An Empirical Study of Multi-Entity Changes in Real Bug Fixes

By Ye Wang, Na Meng and Hao Zhong

Bug fixing and automated tools

- Similar bugs occurs again and again
- Similar bug fixing pattern can be repeatedly used.
- Researchers proposed various tools to generate bug fixes or suggest customized edits

Automatic program repair (APR)

- Automatic program executes a buggy program P with a test suite T, and leverages bug localization techniques to locate a buggy method.
- APR then creates candidate patches to fix the bug, and validates patches via compilation and testing until obtaining a patched program that passes T.
- Different APR approaches generate patches either by randomly mutating code, creating edits from the recurring change patterns of past fixes, or solving the constraints revealed by passed and failed tests .
- However, each fix suggested by current APR approaches only modifies a single method.

Single or Multiple?

- The fixes that these tools focus on are limited to code changes within single methods or edits solving single software faults
- The majority of real fixes solve multiple software faults together

Problem

- Is there any repeated bug-fixing pattern that repetitively applies similar sets of relevant edits to multiple program entities?

Approaches

- Study on 2,854 bug fixes from 4 projects
  - Aries, Cassandra, Derby, Mahout
- Tool: InterPart
  - Doing static analysis to identify the syntactic dependency relationships
- Change Dependency Graphs

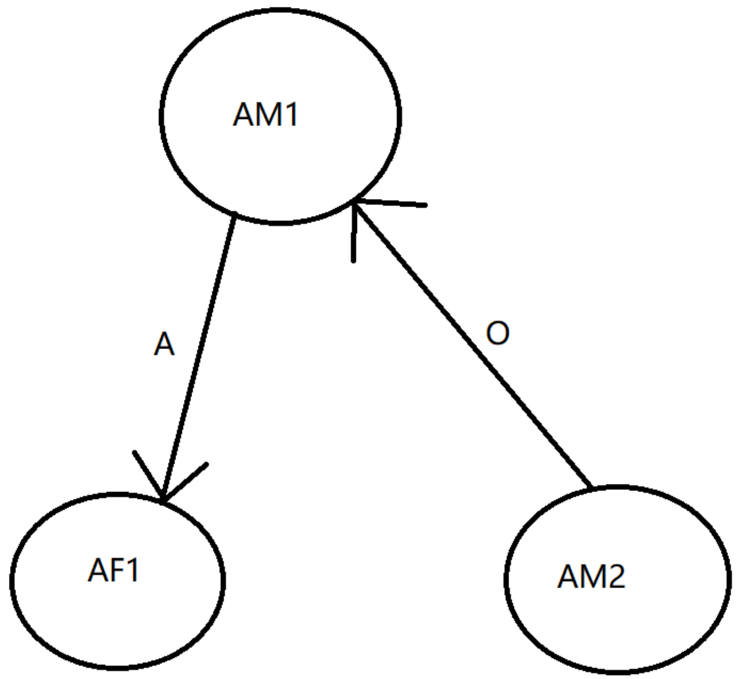
Change Dependency Graphs

- Vertices:Changed Program Entity/Atomic Changes
  - A(Added),/D(Deleted)/C(Changed)
  - +
  - C(Class)/M(Method)/F(Field)
  - Except CF
- Edges: Syntactic dependency relationship
  - Containing
  - Overridding
  - Accessing

Definition of CDG

- CDG $=< V, E >$, where V is a set of vertices representing changed entities, and E is a set of directed edges between the vertices $E \subseteq \{V \times V\}$. There is a directed edge from changed entity u to changed entity v, if and only if u is syntactically dependent on v.

