Interactive Code Review for Systematic Changes

Tianyi Zhang,[1] Myoungkyu Song,[2] Joseph Pinedo,[2] Miryung Kim[1]
[1] University of California, Los Angeles [2] University of Texas at Austin

Code Review

What is code review?

- inspect changes
- find mistakes overlooked by developers

State-of-art

- Eclipse Compare, Gerrit, Phabricator, Code Flow
- line-level differences
- manual process

Motivation

- Reviewers have a hard time to inspect *systematic edits* — similar changes scattered across the program

```
int keyDownEvent (int w) {
-    ExpandItem item = items [index];
   switch (w) {
       case OS.SPACE:
          Event event = new Event ();
-            event.item = item;
-            sendEvent(true, event);
+            event.item = focusItem;
+            sendEvent(event);
+            refreshItem(focusItem);
```

(a) change example

```
int keyReleaseEvent (int wParam) {
-    ExpandItem item = items [index];
   switch (wParam) {
       case OS.SPACE:
          Event ev = new Event ();
-            ev.item = item;
-            sendEvent(true, ev);
+            ev.item = focusItem;
+            sendEvent(ev);
+            refreshItem(focusItem);
```

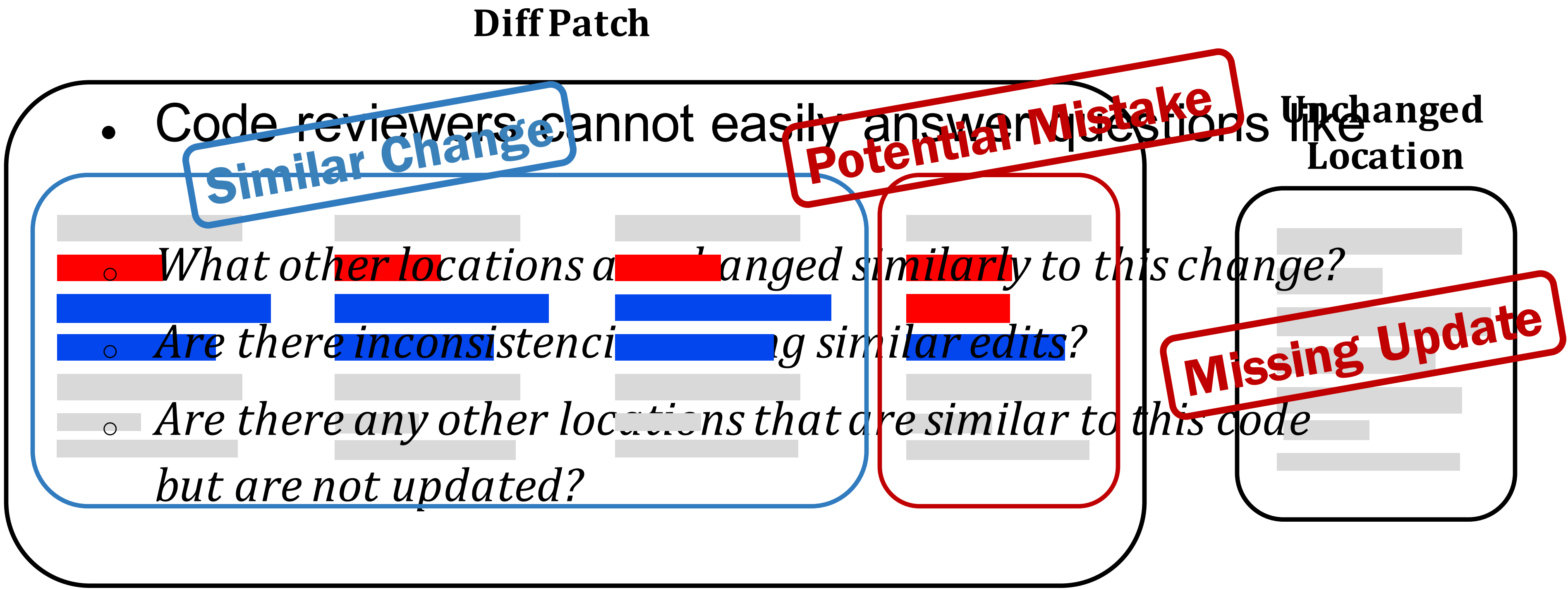
(b) a similar but not identical change

```
int ButtonUpEvent (int wParam) {
-    ExpandItem item = items [index];
   if (wParam == HOVER){
       Event bEvent = new Event ();
-            bEvent.item = item;
-            sendEvent(true, bEvent);
+            sendEvent(bEvent);
+            refreshItem(focusItem);
```

