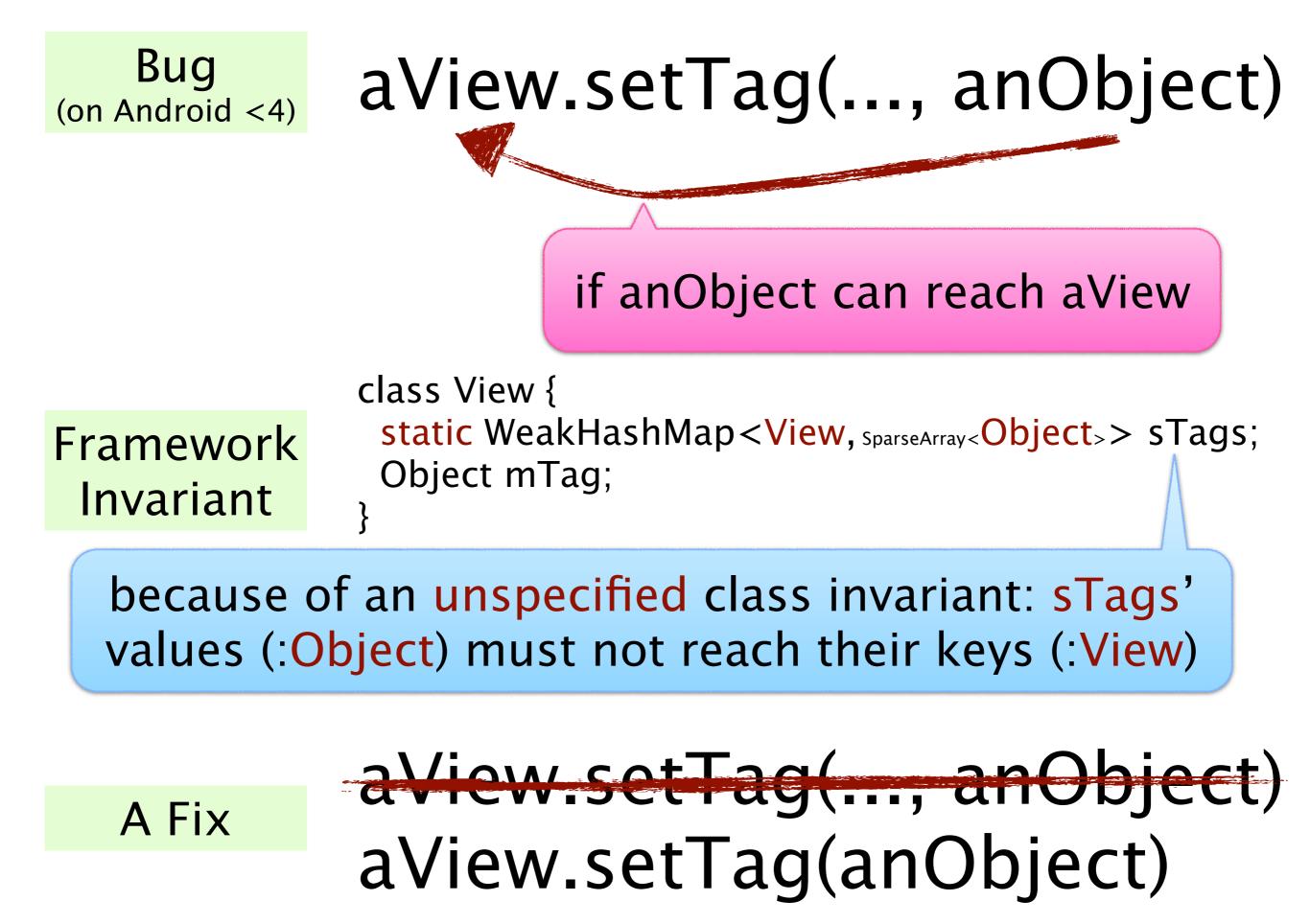
a common misuse of the framework

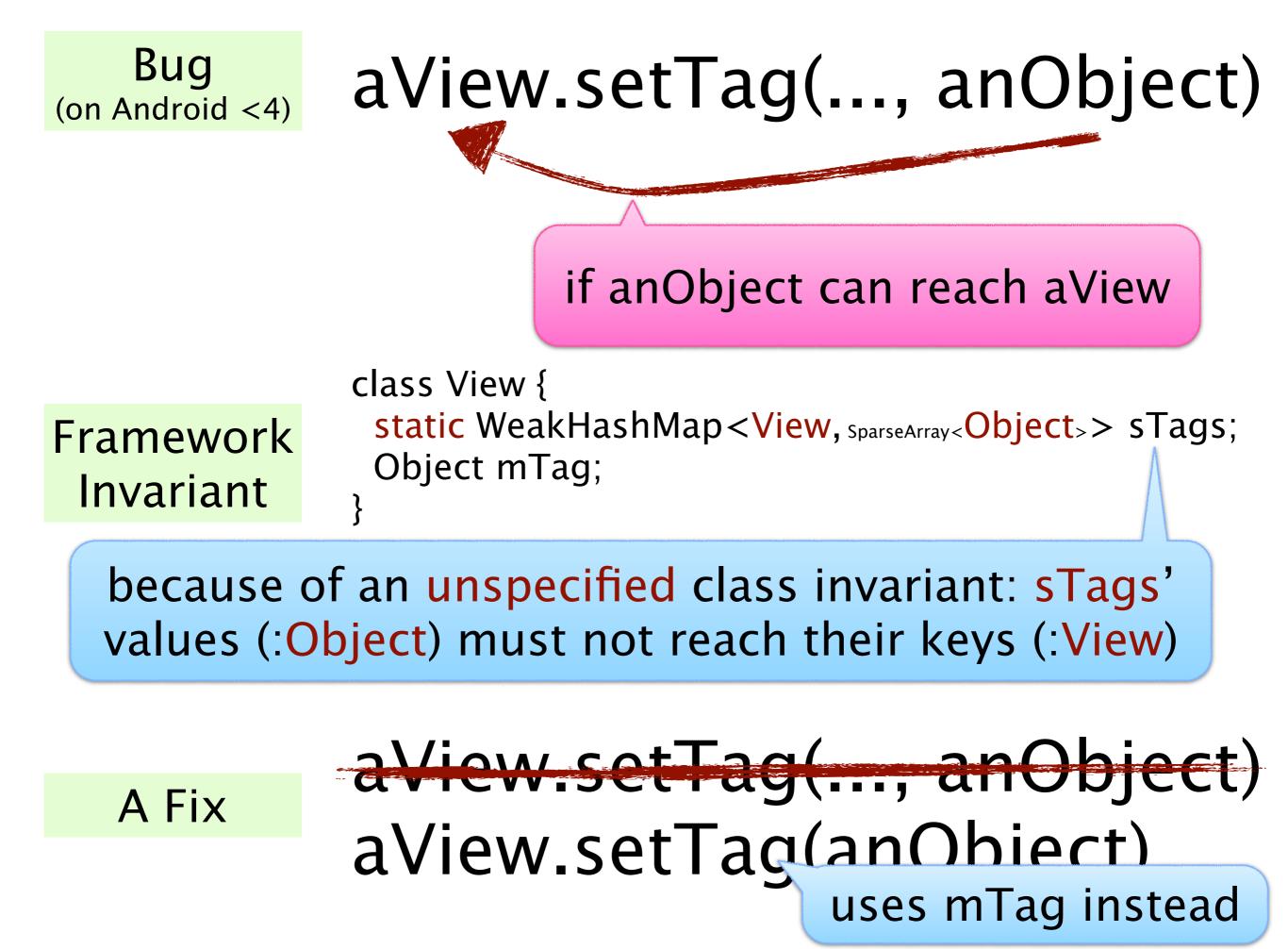
Bug (on Android <4) aView.setTag(..., anObject)

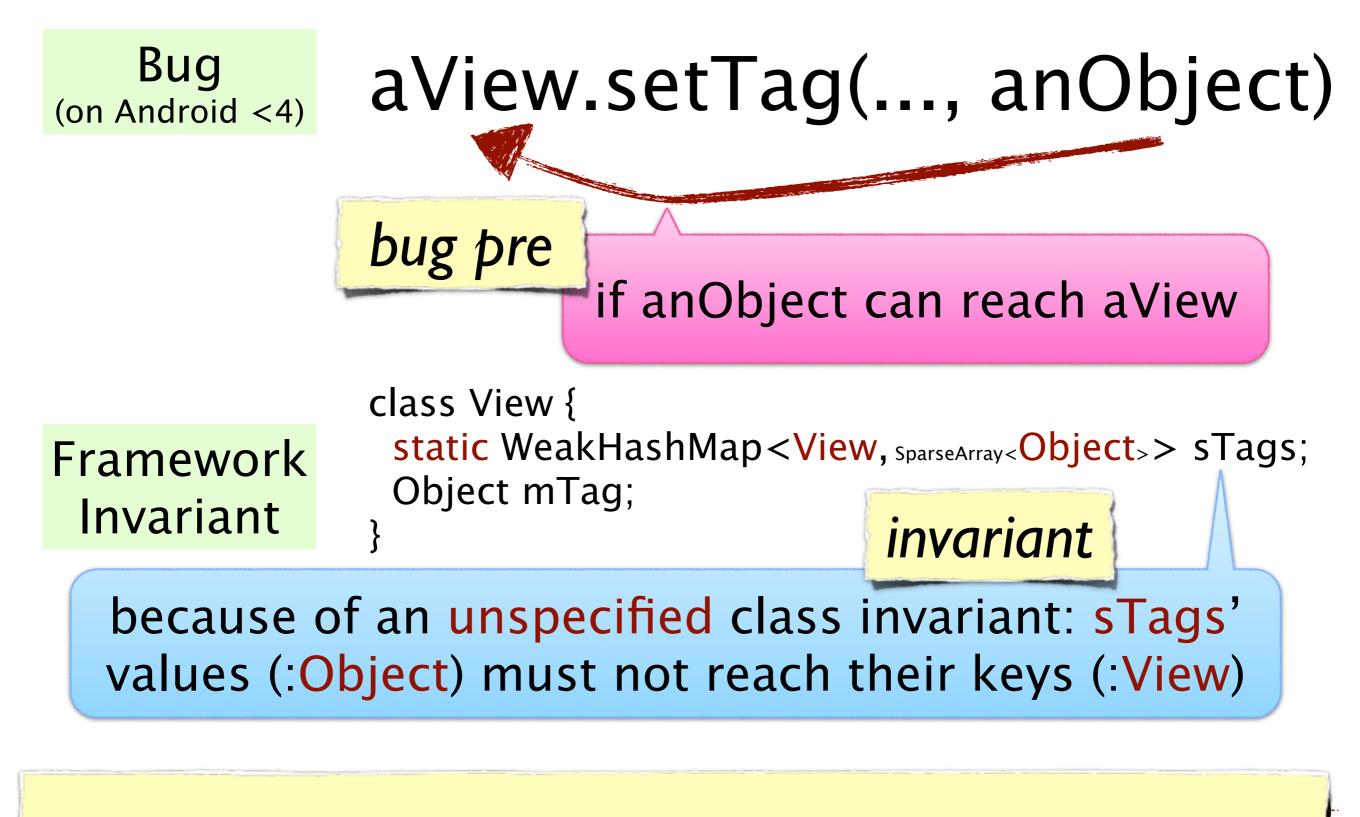






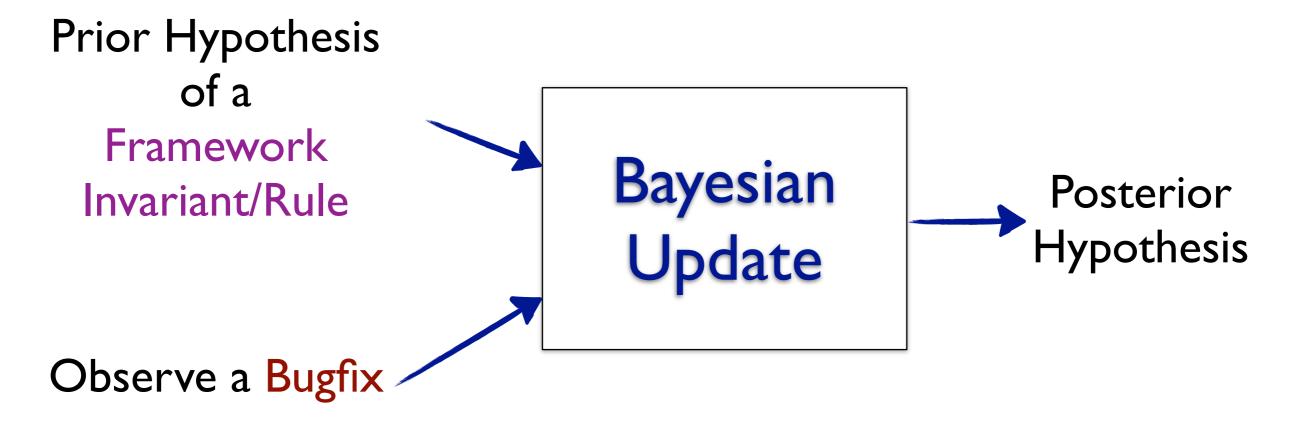




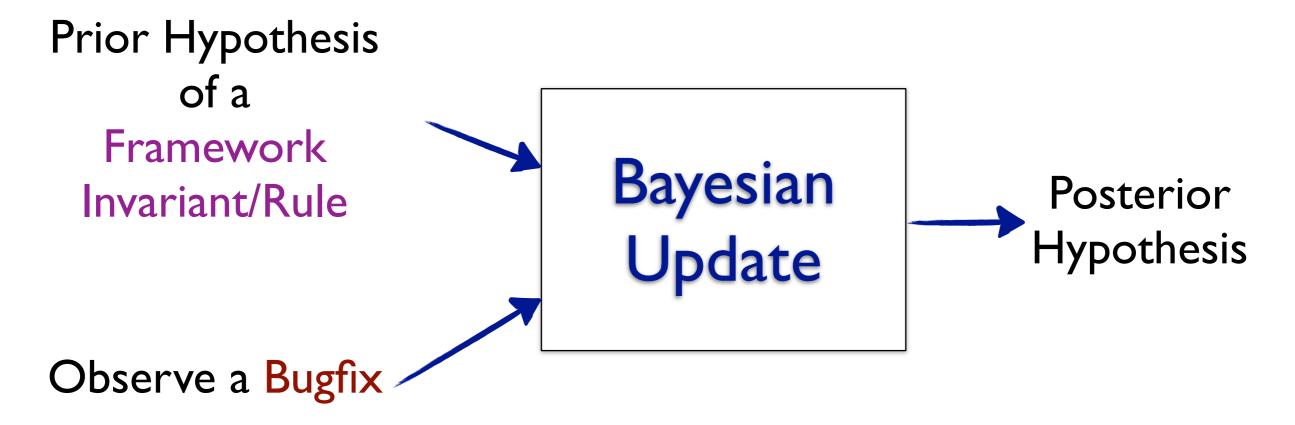


Goal: Produce this repair specification: bug pre, framework invariant, fix suggestion