A simple CFG



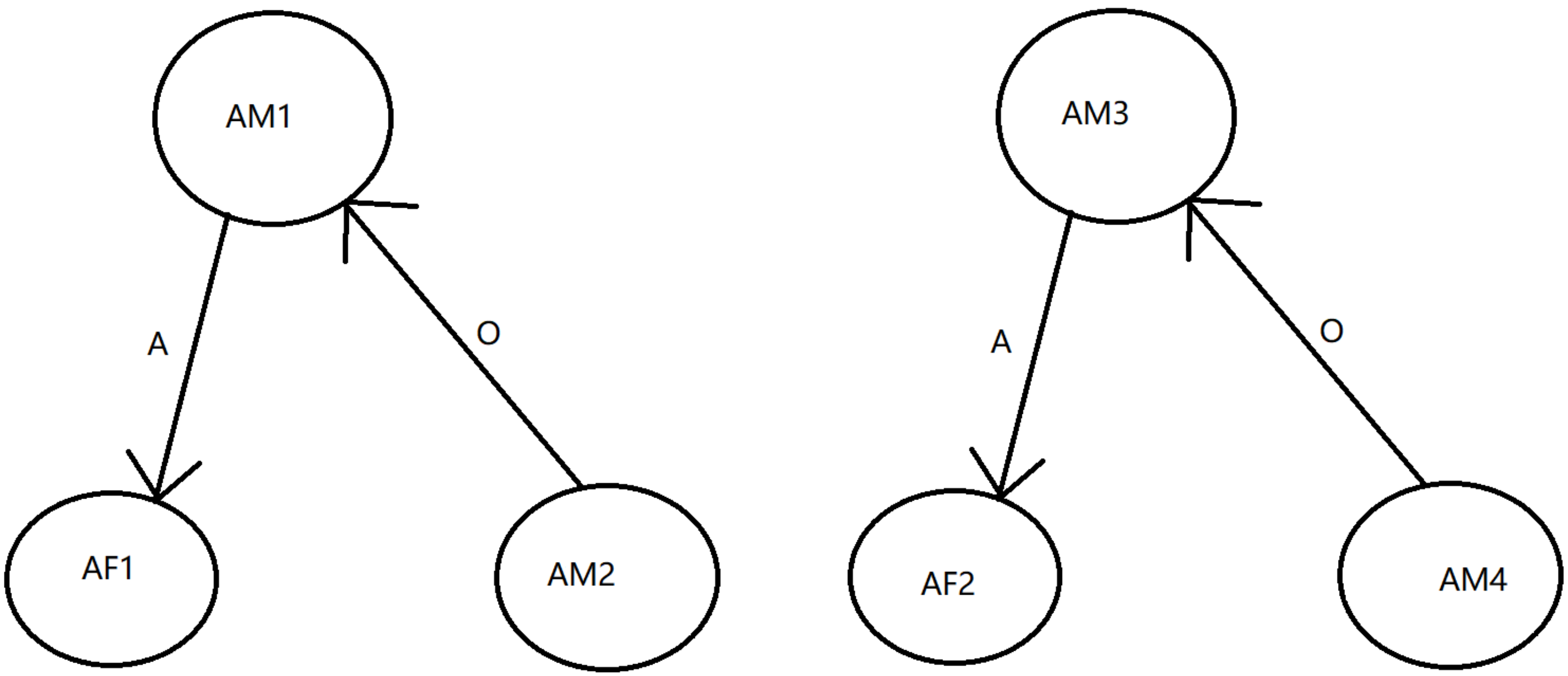
Change Pattern

- Given CDG $=< V, E >$, cp $=< V 0 , E0 >$, where $V 0 \subseteq V$ and $E0 \subseteq E$. A cp should contain at least two nodes and one edge connecting the nodes.

Recurring Change Pattern

- Suppose that the CDGs of code revisions r1 and r2 are $GS1 = \{cdg11, \ldots, cdg1m\}$ and $GS2 = \{cdg21, \ldots, cdg2n\}$. If a change pattern cp occurs in both cdg1i and cdg2j ($i \in [1, m]$ and $j \in [1, n]$), we say that the change pattern is also an rcp.

