Using Changeset Descriptions to Support Feature Location

Muslim Chochlov. Supervised by Dr. Jim Buckley and Dr. Michael English

1 Motivation

» More than 60% of the software development budget is spent on software maintenance tasks.

» Program comprehension contributes to more than 50% of developers' software maintenance effort.

» During comprehension, developers often need to "feature locate" - (find a foothold into a specified user functionality in source code)

» Usually developers resort to textual code search which often employs Information Retrieval approaches. In the majority of cases though they fail to find relevant code because of the "vocabulary mismatch" between their queries and source code comments/ identifiers

» Instead we propose to assist developers with their search leveraging prior code modification descriptions available in Version Control Systems.

2 Changesets as an Alternative Textual Data Source

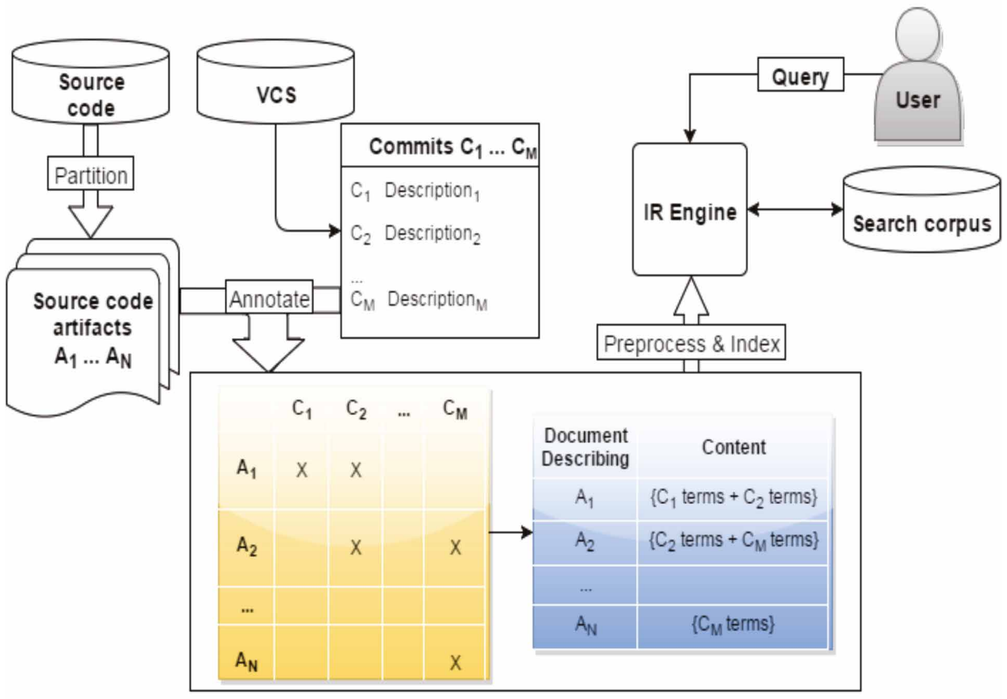
Changesets are advantageous in several ways:
» Descriptions are written abstractly/towards a goal
» Co-change with the code
» Explicit binding between changesets and source code

Changeset	Description
ae8fb47b	Landing Steve Yegge's new Abstract Syntax Tree (AST) API changes.
a06375a7	More progress on strict mode.
8dd43f8f	Disable tests as needed to get all tests passing, and leave TODOs for broken/missing parser functionality.
16d910b4	Restore revision history.

```
ae8fb47b    public ScriptNode transformTree(AstRoot root) {
ae8fb47b        currentScriptOrFn = root;
a06375a7        this.inUseStrictDirective = root.isInStrictMode();
ae8fb47b        int sourceStartOffset = decompiler.getCurrentOffset();
ae8fb47b
8dd43f8f        if (Token.printTrees) {
8dd43f8f            System.out.println("IRFactory.transformTree");
8dd43f8f            System.out.println(root.debugPrint());
8dd43f8f        }
ae8fb47b        ScriptNode script = (ScriptNode)transform(root);
ae8fb47b
ae8fb47b        int sourceEndOffset = decompiler.getCurrentOffset();
ae8fb47b        script.setEncodedSourceBounds(sourceStartOffset,
ae8fb47b                                      sourceEndOffset);
ae8fb47b
ae8fb47b        if (compilerEnv.isGeneratingSource()) {
ae8fb47b            script.setEncodedSource(decompiler.getEncodedSource());
ae8fb47b        }
ae8fb47b
ae8fb47b        decompiler = null;
ae8fb47b        return script;
16d910b4    }
```

Several factors could impact the efficiency of FLT when changesets are used:
» Granularity of artefacts
» Recentness of included changesets
» Type of included changesets

3 The Technique



» **Partition**
Divide source code into a set of artefacts of a given granularity.

» **Annotate**
Extract and match changesets with relevant artefacts to collectively describe them.

» **Pre-process & Index**
Clean text and transform data into format ready to be consumed by the IR engine

4 Current and Future Work

Initial empirical study results

» Analysed 39 requests from 2 open source projects Rhino and Mylyn.Tasks

» When compared to existing textual FLTs that use comments and identifiers, our approach utilizing changeset descriptions showed similarly efficient results

» At the method level granularity, there was a noticeable decrease in terms of effort required to locate a relevant part of a feature in source code

» Inclusion of all changesets in older systems seemed to negatively affect the efficiency of the technique.

Directions for further work

» A large scale experiment statistical comparing the changeset approach to the comments/identifiers approach.

» Assessment to see if filtering certain types of changesets could further improve the technique.

» Evaluating if a combination of comments identifiers and changeset descriptions could lead to a more efficient technique.

Ireland's European Structural and Investment Funds Programmes 2014-2020
Co-funded by the Irish Government and the European Union
European Union
European Regional Development Fund
United Technologies Research Center
SOUTHERN Regional Assembly
Promoting our Region