(c) an inconsistent change

Motivation

- Code reviewers cannot easily answer questions like
  - *What other locations are changed similarly to this change?*
  - *Are there inconsistencies among similar edits?*
  - *Are there any other locations that are similar to this code but are not updated?*



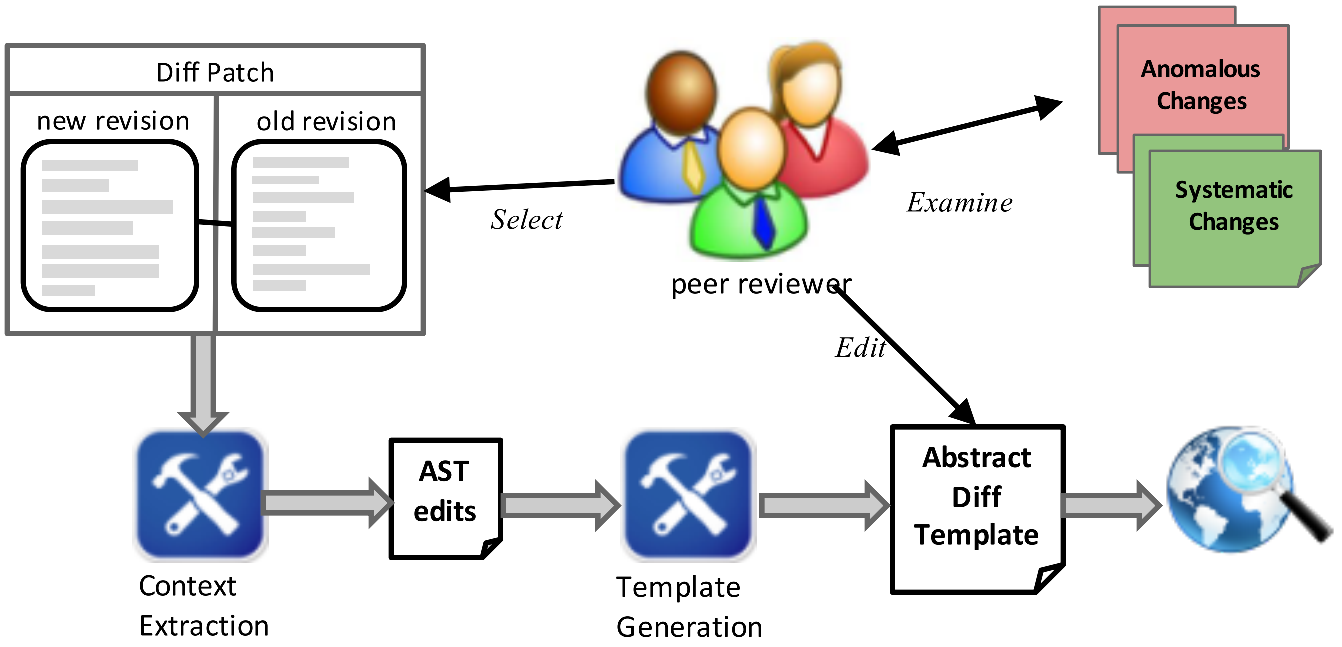
Outline

- Related Work
- Interactive Code Review Approach
    - Phase I: Context-Aware Change Template Generation
    - Phase II: Template Customization
    - Phase III: Change Summarization and Anomaly Detection
- Evaluation
    - Semi-Structured Interviews with Salesforce Engineers
    - A User Study with 12 ECE students at UT Austin
- Conclusion

Related Work

- Modern Code Review and Change Comprehension
  - Decompose large, composite changes into small ones [Rigby et al., Tao et al.]
  - Our work is inspired by these findings.
- Code Clone Analysis
  - Detect duplicated code and find cloning-related bugs [CCFinder, Deckard, CP-Miner, SecureSync]
  - But they are not designed to investigate diff patches.
- Systematic Change Automation
  - Automate similar changes to multiple locations [LASE, Sydit]
  - LASE uses fixed template generation and does not allow interactive customization.
  - LASE is not evaluated with user studies.