A simple RCP



InterPart

- Implementation strategy : Incomplete Static Analysis

- Construct CDG
  - Extracting Changed Entities
  - Correlating Changed Entities

- Extracting Recurring Change Pattern

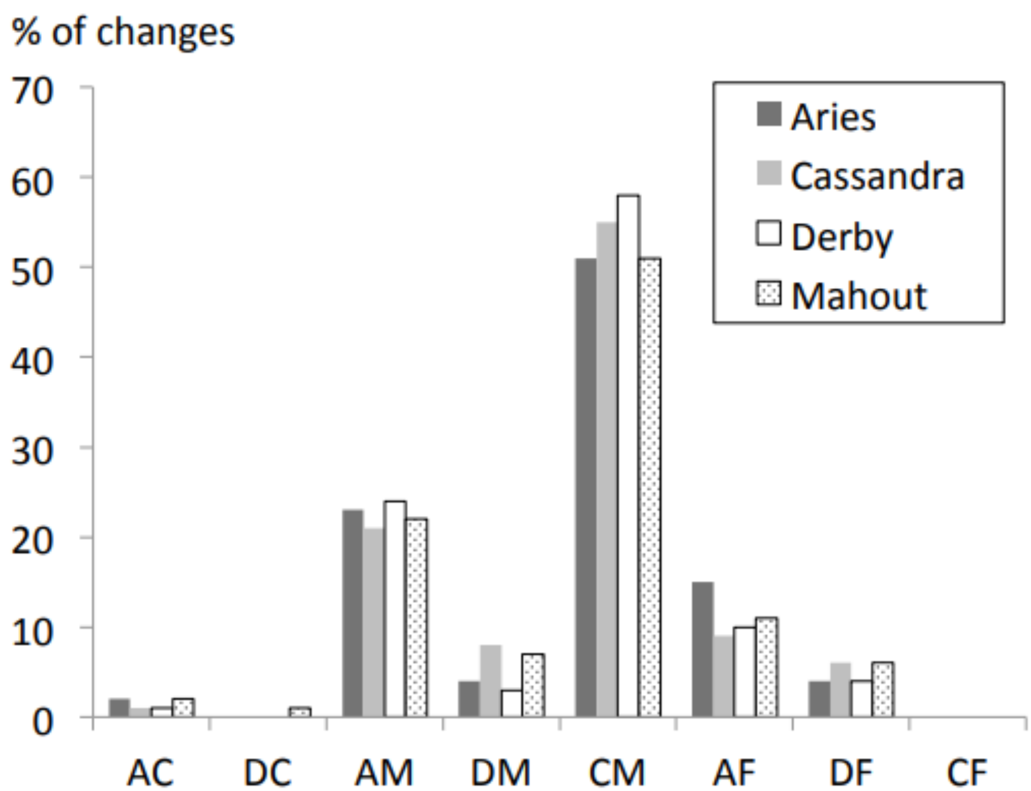
Research Question

- RQ1: What is the frequency of multi-entity bug fixes?
- RQ2: What patterns are contained by multi-entity fixes?
- RQ3: Why do programmers make multiple-entity changes, when they fix real bugs?

The datasets

- Aries, Cassandra, Derby, and Mahout
  - From different application domains
  - Well-maintained issue tracking systems and version control systems
  - Many bug-fixing commits refer to the corresponding bug reports via issue IDs

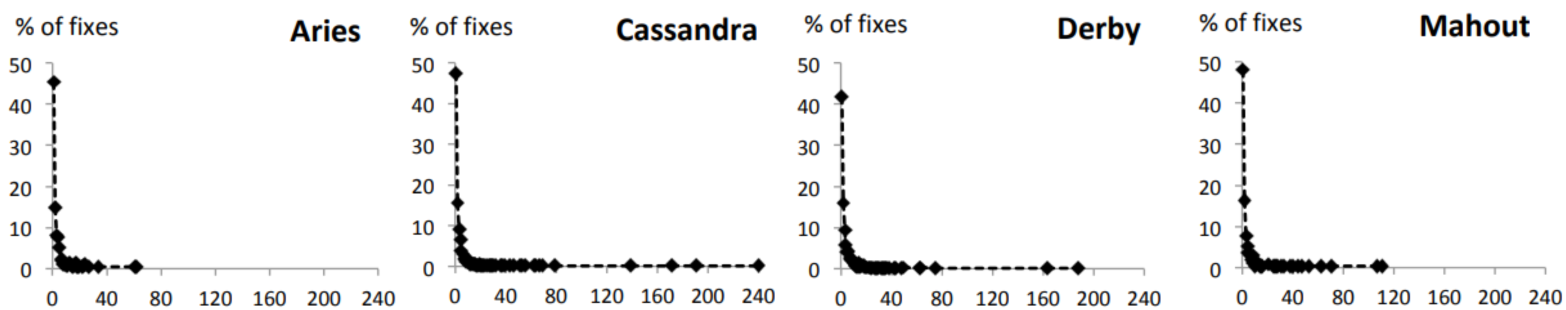
What is the frequency of multi-entity bug fixes?



What is the frequency of multi-entity bug fixes?

- Similar to the fixes generated by APR approaches, real bug fixes also mainly consist of CMs. However, real fixes usually involve a much more diverse set of entities and change types, such as AMs and AFs.