Mining framework specifications with bugfixes



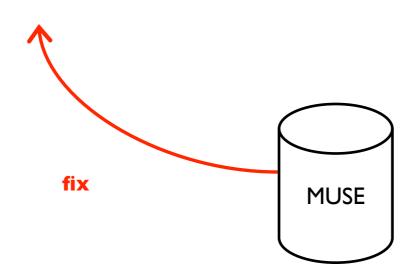
Mining framework specifications with bugfixes

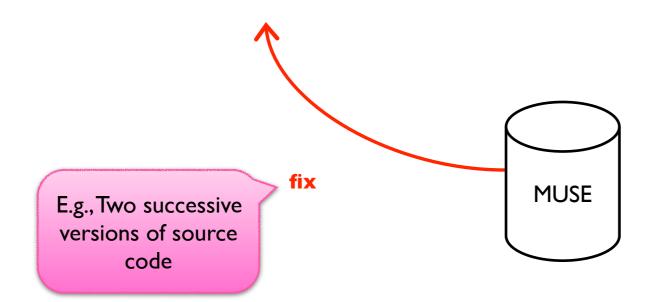


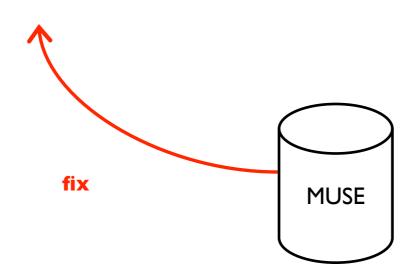


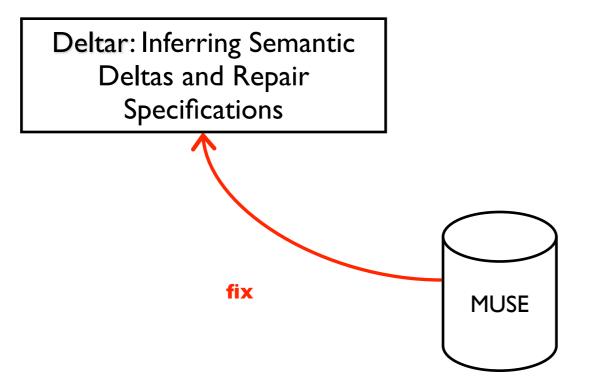
The **Fixr** Loop: Component by Component

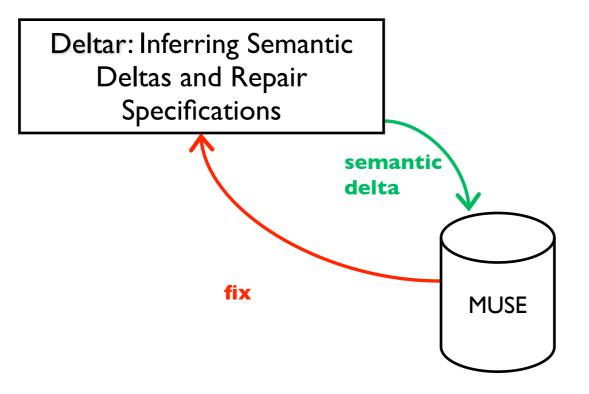


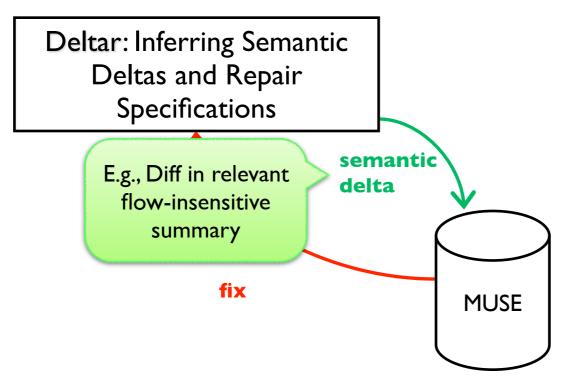


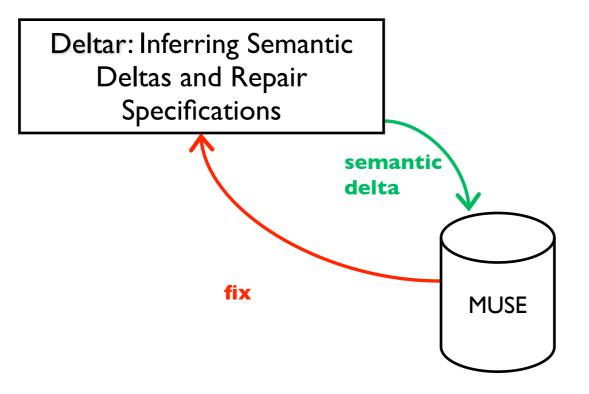


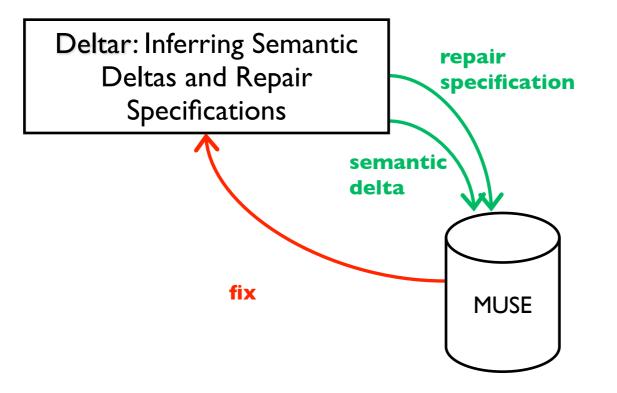


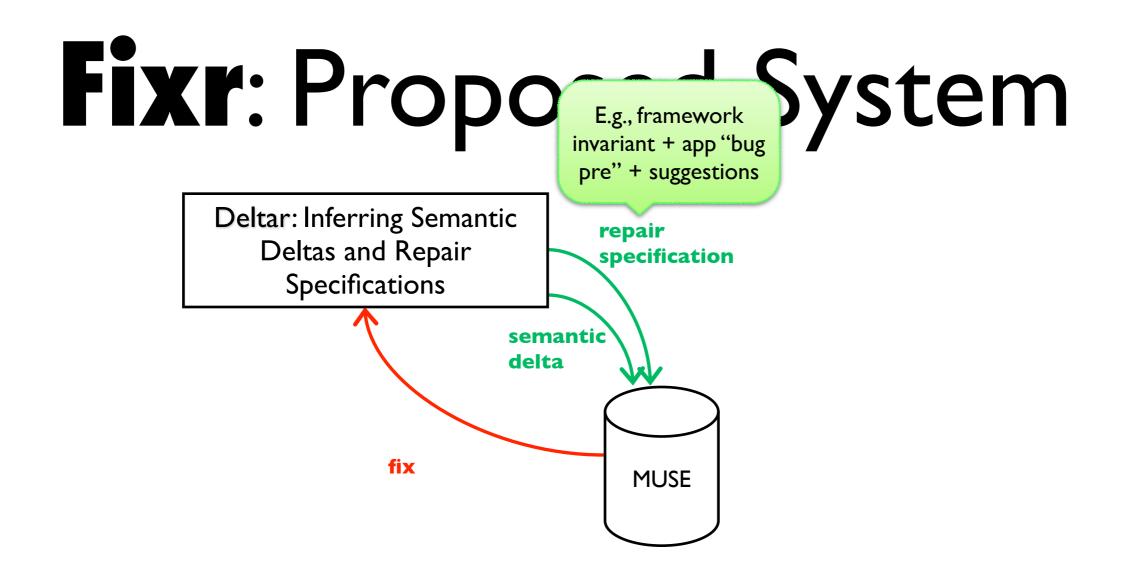


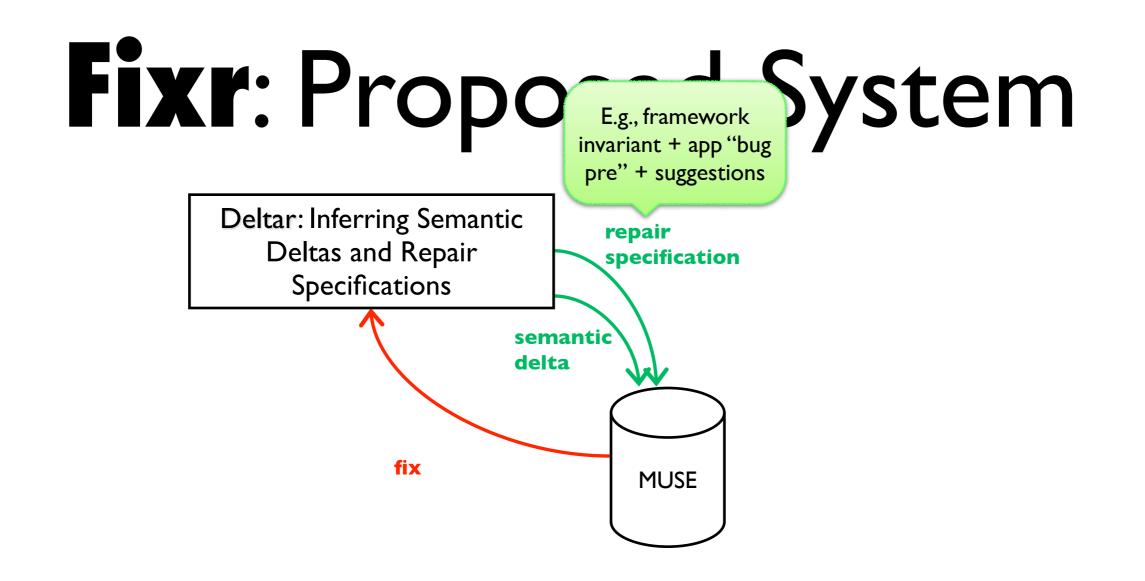












Component: Deltar maps fixes to semantic difference summaries and candidate repair specifications



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

A Fix

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

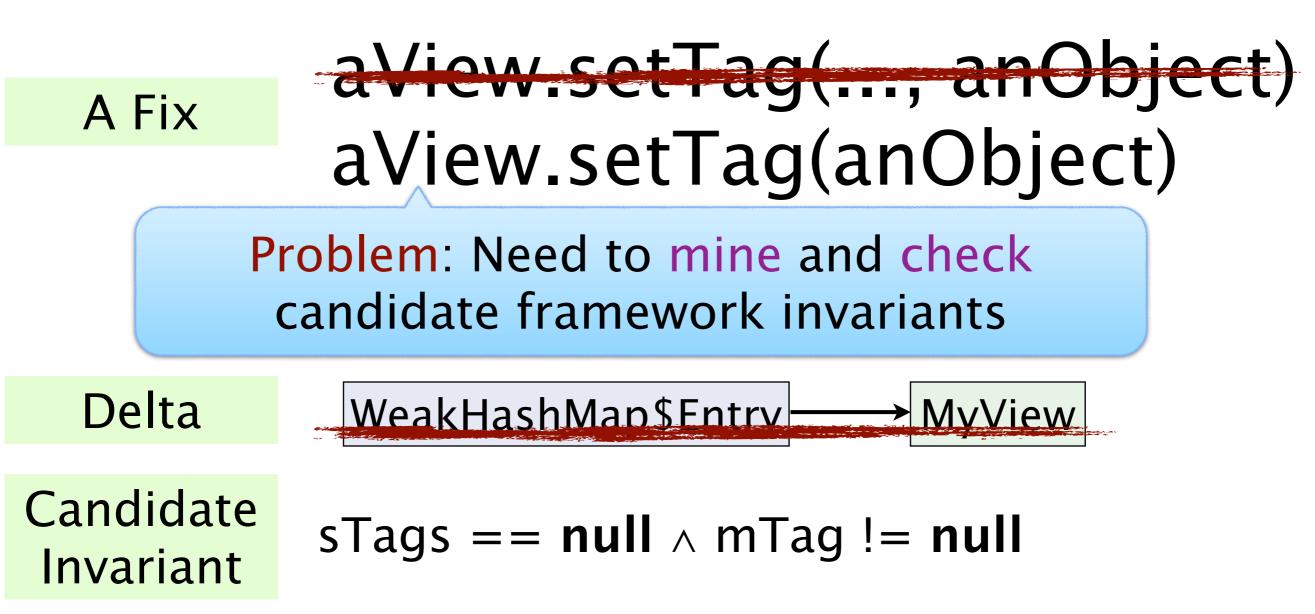
Problem: Need to mine and check candidate framework invariants



