Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets

Background

- Code review is important for software quality assurance
- Understanding changes is difficult, when the changeset consists of multiple, independent, code differences
- There is no tool that automatically decomposes composite changes

Contributions

- The design and implementation of ClusterChanges, a lightweight static analysis technique for decomposing changesets
- A user study to validate the results of ClusterChanges, to understand differences between different types of partitions, and to gauge the tool's potential usefulness

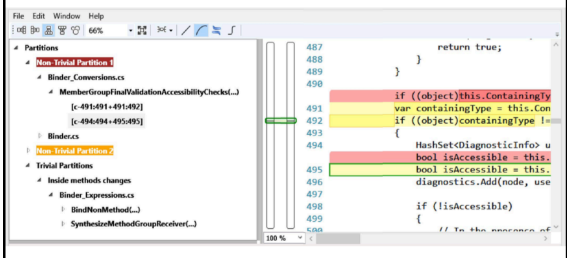
ClusterChanges

- Leverages Roslyn, a Microsoft compiler that provides open APIs, to create an AST for changed files with the best effort
- Uses the def-use relationship to cluster diff-regions based on the edited code in after-files
  - E.g., if a type/field/or method is referenced by a method, the two diff regions are connected

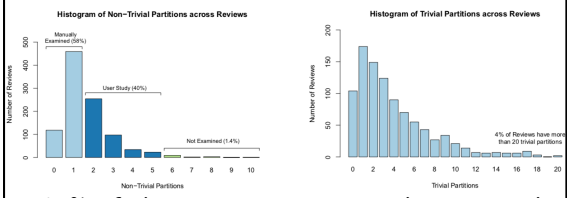
ClusterChanges (cont'd)

- Group diff-regions in the same method together
- Trivial vs. Nontrivial partitions
  - Trivial partitions are one or more diff-regions within the same method, or single diff-regions outside a method
  - Nontrivial partitions contain diff-regions from multiple methods or changed entities

Tree view displaying a changeset



Distribution of non-trivial partitions and trivial partitions



- 45% of changesets contain single nontrivial regions.
- There is a long tail in trivial in-method partitions

7

Research Questions and Interview Questions

- RQ1: Do developers agree with the change decomposition by ClusterChanges?
  - Is the decomposition intuitive?
  - Is the decomposition correct?
- RQ2: What role do trivial partitions play?
  - Are nontrivial partitions more important than trivial partitions?
  - Are trivial partitions easier to understand?

8

Research Questions and Interview Questions (cont'd)

- RQ3: Can organizing a changeset using ClusterChanges' decomposition help reviewers?
  - Does the decomposition help reviewers understand changes?
  - Does the decomposition help structure the changes in a code review?
  - Would you like to use the tool for your next code review?

9

RQ1

- Of the 20 participants, 16 said that the nontrivial partitions were both correct and complete
  - I.e., the nontrivial partitions were indeed independent, the diff-regions within each partition were related, and there were no missing conceptual groups
  - 14 developers would have moved some of the trivial changes (not more than 3) to one of the nontrivial partitions

10

RQ2

- Some of the trivial partitions were incorrect: they should have been included in a nontrivial partition
- Nontrivial partitions are not necessarily more important
- Trivial partitions are easier to understand

11

RQ3

- All participants were positive about the general concept of ClusterChanges
  - To help understand large changesets
  - To help assign reviewers to a specific partition

12

Discussion

- Missed Relations
  - E.g., overridden methods, commonly used tags or annotations
  - Focus on after-files, so miss relationship based on deleted code
  - External framework usage and XML files

13