What is the frequency of multi-entity bug fixes?



What is the frequency of multi-entity bug fixes?

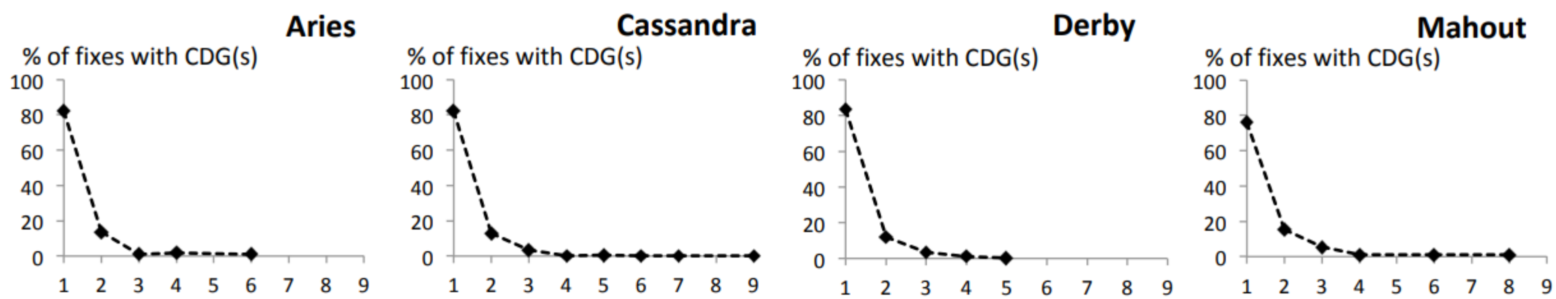


Fig. 5: The distribution of fixes with CDG(s) based on the number of extracted CDGs

What is the frequency of multi-entity bug fixes?

- Differing from the fixes generated by APR approaches, over half of the real fixes mainly involve multientity instead of single-method changes.

What is the frequency of multi-entity bug fixes?

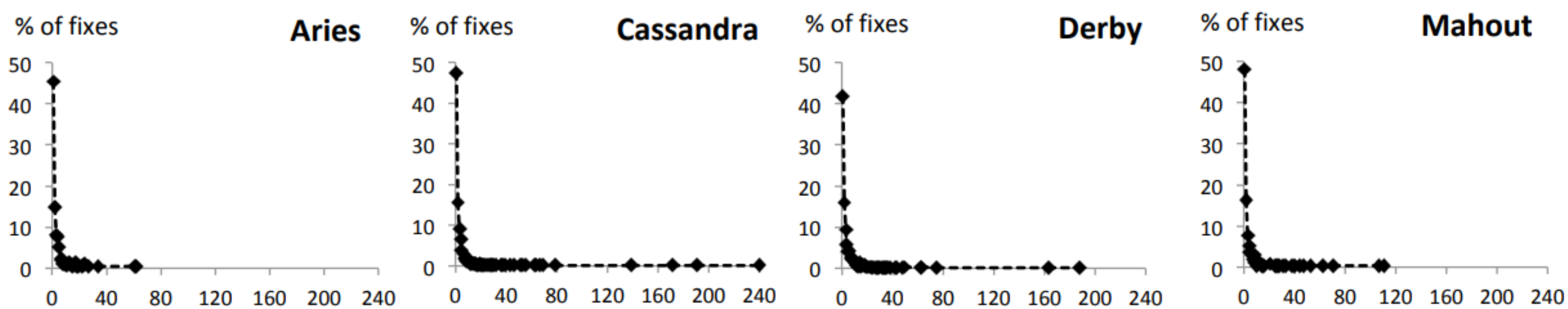


Fig. 4: Bug fix distribution based on the number of included changed entities

What is the frequency of multi-entity bug fixes?

TABLE II: Bug fixes with multi-entity changes

	Aries	Cassandra	Derby	Mahout
# of Fixes with Multi-Entity Changes	135	794	479	139
# of Fixes with CDG(s) Extracted	102	588	357	92
% of Fixes with CDG(s) Extracted	76%	74%	75%	66%

What is the frequency of multi-entity bug fixes?

- Among the fixes with multi-entity changes, 66- 76% of the fixes contain related changed entities, and 76- 83% of such fixes have entities connected in one or more CDGs. This indicates that comparison/recommendation tools that relate co-applied changes will be valuable

What patterns are contained by multi-entity fixes?