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

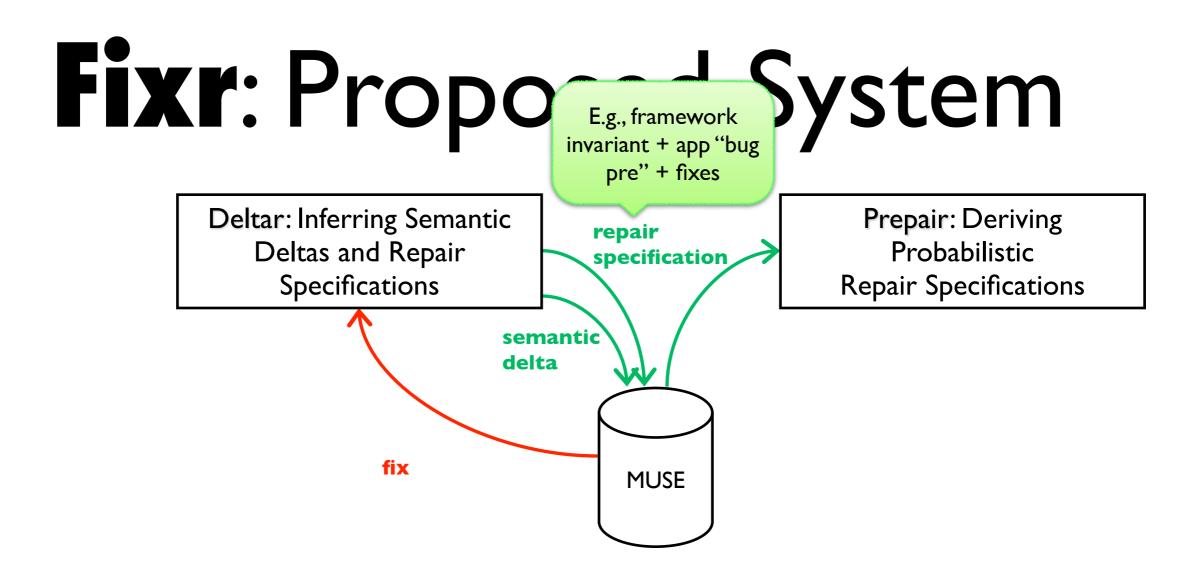
Problem: Need to mine and check candidate framework invariants

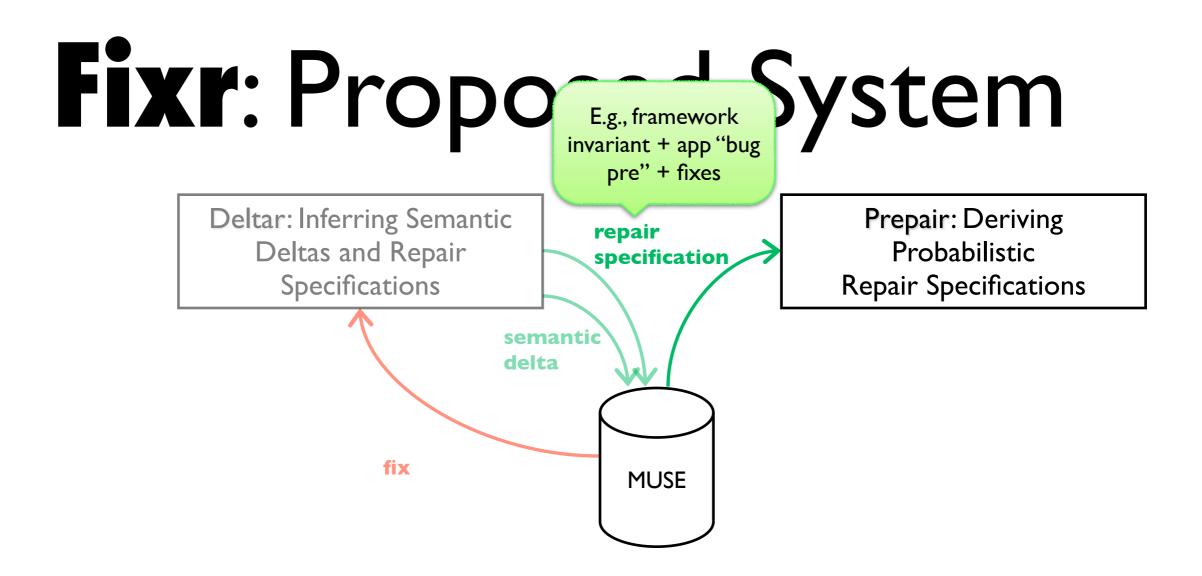


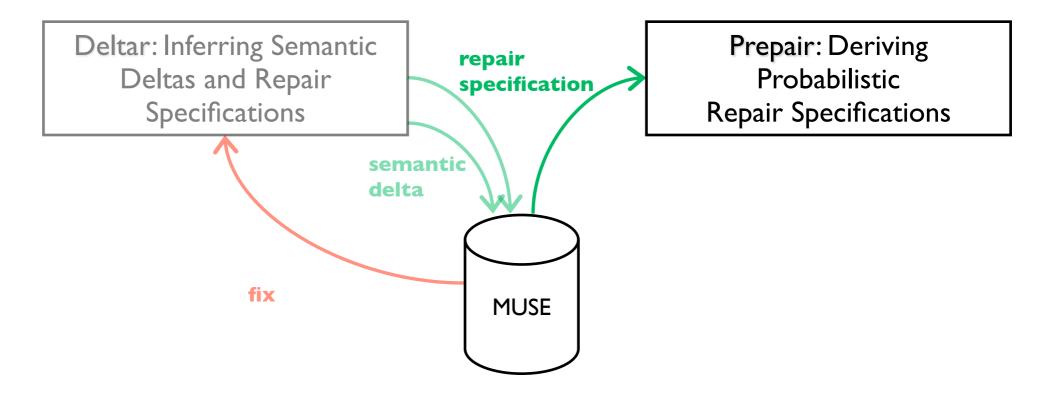


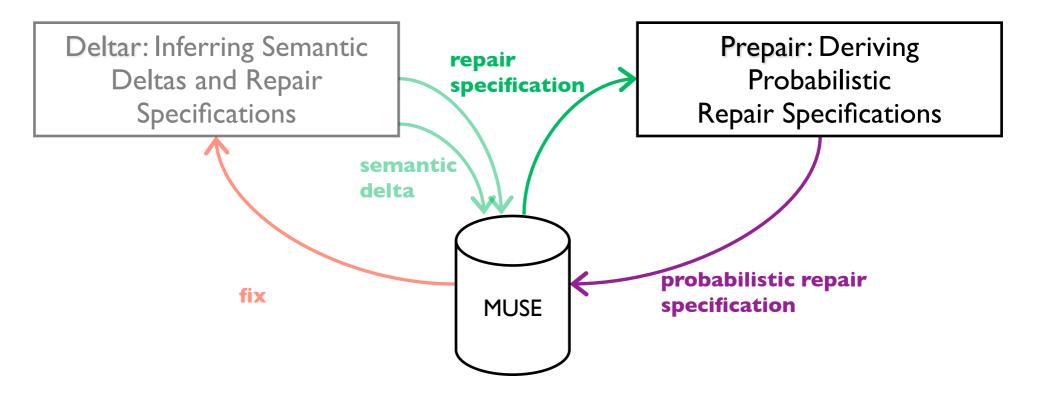
A Fix	aView.setTag(, anObject) aView.setTag(anObject)
Problem: Need to mine and check candidate framework invariants	
Delta	WeakHashMap\$Entry
Candidate Invariant	sTags == null ∧ mTag != null
_	

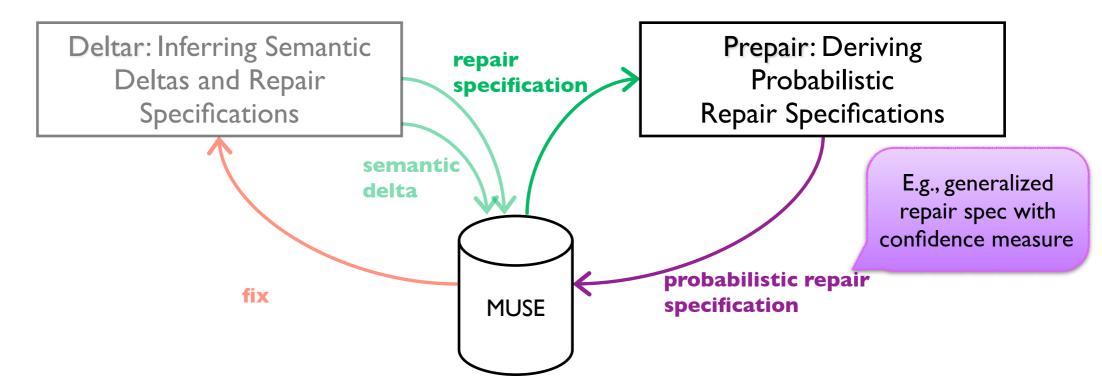
Approach: Refine coarse, global summaries and verify candidate invariant on fixed version (scalably with "almost everywhere type analysis")

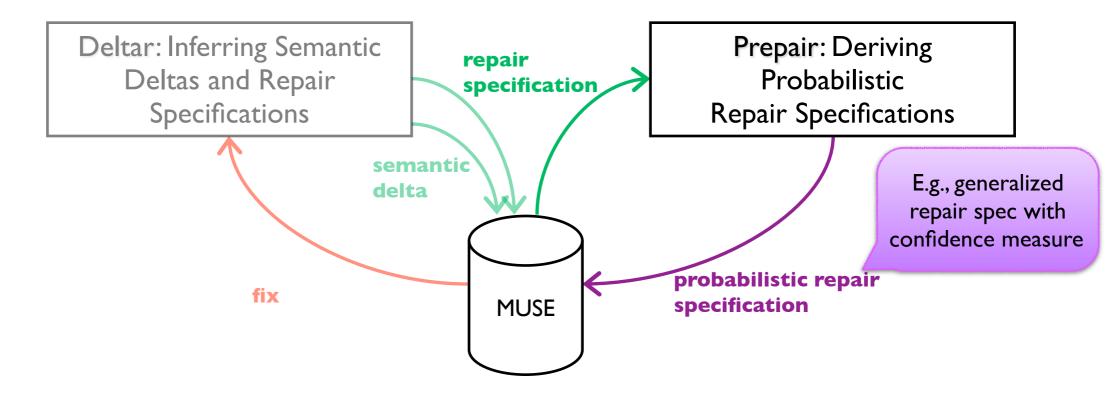












Component: Prepair reduces candidate repair specifications to generalized probabilistic repair specifications

sTags == null \land mTag != null

Candidate Invariant

Candidate Invariant

sTags == null \land mTag != null $\forall i. sTags[v][i] \Rightarrow v$

Candidate Invariant

sTags == null \land mTag != null $\forall i. sTags[v][i] \Rightarrow v$ $\forall v. sTags[v][0] \Rightarrow^* v$

Candidate Invariant sTags == null \land mTag != null $\forall i$. sTags[v][i] $\Rightarrow v$

∀v. sTags[v][0] →* v

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

Candidate Invariant sTags == null \land mTag != null $\forall i. sTags[v][i] \Rightarrow v$ $\forall v. sTags[v][0] \Rightarrow^* v$

Problem: Multiple (overly-specific or underspecified) 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.