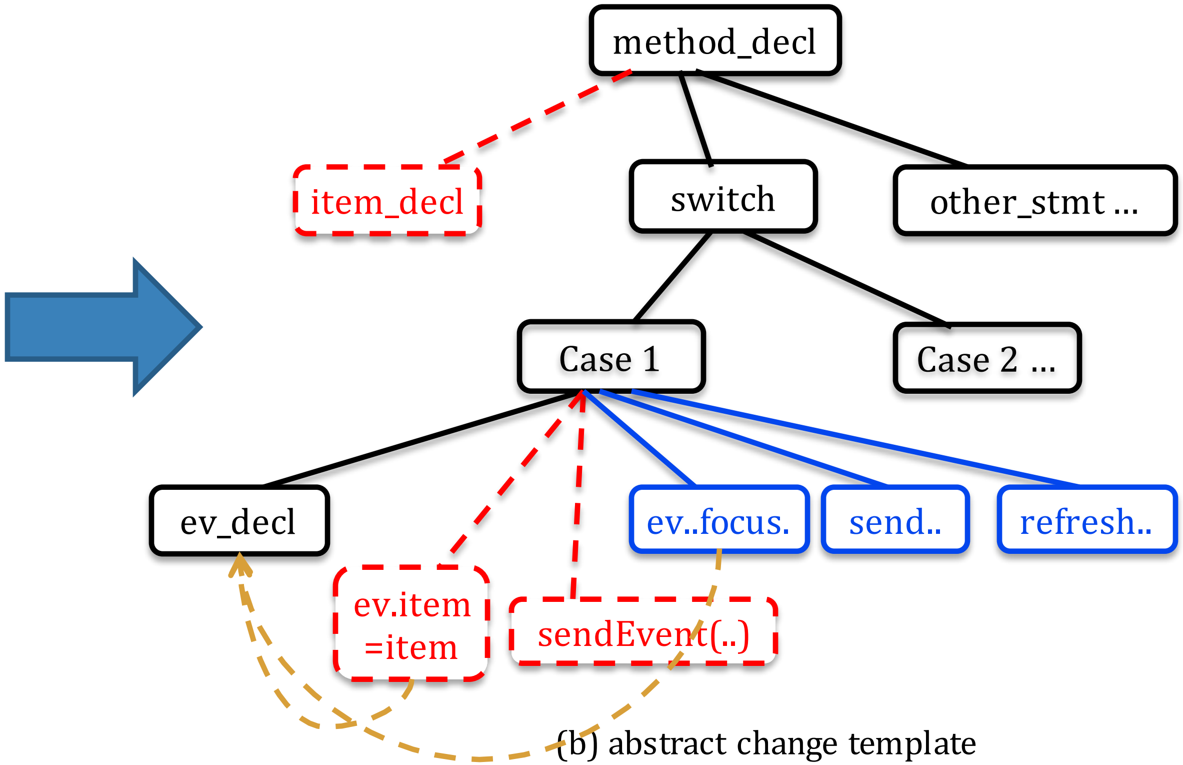
Critics: Interactive Code Review Approach for Systematic Changes



Phase I: Context-Aware Change Template Generation

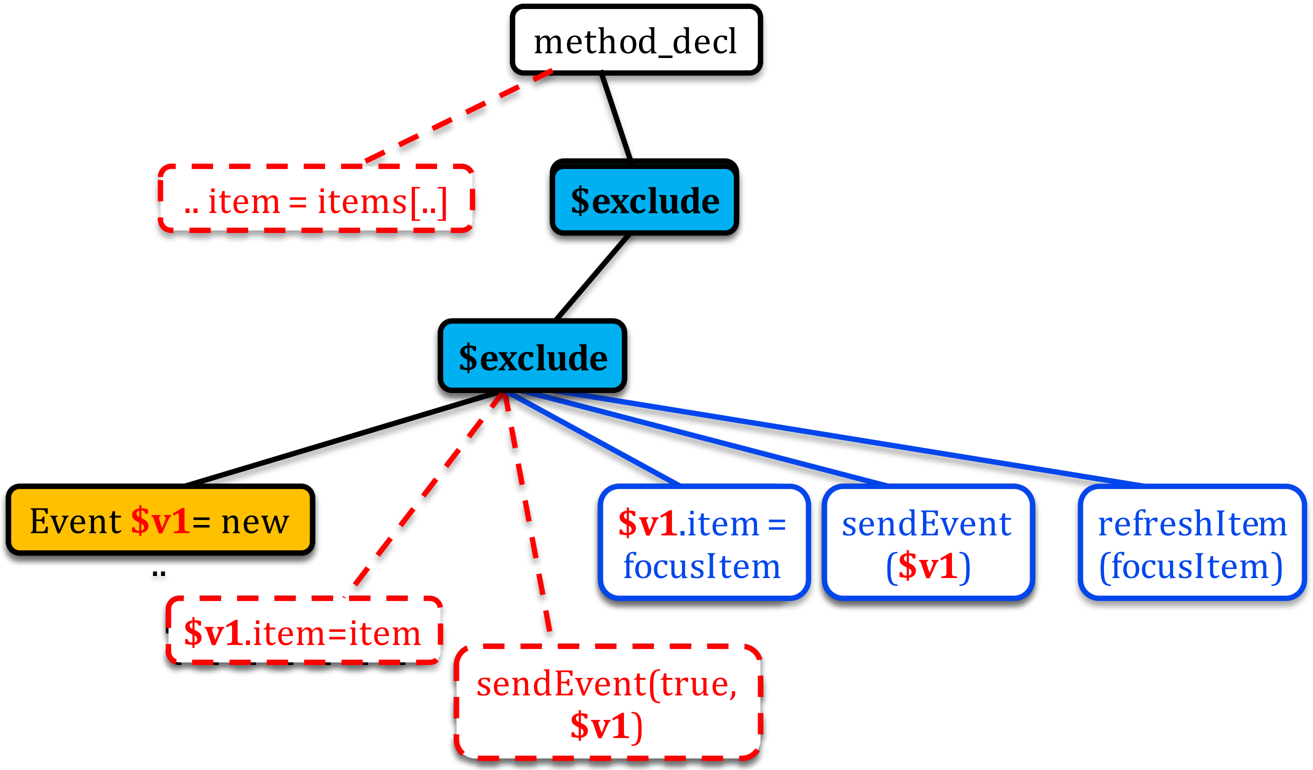
```
int keyDownEvent (int w) {
-   ExpandItem item = items [index];
    switch (w) {
        case OS.SPACE:
            Event ev = new Event ();
-           ev.item = item;
-           sendEvent(true, ev);
+           ev.item = focusItem;
+           sendEvent(ev);
+           refreshItem(focusItem);
```