TABLE III: Recurring change patterns and their matches

	Aries	Cassandra	Derby	Mahout
# of Patterns	26	125	87	24
# of Fixes Matching the Patterns	97	585	352	87
# of Subgraphs Matching the Patterns	267	1,883	1,270	239

What patterns are contained by multi-entity fixes?

- The fix patterns of multi-entity changes commonly exist in all the investigated projects. This indicates that such patterns may be usable to guide APR approaches and to generate patches changing multiple entities.

What patterns are contained by multi-entity fixes?

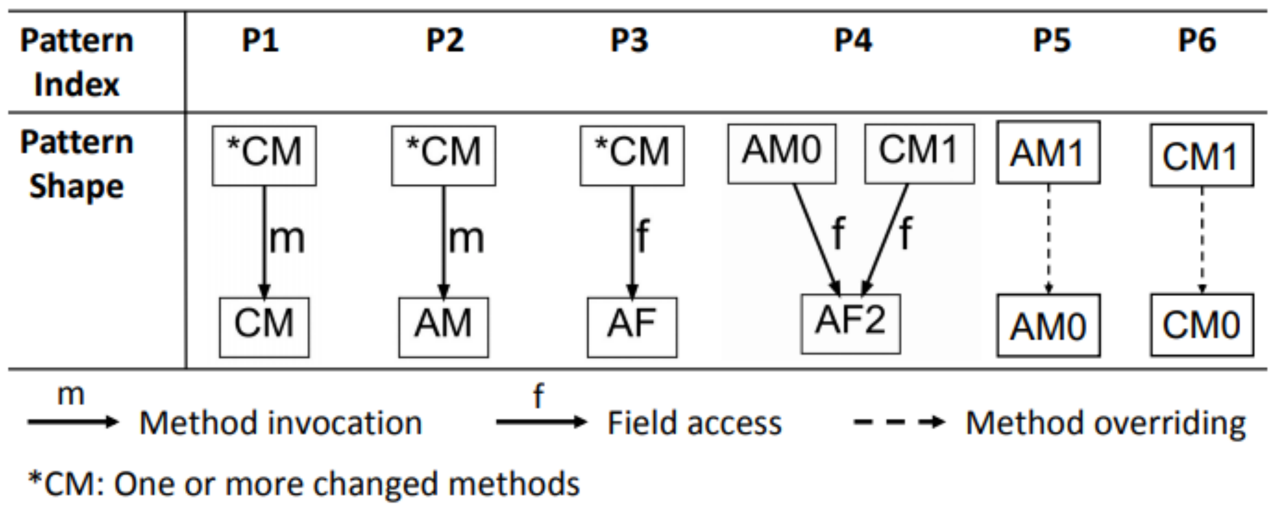


Fig. 6: Six recurring change patterns existing in all projects.

What patterns are contained by multi-entity fixes?

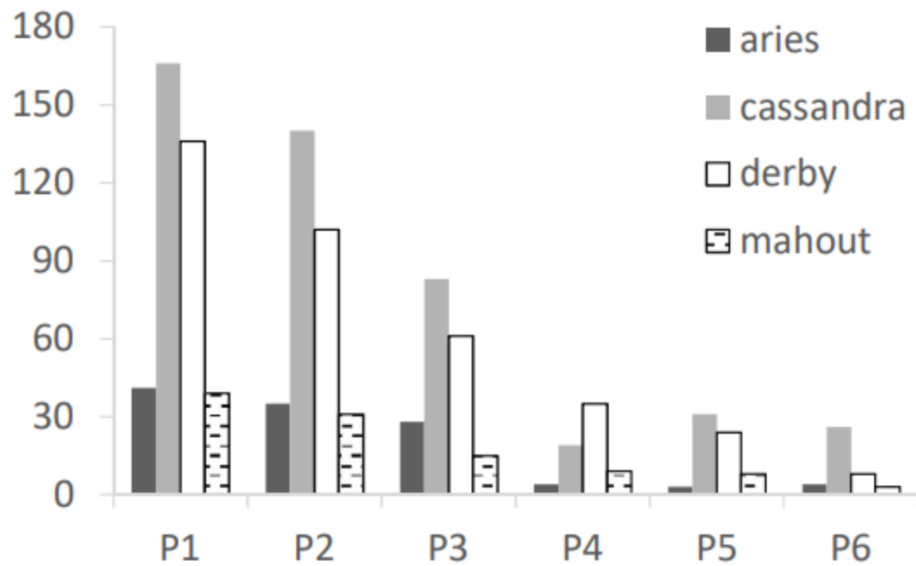


Fig. 7: Bug fixes matching the six frequent patterns

What patterns are contained by multi-entity fixes?