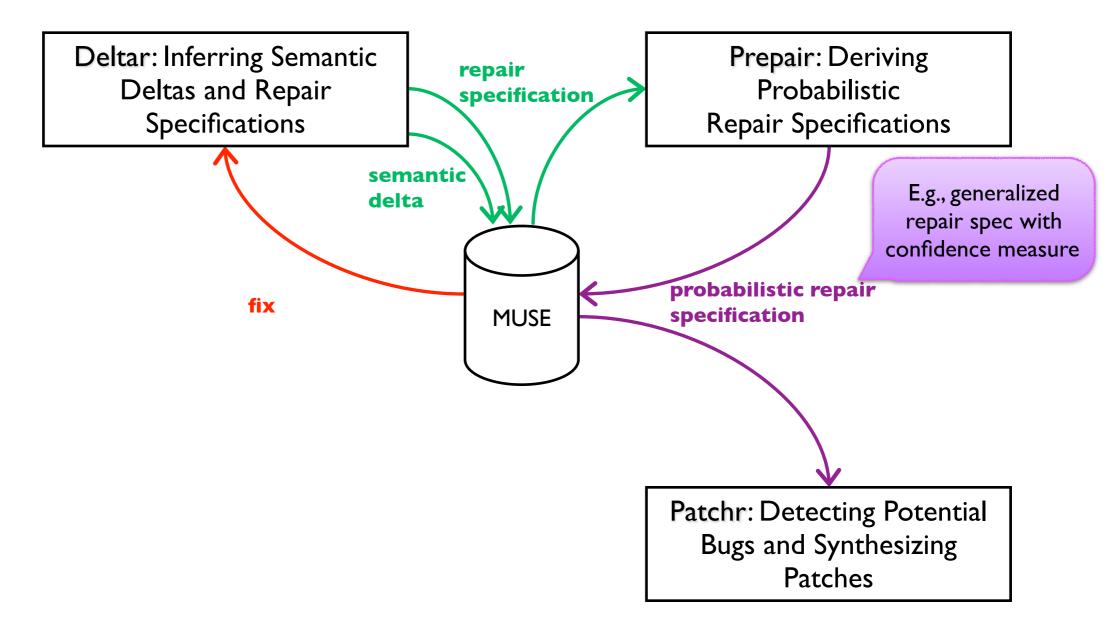
Candidate Invariant

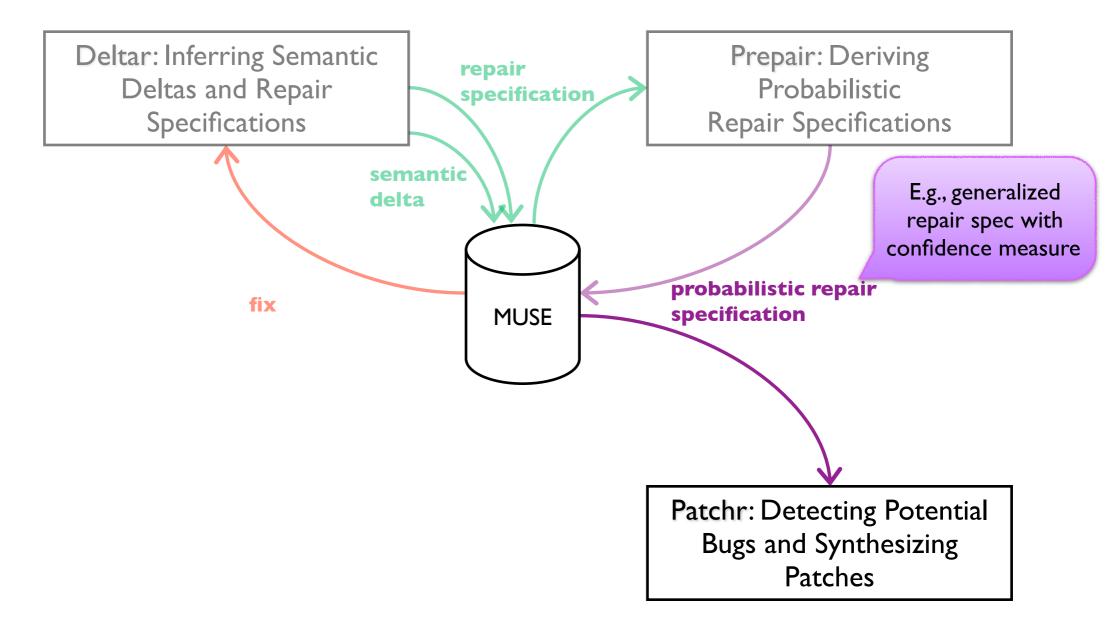
sTags == null
$$\land$$
 mTag != null
 $\forall i. sTags[v][i] \Rightarrow v$

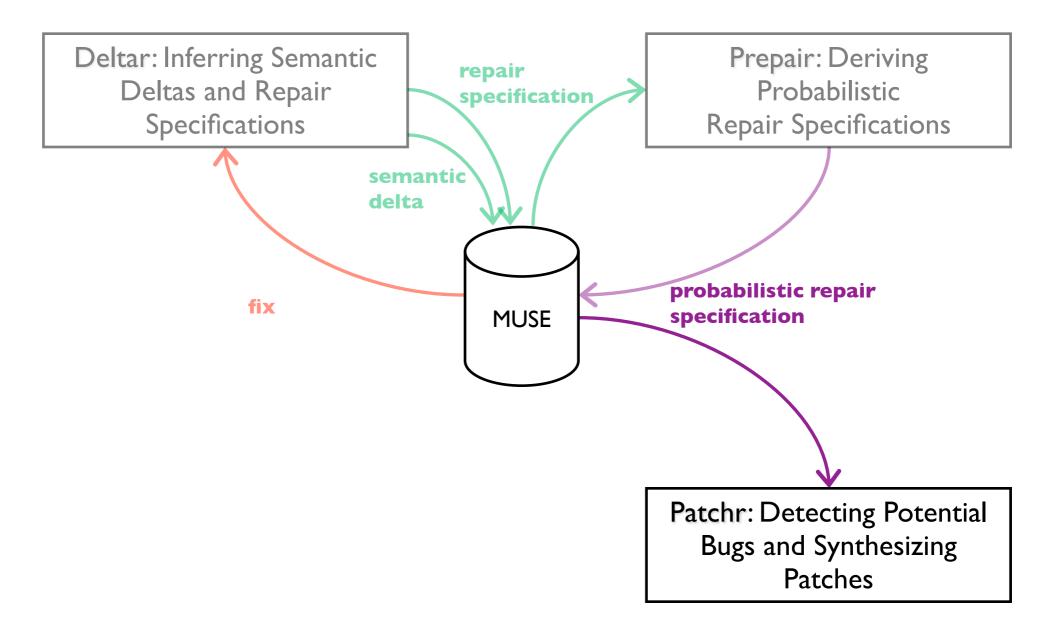
0.9

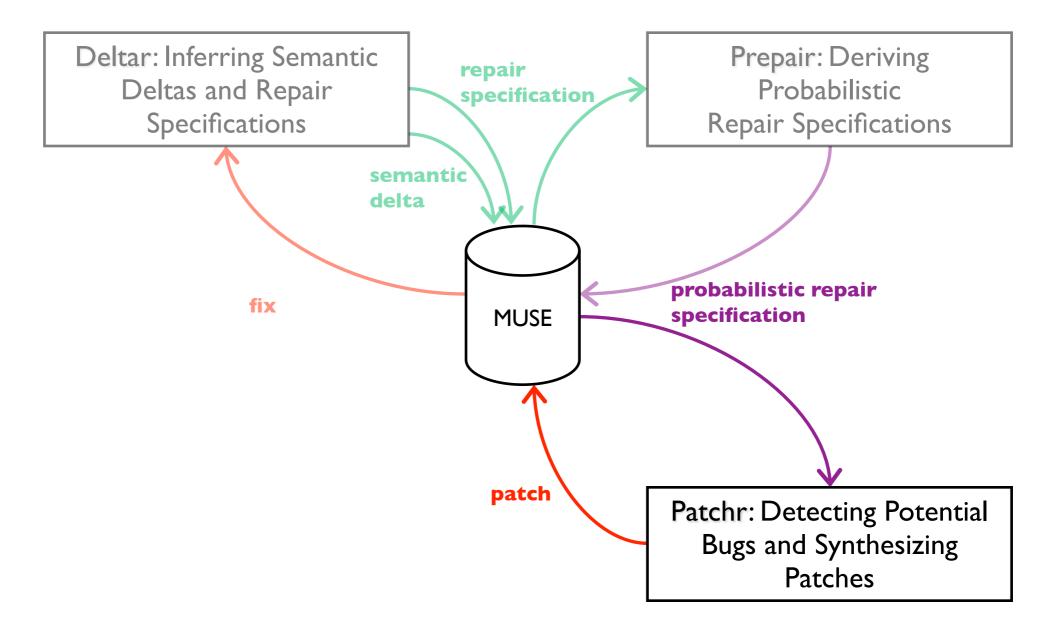
Problem: Multiple (overly-specific or underspecified) candidate repair specifications

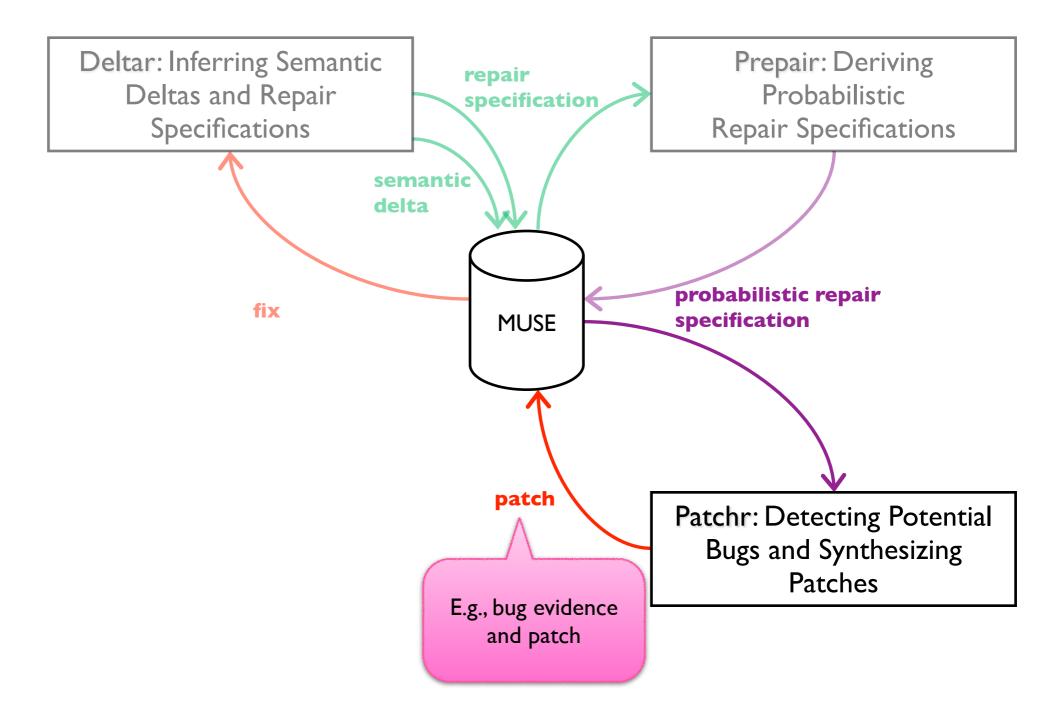
Approach: Static analysis as a form of Bayesian updating of priors to derive posteriors. Prevalence of fixes in MUSE database provides priors.

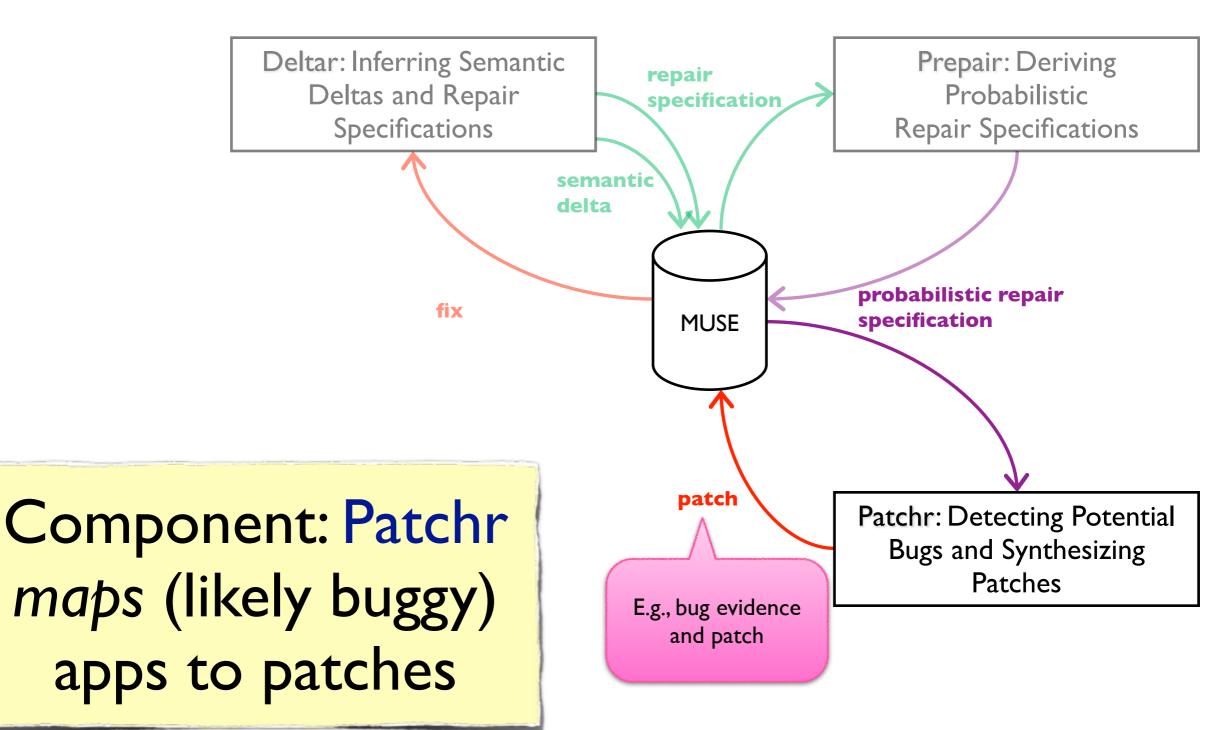












Candidate Invariant

sTags == null ^ mTag != null

0.9

Candidate Invariant

$$sTags == null \land mTag != null 0.9$$

Problem: How do we validate repair specifications?

Candidate Invariant

Problem: How do we validate repair specifications?

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

Candidate Invariant

0.9 sTags == null \land mTag != null

Problem: How do we validate repair specifications?