(a) selected change



(b) abstract change template

Phase II: Template Customization



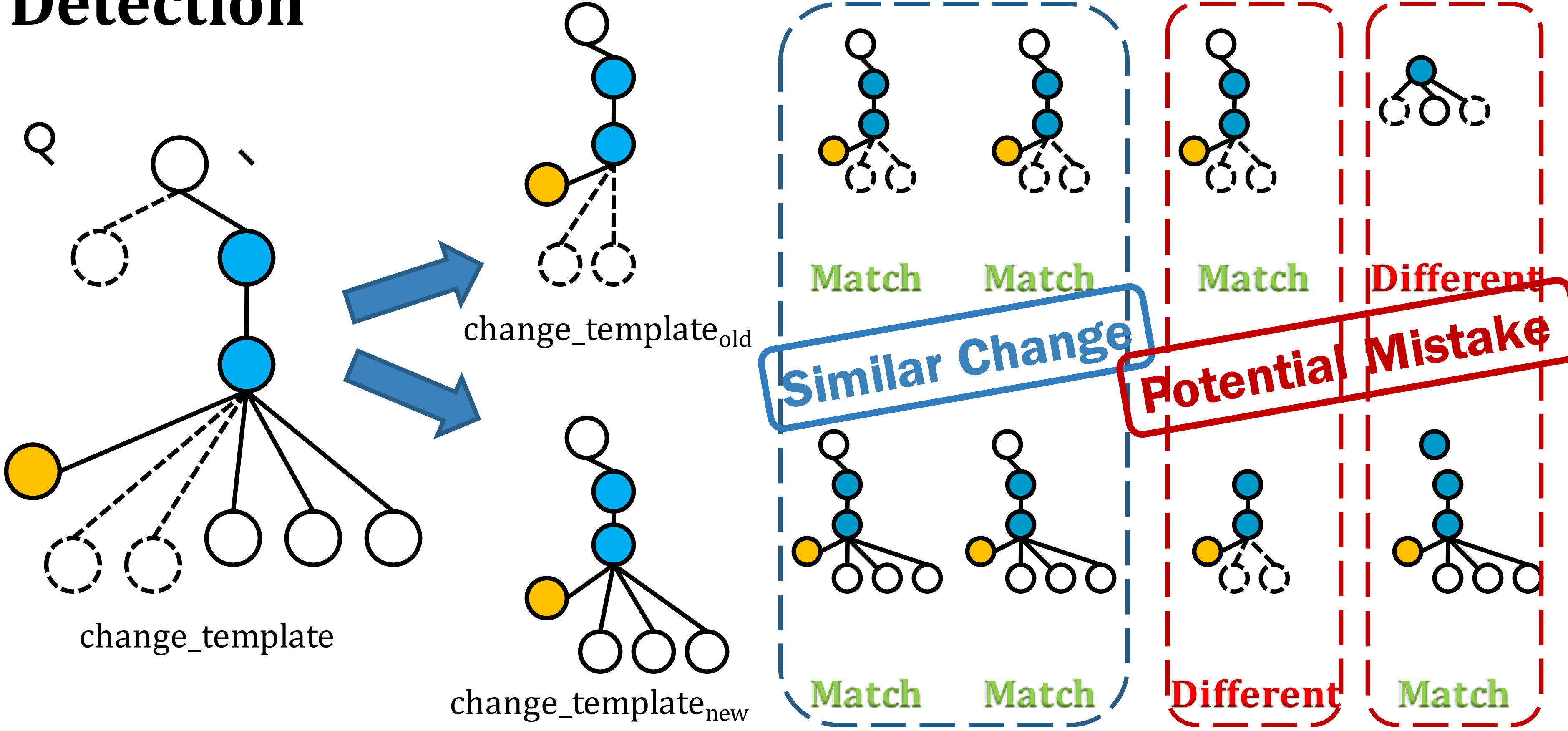
(c) customized change template

Phase III: Change Summarization and Anomaly Detection

- Critics searches for similar locations in the old revision and the new revision respectively.

	Match Old	Not Match Old
Match New	Correct similar change	**Similar change to different contexts**
Not Match New	**Missing similar change**	Irrelevant

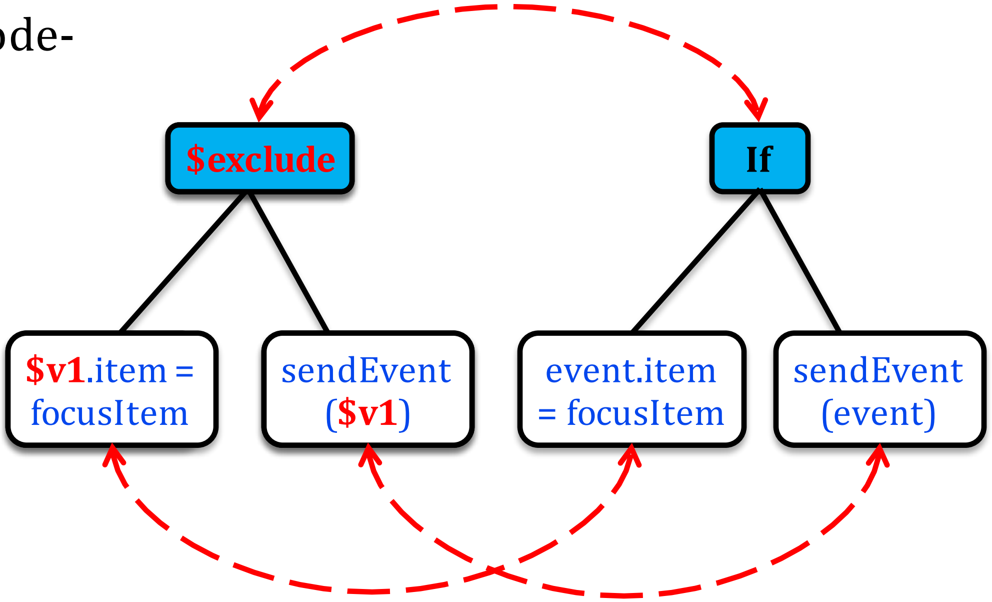
Phase III: Change Summarization and Anomaly Detection