- Four out of the six most frequent fix patterns apply multiple CM changes. It indicates that existing APR approaches can be extensible to generate multi-entity fixes by modifying several methods that call the same changed method or access the same added field.

Why do programmers make multiple-entity changes, when they fix real bugs?

- Two case studies
  - 1.Examining 291 fixes that contained any of P1-P3, and explored why and how the multi-entity changes were applied
  - 2. Extracted the entity pairs that were repetitively co-changed in version history, and inspected 20 of such pairs to investigate any characteristics.

Case Study 1

TABLE IV: The scenarios where the most frequent three recurring change patterns occur

Pattern Index	Scenario	% of examined fixes
P1	1. Consistent changes between the callee and callers	23%
	2. Signature change of the callee method	17%
	3. Add, delete, or update the caller-callee relationship	17%
	4. Not closely related changes	43%
P2	1. Add a method for refactoring	28%
	2. Add extra logic or data processing	72%
P3	1. Add a field for refactoring	9%
	2. Partially replace an existing entity	25%
	3. Add extra logic or data processing	66%

Case Study 1

- Scenarios for P1 (*CM→CM(invocation))
  - 23% : developers applied consistent changes to these methods.
  - 17% : developers changed caller methods
  - 17% : developers changed the implementation logic of callees
  - 43% :no obvious relationship between the edits in co-changed methods

Case Study 1

- Scenarios for P2 (*CM→AM(invocation))
  - 28% : developers added the new method for refactoring purposes
  - 72% : developers added a method to implement new logic, and changed current methods to invoke the added method

Case Study 1

- Scenarios for P3 (*CM→AF(access)).
  - 9% : developers added a field for refactoring
  - 25% : developers applied changes to enhance existing features
  - 66% : developers applied changes to add new features

Case Study 1

- Among P1-P3, we did not see any identical fixes. It means that APR approaches are unlikely to independently suggest a correct multi-entity fix purely based on past fixes, although it is still feasible for new tools to help complete developers' fixes.

Case Study 2

TABLE V: The repetitively co-changed entity pairs

Co-Changed Entities	Aries	Cassandra	Derby	Mahout
Two fields	3	6	0	0
Two methods	16	203	93	22
One class and one field	0	1	0	0

Case Study 2

TABLE VI: Characteristics of repetitively co-changed pairs

Characteristics	# of Pairs	Similar statement change?
Similar statements	7	✓
Relevant usage of fields	5	✗
Commonly invoked methods	4	✓
Unknown	4	✗

Case Study 2

- The repetitively co-changed entities usually share common characteristics like similar content, relevant field usage, or identical method invocations, among which the similar usage of fields or methods has not been leveraged to automatically complete developers’ fixes.

Related Works

- Among P1-P3, we did not see any identical fixes. It means that APR approaches are unlikely to independently suggest a correct multi-entity fix purely based on past fixes, although it is still feasible for new tools to help complete developers' fixes.

Related Works

- Empirical Studies on Code Changes
- What's new:
  - Examining reasons to explain the co-changed related entities
  - Same, but more accurate

Related Works

- Change Impact Analysis
- What's new:
  - Exploring the recurring patterns of co-applied atomic changes
  - Static analysis on partial code

Related Works

- Automatic Program Repair (APR)
- What's new:
  - Showing the significant gap between APR fixes and real fixesStatic analysis on partial code
  - Potential ways to close the gap

Threats to Validity

- External Validity : may not generalize to other projects
- Construct Validity : may be subject to human bias
- Internal Validity : InterPart may miss some dependency relations between co-applied changes

Conclusion

- Multi-entity fixes are frequently applied by developers
- There are three major recurring patterns that frequently connect relevant co-changed entities
- Although a multi-entity fix is never identical to other fixes, the fix may apply similar or divergent edits to the entities with similar textual content, field usage, or method invocations.

Discussion

- How can developers help the automated tools to fix bugs?
- For multi-entity bug fixing, which is the best way to fix bugs?

Thanks!