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

A Patch



Candidate Invariant

0.9 sTags == null \land mTag != null

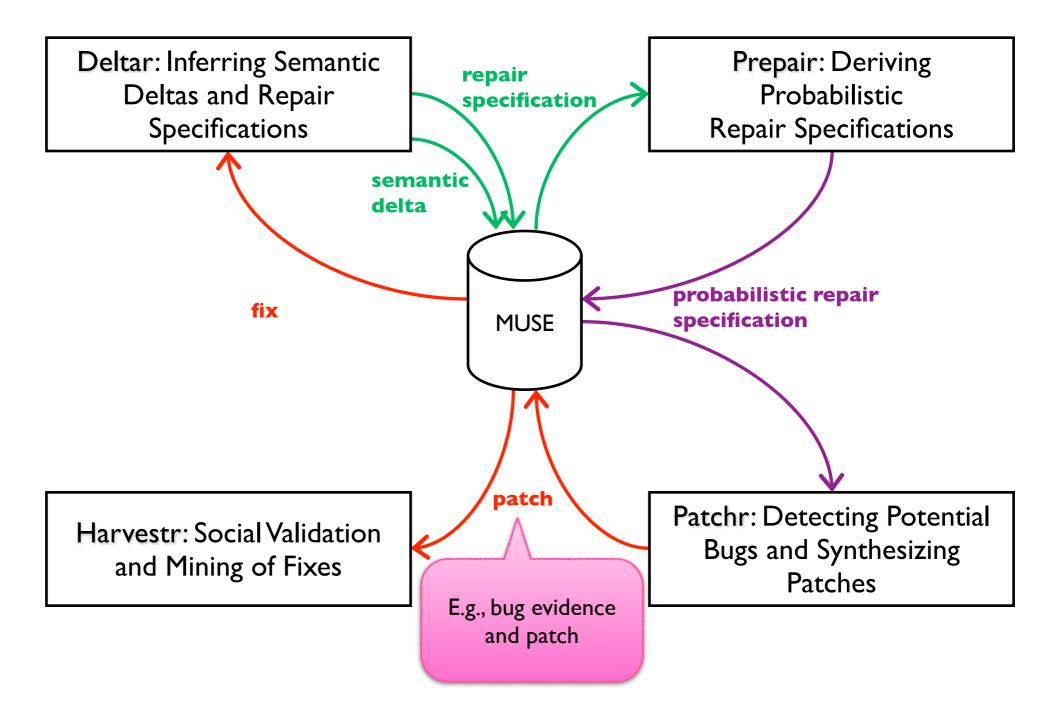
Problem: How do we validate repair specifications?

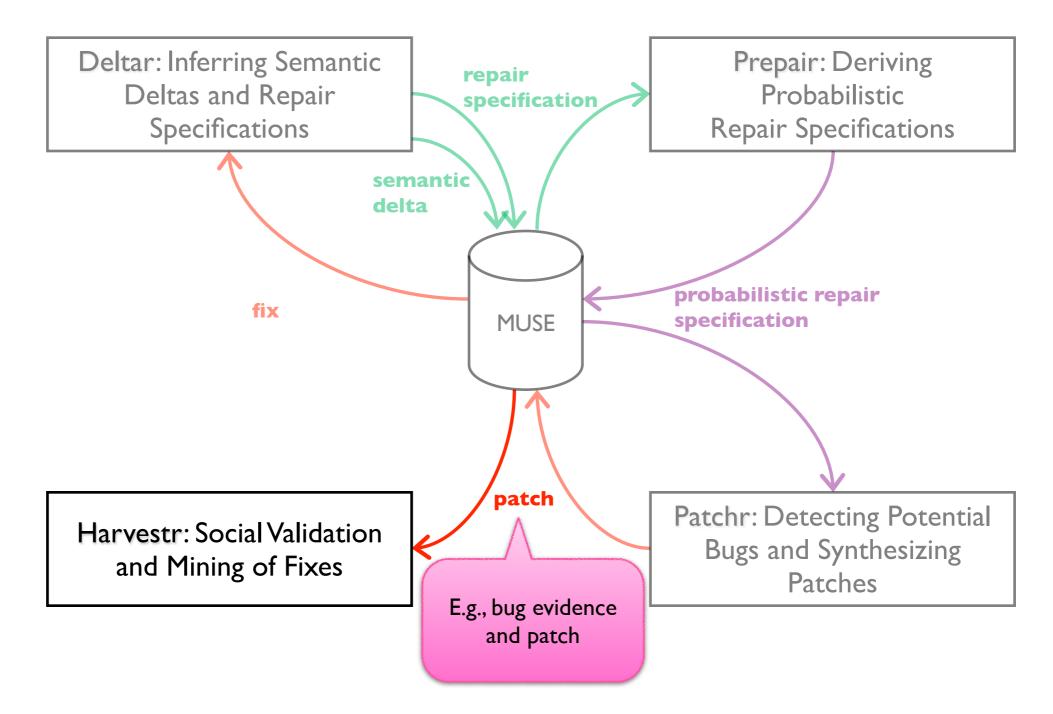
Approach: Synthesize patches for human validation (easier to understand and immediately useful)

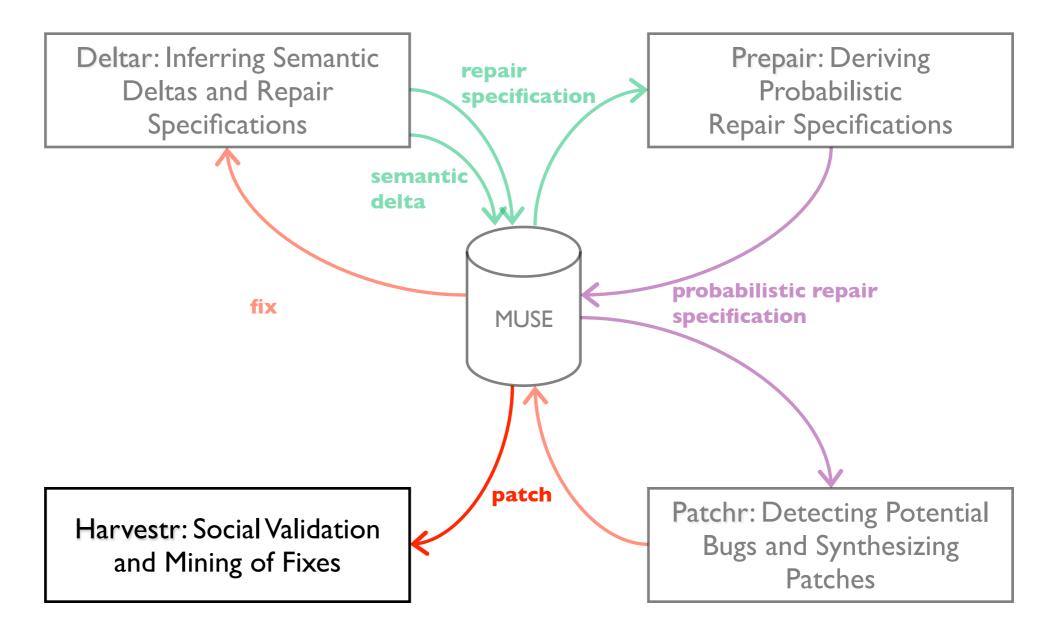
A Patch

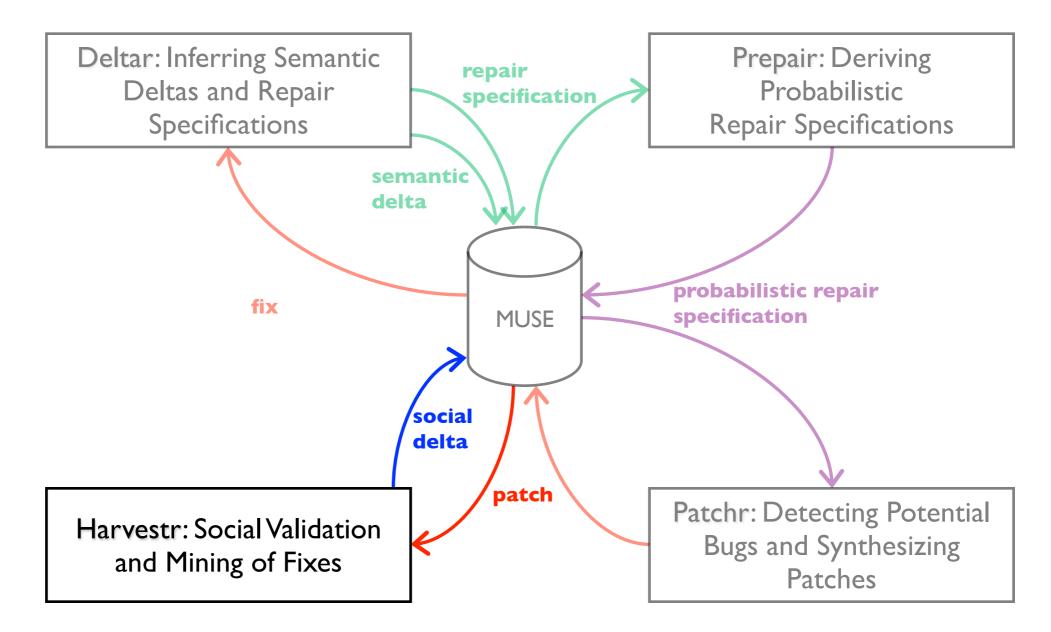
otherView.setTag(..., o) otherView.setTag(o)

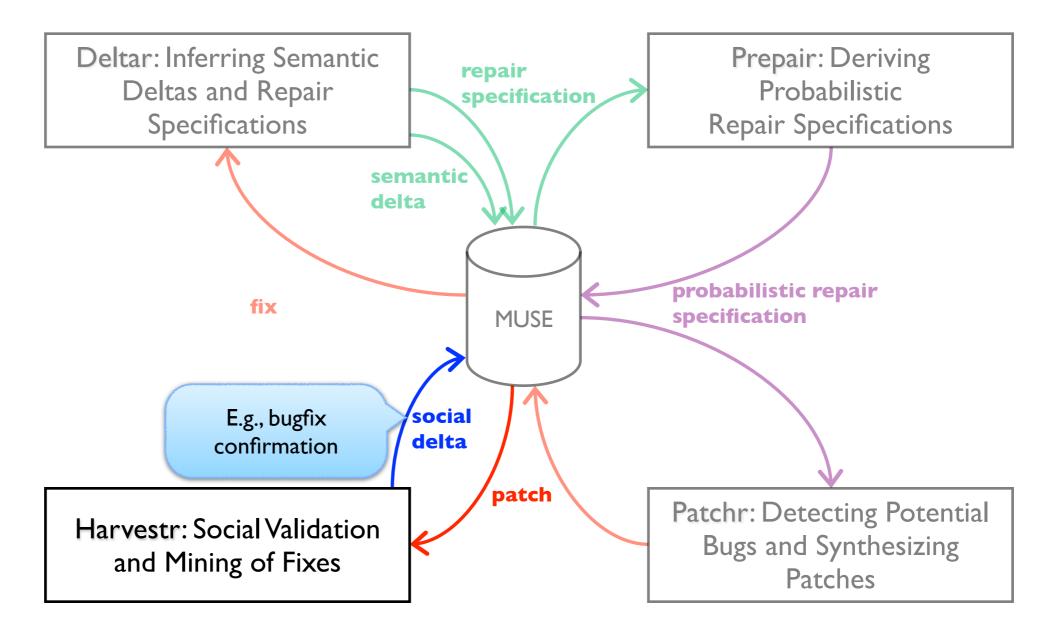
need to find apps satisfying "bug pre"