Phase III: Change Summarization and Anomaly Detection

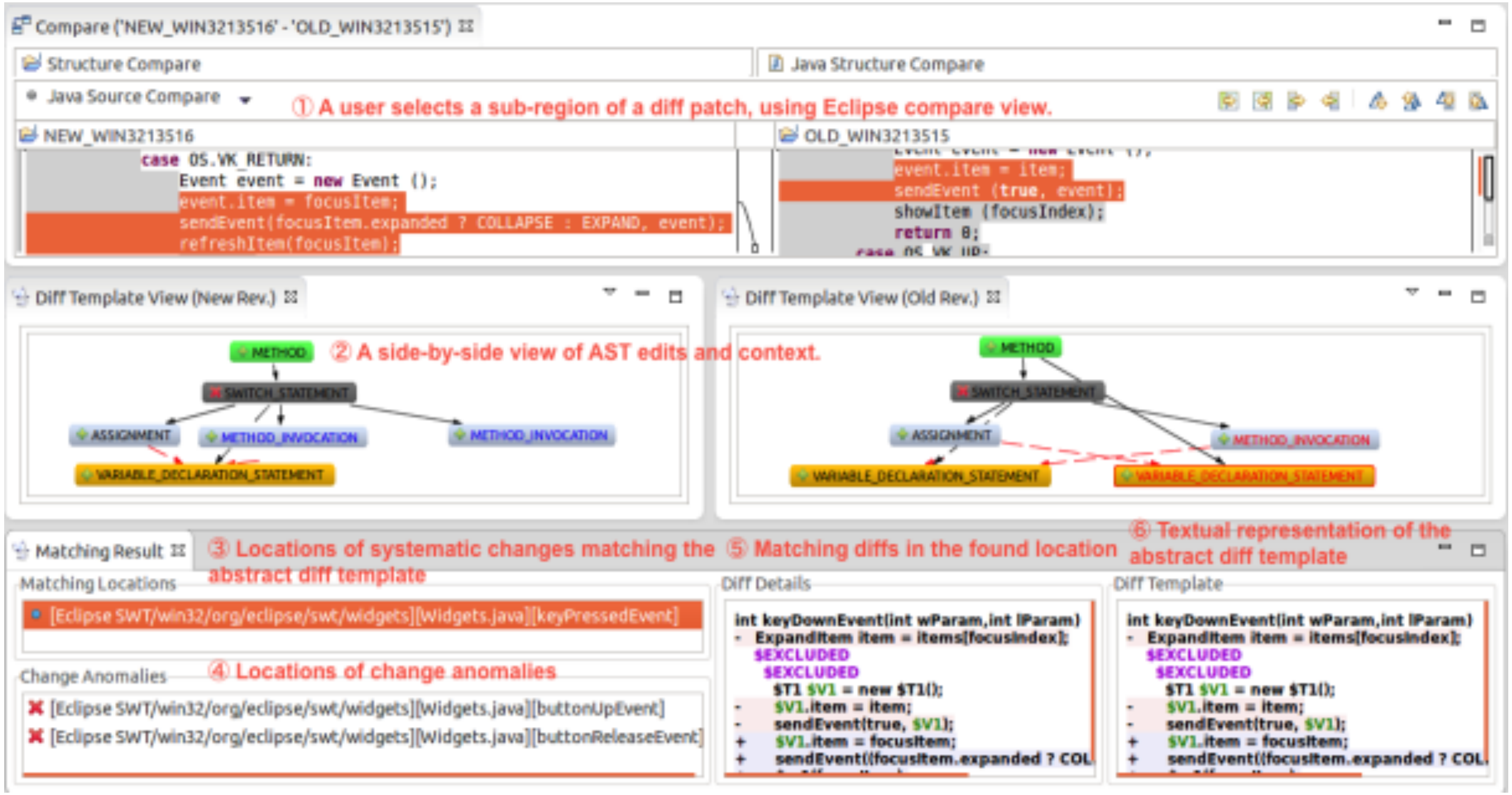
- Original RTED algorithm[1] computes node-level alignment between two trees.

- Critics extended RTED in two ways.
  - match a parameterized token with any concrete token.
  - match an excluded node with any node.



1. Pawlik, Mateusz, and Nikolaus Augsten. "RTED: a robust algorithm for the tree edit distance." *Proceedings of the VLDB Endowment* 5.4 (2011): 334-345.

Critics Plug-in


Compare ('NEW_WIN3213516' - 'OLD_WIN3213515')
Structure Compare
Java Structure Compare
Java Source Compare
① A user selects a sub-region of a diff patch, using Eclipse compare view.
NEW_WIN3213516
OLD_WIN3213515
Diff Template View (New Rev.)
Diff Template View (Old Rev.)
② A side-by-side view of AST edits and context.
METHOD
SWITCH_STATEMENT
ASSIGNMENT
METHOD_INVOCATION
METHOD_INVOCATION
VARIABLE_DECLARATION_STATEMENT
METHOD
SWITCH_STATEMENT
ASSIGNMENT
METHOD_INVOCATION
VARIABLE_DECLARATION_STATEMENT
VARIABLE_DECLARATION_STATEMENT
Matching Result
③ Locations of systematic changes matching the abstract diff template
⑤ Matching diffs in the found location
⑥ Textual representation of the abstract diff template
Matching Locations
Change Anomalies
④ Locations of change anomalies
Diff Details
Diff Template

NEW_WIN3213516:

```
case OS.VK_RETURN:
    Event event = new Event ();
    event.item = focusItem;
    sendEvent(focusItem.expanded ? COLLAPSE : EXPAND, event);
    refreshItem(focusItem);
```

OLD_WIN3213515:

```
    event.item = item;
    sendEvent (true, event);
    showItem (focusIndex);
    return 0;
case OS.VK_UP:
```

Matching Locations:

- [Eclipse SWT/win32/org/eclipse/swt/widgets][Widgets.java][keyPressedEvent]

Change Anomalies:

- ✗ [Eclipse SWT/win32/org/eclipse/swt/widgets][Widgets.java][buttonUpEvent]
- ✗ [Eclipse SWT/win32/org/eclipse/swt/widgets][Widgets.java][buttonReleaseEvent]

Diff Details:

```
int keyDownEvent(int wParam,int lParam)
-  ExpandItem item = items[focusIndex];
   $EXCLUDED
    $EXCLUDED
     $T1 $V1 = new $T1();
-    $V1.item = item;
-    sendEvent(true, $V1);
+    $V1.item = focusItem;
+    sendEvent((focusItem.expanded ? COL
```

Diff Template:

```
int keyDownEvent(int wParam,int lParam)
-  ExpandItem item = items[focusIndex];
   $EXCLUDED
    $EXCLUDED
     $T1 $V1 = new $T1();
-    $V1.item = item;
-    sendEvent(true, $V1);
+    $V1.item = focusItem;
+    sendEvent((focusItem.expanded ? COL
```

Eclipse plug-in are available at https://sites.google.com/a/utexas.edu/critics/. (Zhang et al. FSE 14’ Demo)

Research Questions

- RQ1: How critics could be used in practice?
- RQ2: How accurately does a reviewer locate similar edits and mistakes with Critics?
- RQ3: How much time can a reviewer save by using Critics?

Semi-Structured Interview at Salesforce

Subject	Role	Gender	Age	Java Experience	Code Review Frequency
1	Developer	Male	21-30	4	Weekly
2	QE	Female	21-30	3	Weekly
3	Manager	Male	41-50	4	Seldom
4	QE	Male	31-40	5	Weekly
5	QE	Female	31-40	10	Weekly
6	Developer	Male	41-50	14	Daily

Semi-Structured Interview at Salesforce

- 20-minute tutorial about how to use Critics
- A hands on trial of Critics[1] with one of four diff patches authored by their own team.

No.	Patch Description	Changed LOC	Num of Changed Files
1	Refactor test cases by moving bean maps to respective utils classes	743	22
2	Refactoring the API to get versioned field values	943	34
3	Refactor test cases to use try-with-resources statements	484	10
4	Update common search tests by getting versioned test data	2224	12

1. Critics is implemented as an Eclipse plug-in, http://sites.google.com/utexas/edu/critics/

How could Critics help them with code reviews?

- *"... REST APIs across different versions generally share similar code snippets ... It's hard and time-consuming to find mistakes on similar changes on those locations..."*

- *"The feature in your tool can free us from piling code review tasks on our senior developers..."*

How do they like or dislike Critics?

- *“Currently COLLABORATOR only highlights the changed location in a very naive way. A feature like extracting and visualizing the change context can help us better understand the change itself as well as find some underlying change patterns between related changes.”*

- *“It will be helpful if Critics can provide some hints about template customization.”*

User Study at UT Austin

- We recruited 12 UT Austin students
  - 4 of them are ECE undergrads, the others are graduate students in Software Engineering
  - All of them have at least one year experience of Eclipse IDE
  - All but one have code review experience using diff tools such as Eclipse Compare and SVN/Git diff.
- We gave them a 20-minute tutorial on how to use Critics plug-in

Code Review Patches

	Version	Change Description	Similar Change	Inconsistent Change	Missing Update	Size(LOC)
Patch 1	JDT 9800 vs JDT 9801	Initiate a variable in a for loop instead of using a hashmap	getTrailingComments(ASTNode) getLeadingComments(ASTNode) getExtendedEnd(ASTNode)	getExtendedStartPosition(ASTNode)	getComments(ASTNode) getCommentsRange(ASTNode)	190
Patch 2	JDT 16010 Vs JDT 10611	extract the logic of unicode traitement to a method	getNextChar() getNextCharasDigit() getNextToken() … 9 locations in total	getNextCharAsJavaIdentifierPart(ASTNode)	jumpOverMethodBody() … 11 locations in total	680

User Study Tasks

- Each participant carried out code review tasks on two different patches, one with Critics and the other with Eclipse Compare
  - Q1： Given the change in the method *getTrailingComments*, what other methods containing **similar changes** can you find?
  - Q2：Which of the following methods contains **inconsistent changes** compared with the change in *getTrailingMethods*?
  - Q3： How many methods share context similar to the change in *getTrailingMethods* but have **missed updates**?
- We measured task completion time and accuracy.