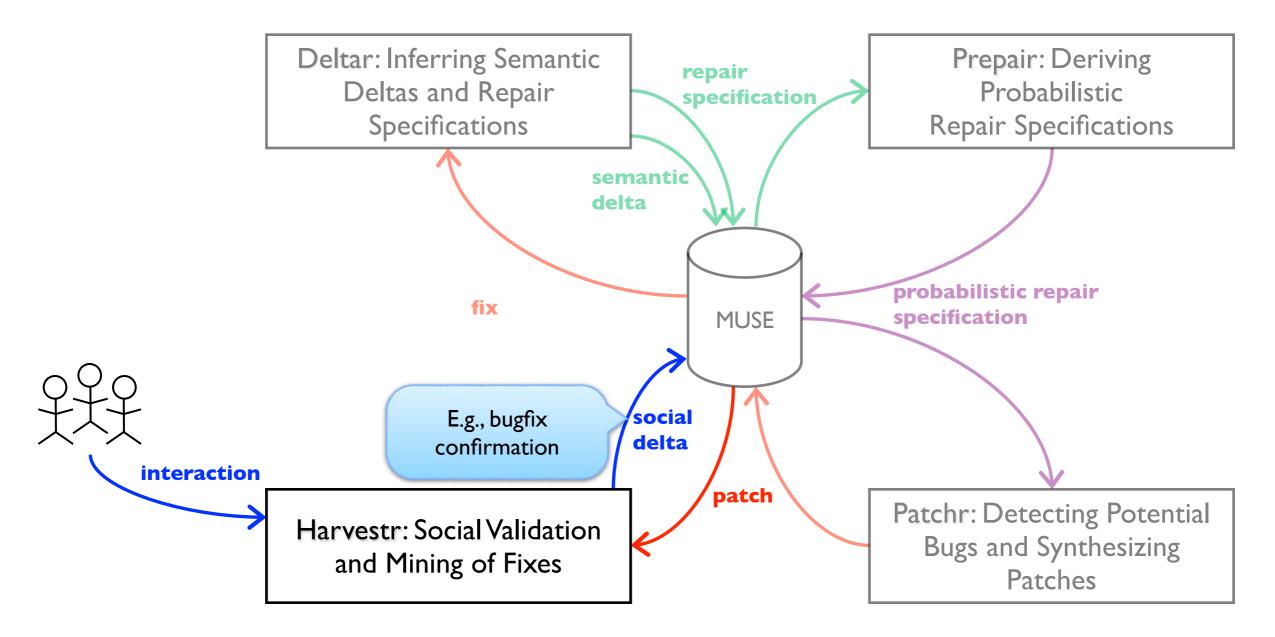




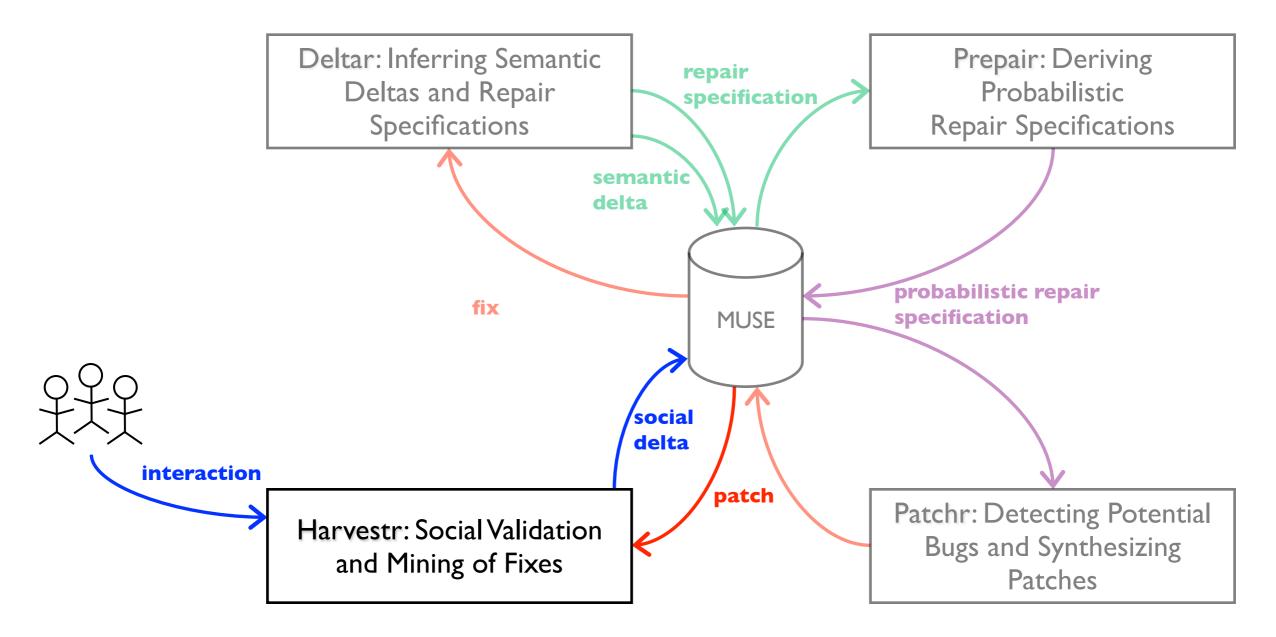


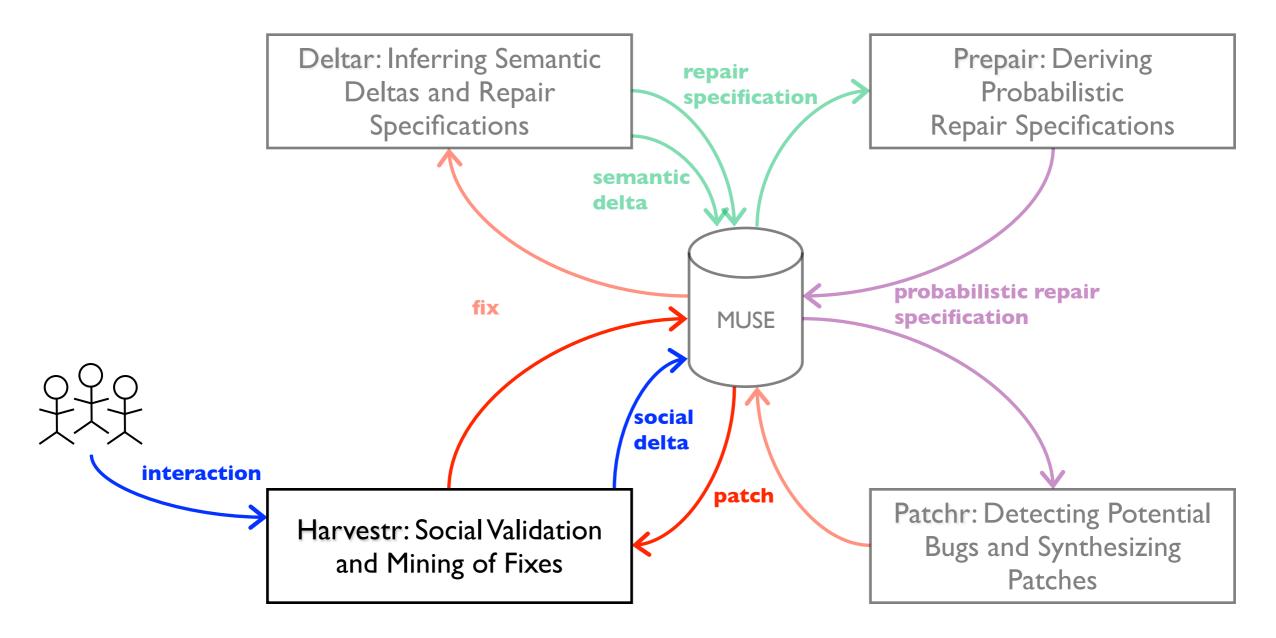


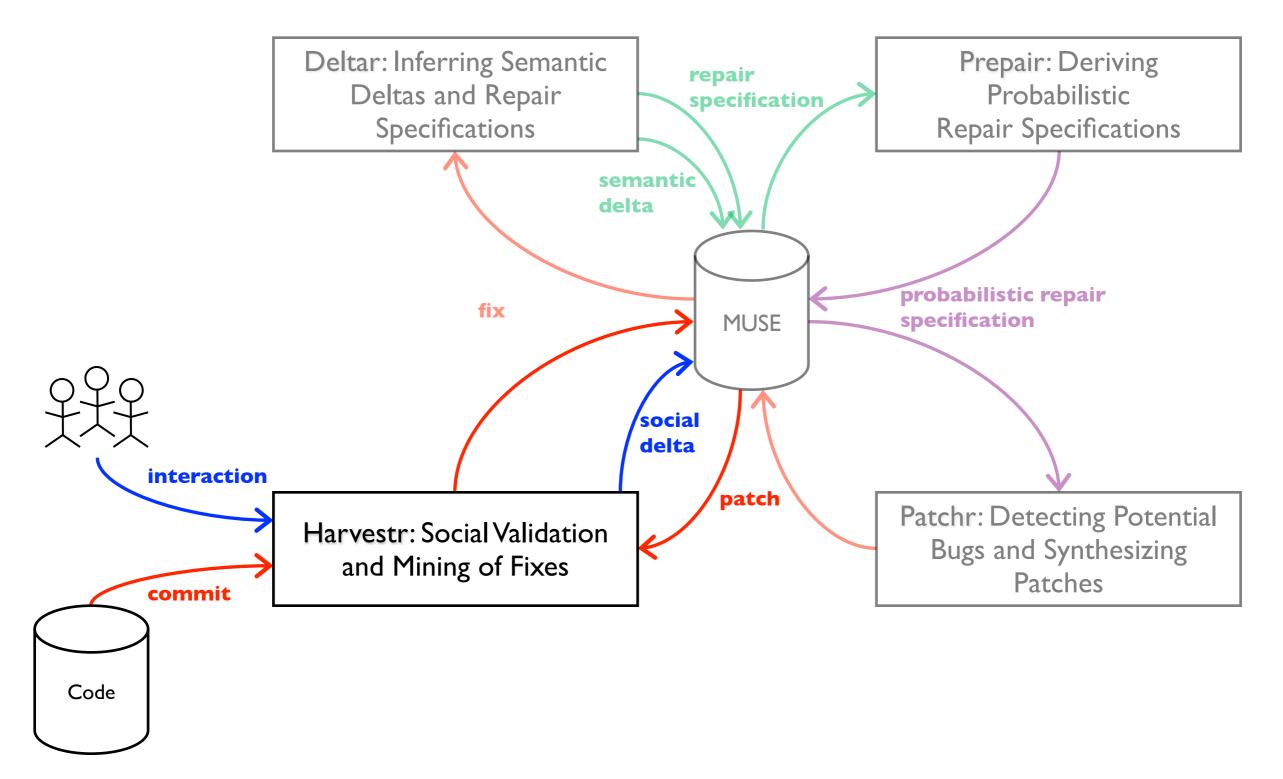




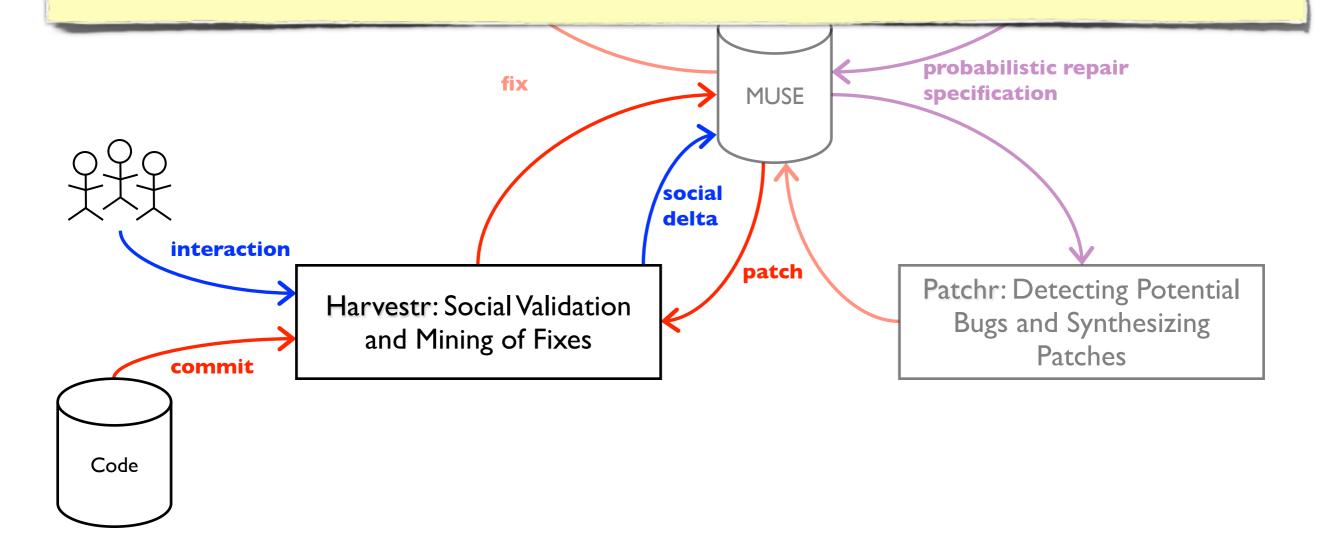
	Deltar: Inferring Semantic repa	ir Prepair: Deriving
	hsoukai / graspngo	Watch → 1 ★ Star 0 Ŷ Fork 1
	Fix setTag to avoid memory leak #1 Open fixrbot wants to merge 1 commit into nhsoukai:master from fixrbot:master Conversation • Commits 1 Files changed 1 Showing 1 changed file with 4 additions and 4 deletions.	Edit New issue
H H interact	E.g., bugfix social delta	
	Harvestr: Social Validation and Mining of Fixes	Patchr: Detecting Potential Bugs and Synthesizing Patches







Component: Harvestr maps commits and patches to candidate fixes



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

Browse code

choose an of themeoritag gettag	9				
Signed-off-by: Ian Lake ≺ian.hannibal.lake@gmail.co		2	ee -209,15 +210,7 ee public void oncreateopcionsmenu(tinai menu menu, tinai menuintiaten intiaten)		
	209	210	public View onCreateView(final LayoutInflater inflater,		
	210	211	final ViewGroup container, final Bundle savedInstanceState)		
0 · · · · · · · · · · · · · · · · · · ·	211	212	{		
ianhanniballake authored on Sep 16, 2012	212		 final View view = inflater.inflate(R.layout.fragment_view, container, 		
	213		- false);		
	214		<pre>- view.setTag(R.id.start_time, view.findViewById(R.id.start_time));</pre>		
	215		 view.setTag(R.id.start_date, view.findViewById(R.id.start_date)); 		
	216		 view.setTag(R.id.end_time, view.findViewById(R.id.end_time)); 		
	217		 view.setTag(R.id.end_date, view.findViewById(R.id.end_date)); 		
	218		 view.setTag(R.id.duration, view.findViewById(R.id.duration)); 		
	219		 view.setTag(R.id.note, view.findViewById(R.id.note)); 		
	220		- return view;		
		213	<pre>+ return inflater.inflate(R.layout.fragment_view, container, false);</pre>		
	221	214	}		
	222	04.5			

Closes GH-97 - Remove View.setTag/getTag Pattern			Browse code
Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.co< th=""><th>P-</th><th>ee -209,15 +210,7 ee public void oncreaceopcionsmenu(tinai menu menu, tinai menuintiacer intiacer)</th></ian.hannibal.lake@gmail.co<>		P-	ee -209,15 +210,7 ee public void oncreaceopcionsmenu(tinai menu menu, tinai menuintiacer intiacer)
	209	210	public View onCreateView(final LayoutInflater inflater,
	210	211	final ViewGroup container, final Bundle savedInstanceState)
,		212	{
ianhanniballake authored on Sep 16, 2012	212		 final View view = inflater.inflate(R.layout.fragment_view, container,
	213		- false);
	214		<pre>- view.setTag(R.id.start_time, view.findViewById(R.id.start_time));</pre>
			 view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
	216		<pre>- view.setTag(R.id.end_time, view.findViewById(R.id.end_time));</pre>
	217		 view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
	218		 view.setTag(R.id.duration, view.findViewById(R.id.duration));
	219		 view.setTag(R.id.note, view.findViewById(R.id.note));
	220		- return view;
			<pre>+ return inflater.inflate(R.layout.fragment_view, container, false);</pre>

Problem: How do we find relevant bugfixes?

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

Browse code

Owner

Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.co< th=""><th>2</th><th>@@ -209,15 +210,7 @@ public volu oncreateoprionsmenu(tinal menu menu, tinal menuintiater intiater)</th></ian.hannibal.lake@gmail.co<>		2	@@ -209,15 +210,7 @@ public volu oncreateoprionsmenu(tinal menu menu, tinal menuintiater intiater)
	209	210	public View onCreateView(final LayoutInflater inflater,
2 master 🗞 r2.0-beta-2 r1.8.9	210	211	final ViewGroup container, final Bundle savedInstanceState)
		212	{
ianhanniballake authored on Sep 16, 2012			 final View view = inflater.inflate(R.layout.fragment_view, container,
	213		- false);
	214		 view.setTag(R.id.start_time, view.findViewById(R.id.start_time));
	215		 view.setTag(R.id.start_date, view.findViewById(R.id.start_date));
	216		<pre>- view.setTag(R.id.end_time, view.findViewById(R.id.end_time));</pre>
	217		<pre>- view.setTag(R.id.end_date, view.findViewById(R.id.end_date));</pre>
	218		 view.setTag(R.id.duration, view.findViewById(R.id.duration));
	219		 view.setTag(R.id.note, view.findViewById(R.id.note));
	220		- return view;
			<pre>+ return inflater.inflate(R.layout.fragment_view, container, false);</pre>

Problem: How do we find relevant bugfixes?



ianhanniballake commented on Sep 16, 2012

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."

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

Browse code

Owner

	-		
Signed-off-by: Ian Lake <ian.hannibal.lake@gmail.co< th=""><th>2</th><th>ee -203,15 +210,7 ee public volu oncreateoptionsmenu(tinal menu menu, tinal menuintiater inflater)</th></ian.hannibal.lake@gmail.co<>		2	ee -203,15 +210,7 ee public volu oncreateoptionsmenu(tinal menu menu, tinal menuintiater inflater)
	209	210	public View onCreateView(final LayoutInflater inflater,
2 master 🗞 r2.0-beta-2 r1.8.9	210	211	final ViewGroup container, final Bundle savedInstanceState)
		212	{
ianhanniballake authored on Sep 16, 2012			 final View view = inflater.inflate(R.layout.fragment_view, container,
e	213		- false);
	214		<pre>- view.setTag(R.id.start_time, view.findViewById(R.id.start_time));</pre>
	215		<pre>- view.setTag(R.id.start_date, view.findViewById(R.id.start_date));</pre>
	216		<pre>- view.setTag(R.id.end_time, view.findViewById(R.id.end_time));</pre>
	217		 view.setTag(R.id.end_date, view.findViewById(R.id.end_date));
	218		 view.setTag(R.id.duration, view.findViewById(R.id.duration));
	219		 view.setTag(R.id.note, view.findViewById(R.id.note));
	220	Δ	- return view;
			<pre>+ return inflater.inflate(R.layout.fragment_view, container, false);</pre>

Problem: How do we find relevant bugfixes?

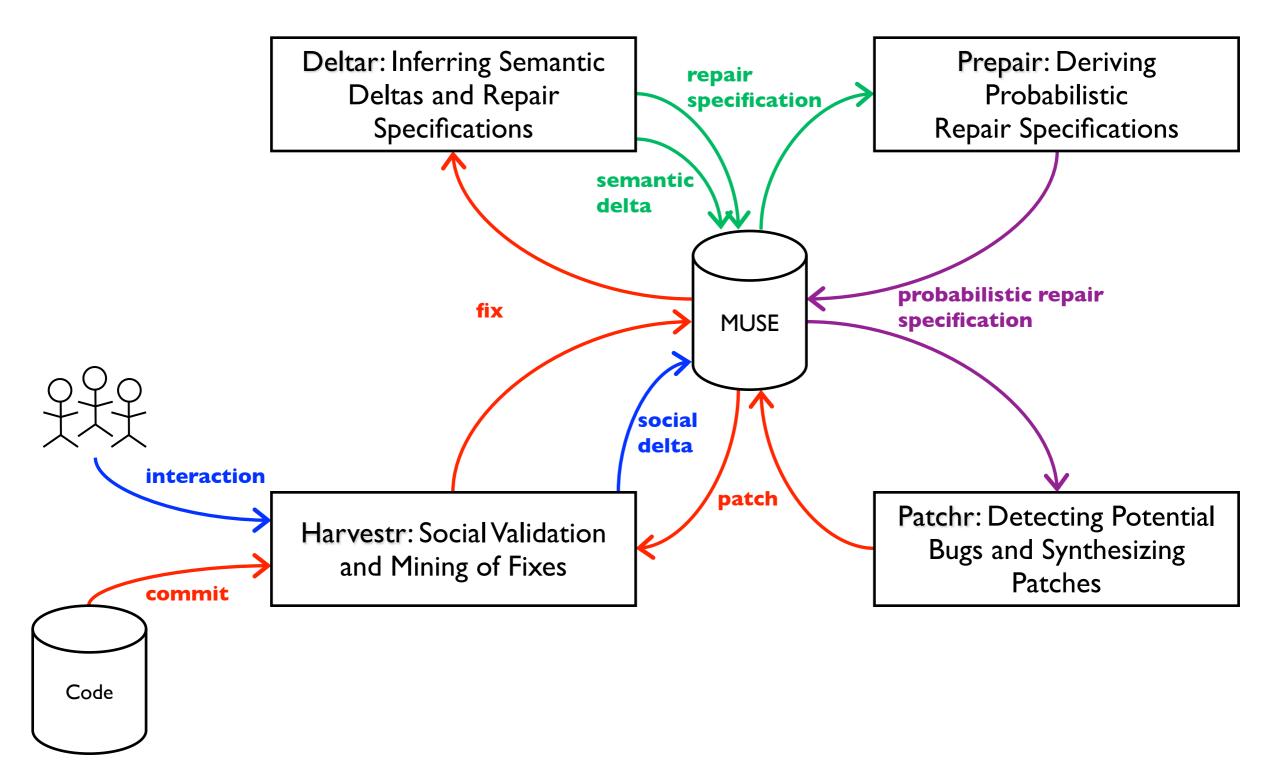


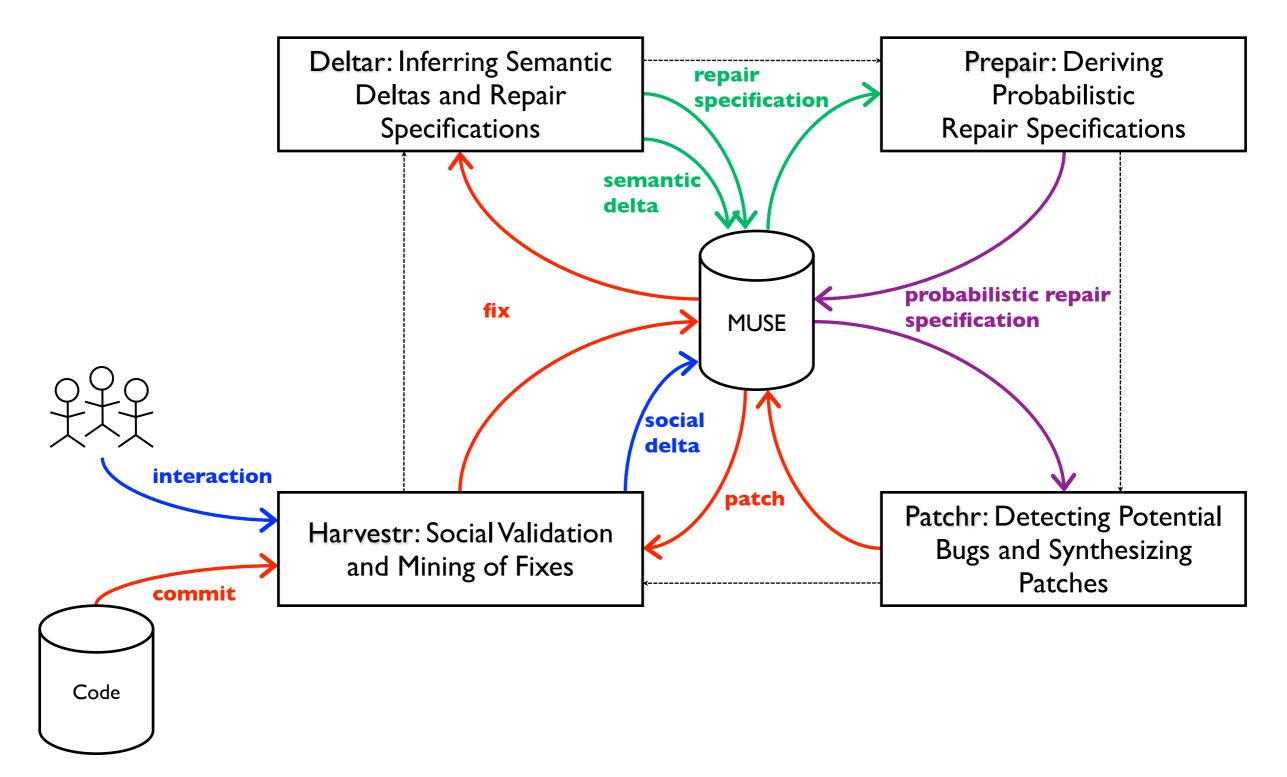
ianhanniballake commented on Sep 16, 2012

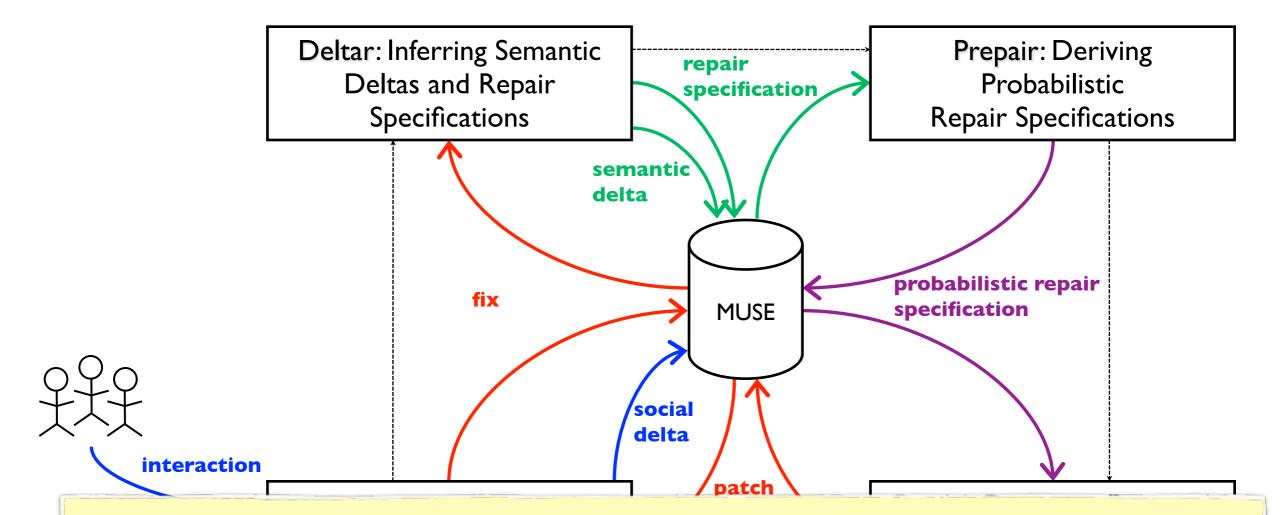
Remove View.setTag/getTag pattern to prevent crashes due to out of memory error as per Lint error:

"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."