Subjects	Critics				Eclipse Compare			
	Q1	Q2	Q3	Time	Q1	Q2	Q3	Time
1	√	√	√	13:30	N/A	N/A	N/A	26:37
2	√	√	√	13:18	✗	√	✗	47:21
3	√	√	√	18:29	√	√	✗	24:54
8	✗	√	√	20:00	✗	✗	✗	18:14
9	√	√	√	29:00	✗	✗	√	15:00
10	√	√	√	16:11	√	√	✗	37:57
11	√	√	√	14:27	√	√	✗	25:45
12	√	√	√	35:17	✗	√	✗	22:46
Average	83%	100%	92%	19:26	42%	58%	33%	25:39

Human subjects can answer questions about systematic changes 47.3% more correctly with 31.9% saving in time using Critics

Comparison with LASE

- LASE automates systematic editing by searching for locations and applying edits to individual locations. [Meng et al.]
- It is challenging to directly compare LASE and Critics:
    - **fixed vs. interactive** template generation

- Simulate observed template customization patterns
- Compare the locations found by the two techniques

Comparison with LASE

- In five out of six cases, Critics achieves the same or higher accuracy than LASE within a few iterations.

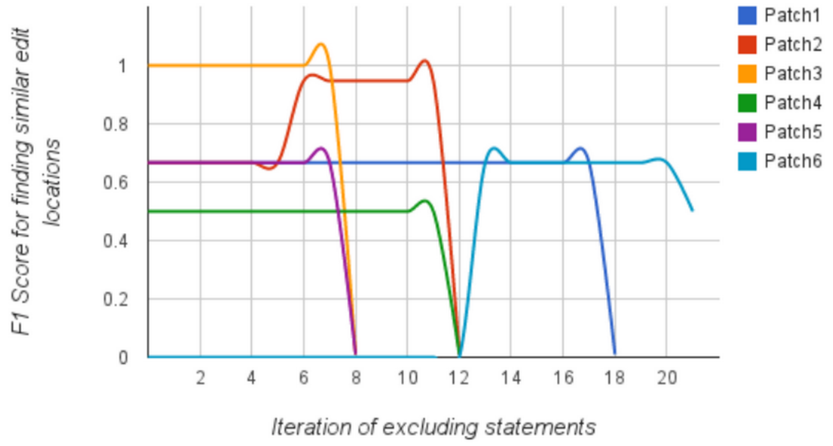
	Critics				LASE	
	Precision	Recall	Iterations	Time(sec)	Precision	Recall
Patch 1	1	1	4	1.66	1	1
Patch 2	1	0.9	6	8.95	0.92	0.75
Patch 3	1	1	0	13.52	1	1
Patch 4	1	1	7	71.98	1	0.33
Patch 5	1	1	4	6.86	1	1
Patch 6	1	0.33	3	1.47	1	1
Average	1	0.87	4	17.41	0.99	0.84

Conclusion

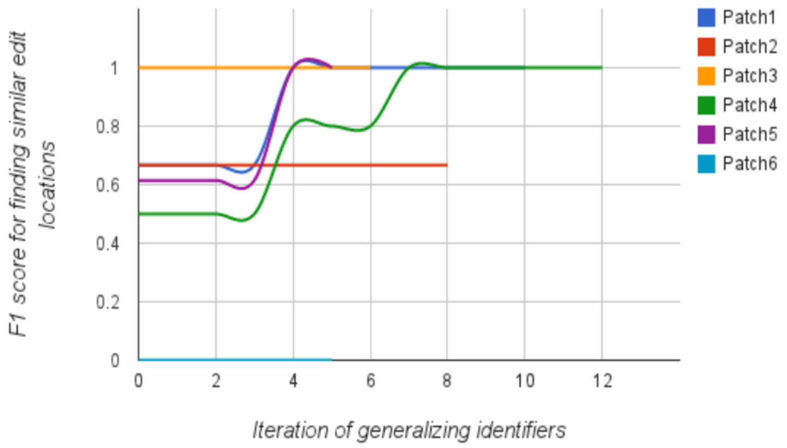
- We present Critics, a novel interactive code review approach for searching systematic changes and detecting potential mistakes.
- A study at Salesforces show that Critics scales to an industry-scale project and can be easily adopted by professional developers
- Human subjects using Critics can answer questions about systematic changes more correctly within less time, in the comparison of the baseline use of Eclipse Compare.

Q&A

Accuracy variation in Critics's Simulation



(a) F1 score for finding similar edit locations by excluding statements.



(b) F1 score for finding similar edit locations by parameterizing identifiers.

Subjects and Metrics

- Six patches drawn from Eclipse JDT and SWT [Meng et al.]
  - Patch size ranges from 190 to 680 lines of changed code
  - Consisted of three to ten systematic edits
- Metrics
  - precision
  - recall
  - F_1 score