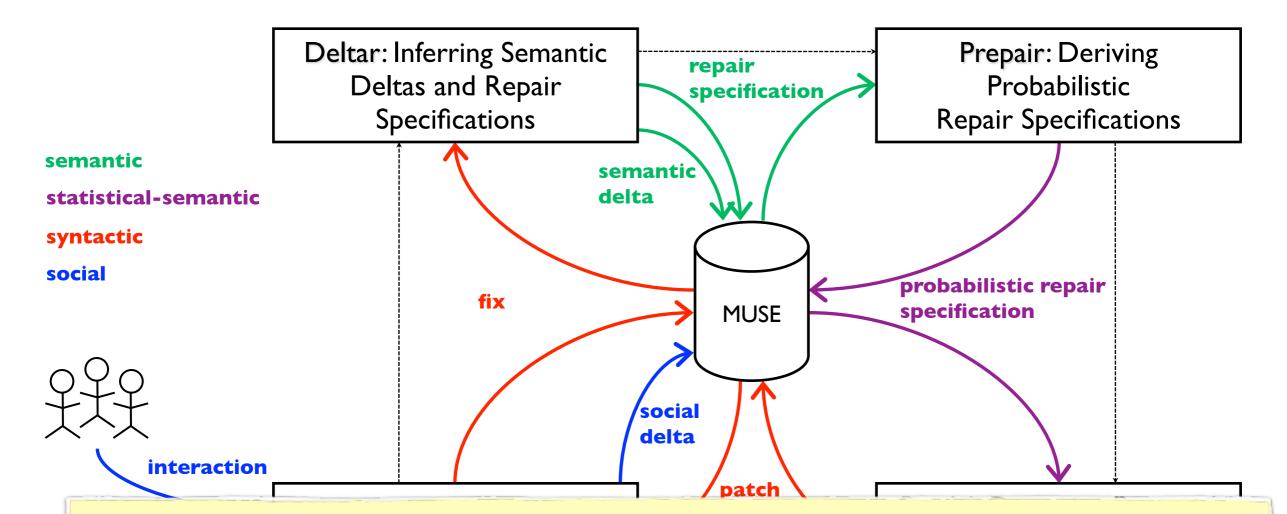
Approach: Mine meta-data artifacts



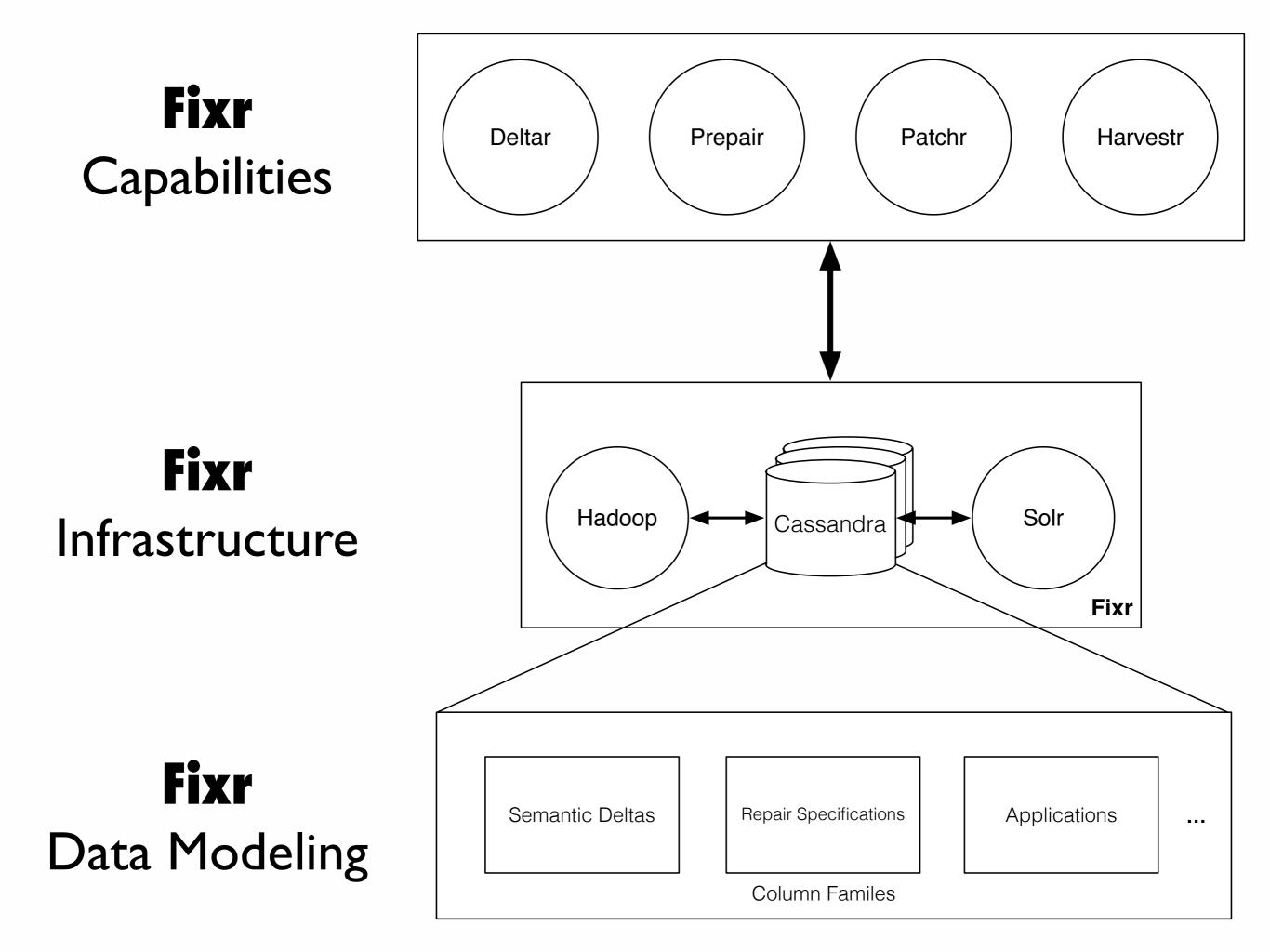


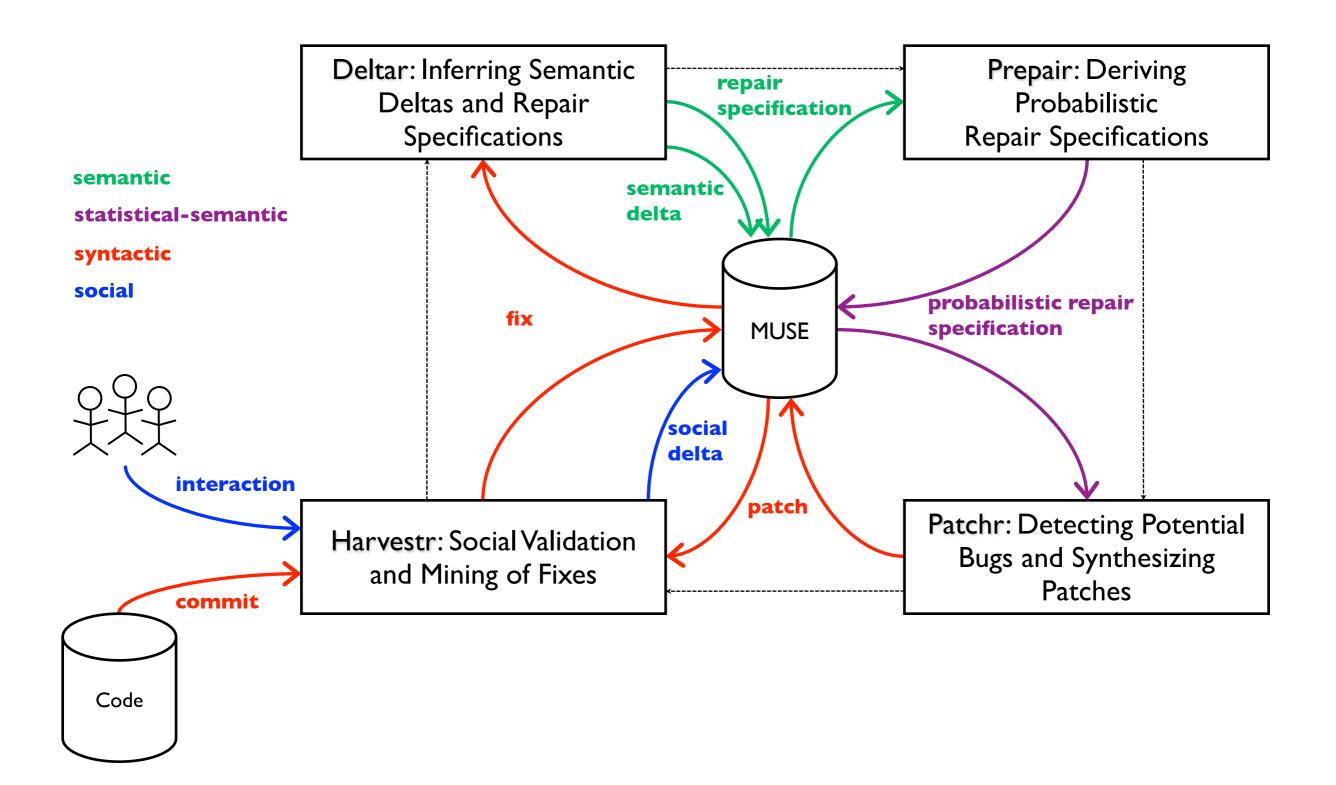


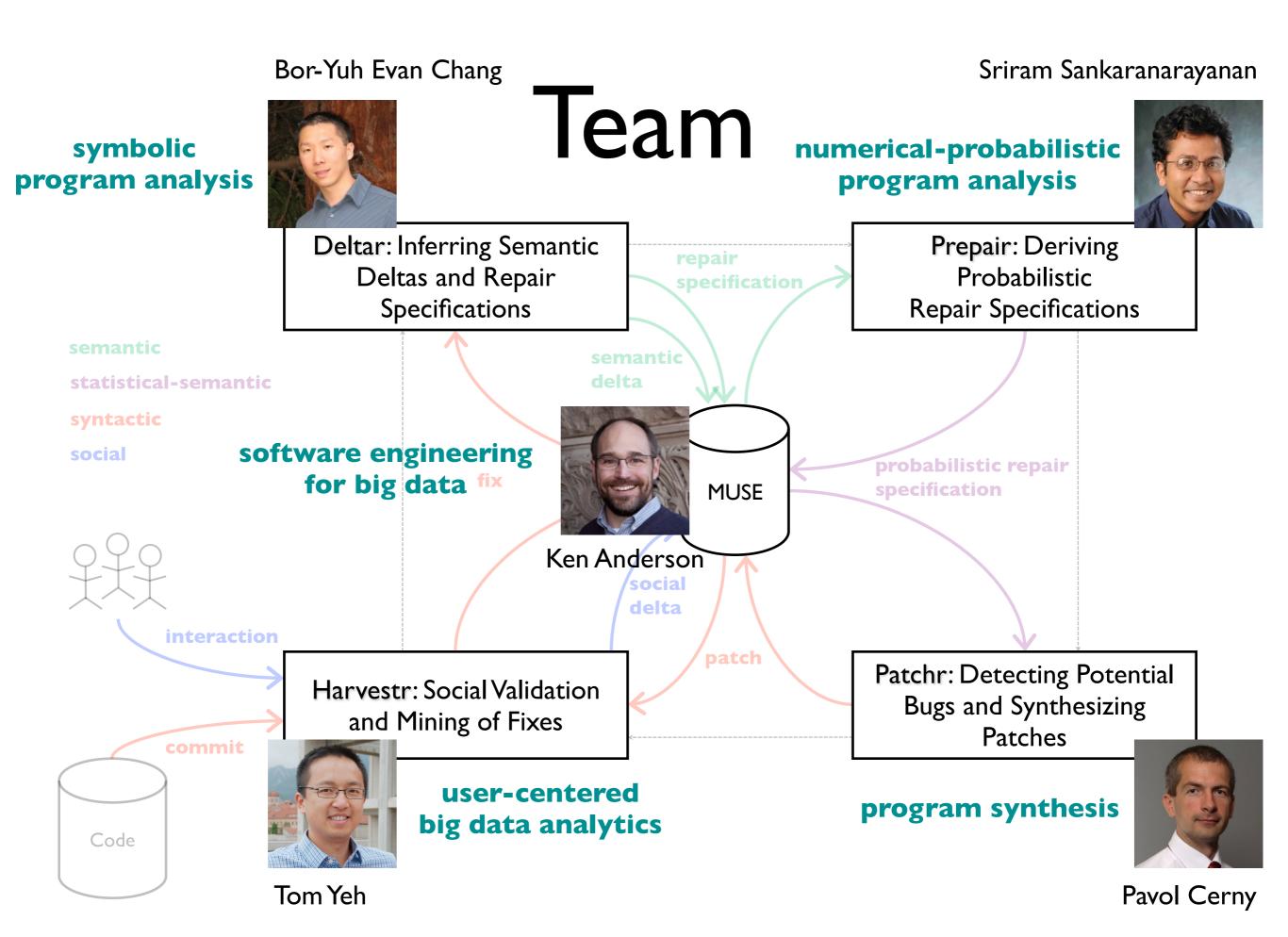
Goal: Create a positive feedback loop to derive high-confidence repair specifications



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



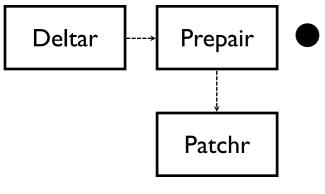




• Iterative and incremental design and evaluation of the **Fixr** loop

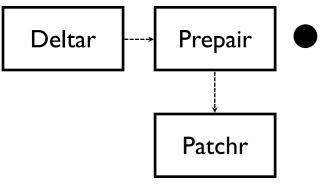
- Iterative and incremental design and evaluation of the **Fixr** loop
 - Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?

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



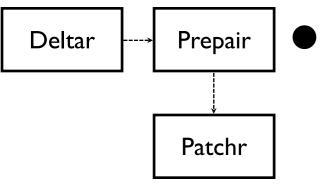
 Effectiveness of Bugfix Transfer: Given an isolated bugfix, can we derive high-quality repair specifications to lead to useful patches?

 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?

 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?

Deltar			
Harvestr	«	Patchr	

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

Prepair for **Fixr**

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



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.