An Exploratory Study on Assessing Feature Location Techniques

Meghan Revelle and Denys Poshyvanyk

Software Engineering Maintenance and Evolution Research Unit
at the College of William and Mary

Goals of our work

- Investigate approaches that use textual, dynamic, and static analyses for feature location
- Evaluate in terms of ability to find *near-complete* feature implementations
- Develop guidelines for assessment of feature location techniques
- Report results and observations from exploratory study

Textual Feature Location

- Use Information Retrieval
  - Latent Semantic Indexing (LSI)
  - Output: List of ranked methods
- Two types
  - *nl-query*
  - *method-query*
- Approaches
  - IR_{query}
  - IR_{seed}

Method	Cosine Similarity
Search.setReverseSearch	0.63
Search.find	0.53
SearchMatcher.nextMatch	0.48
Search.getReverseSearch	0.47
…	…

IR: Information Retrieval

Textual + Dynamic Feature Location

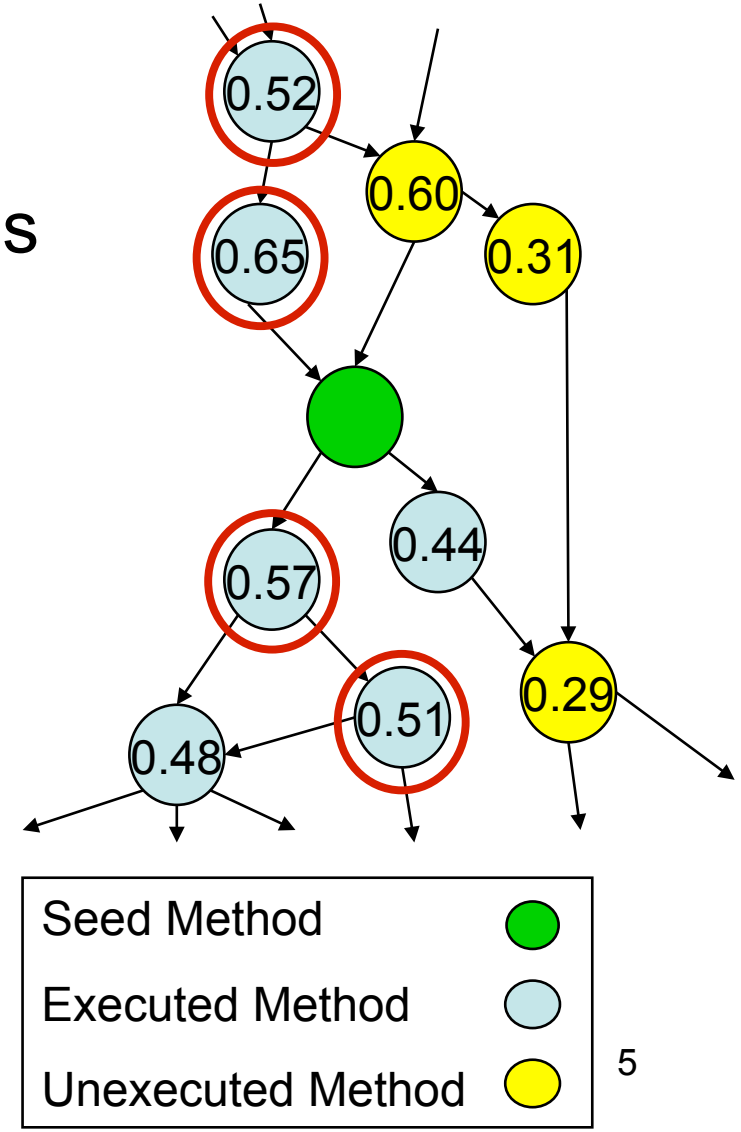
- Prune a list of ranked methods
- Two types
  - *Full trace*
  - *Marked trace*
- Approaches
  - $IR_{query} + Dyn_{marked}$
  - $IR_{query} + Dyn_{full}$
  - $IR_{seed} + Dyn_{marked}$
  - $IR_{seed} + Dyn_{full}$

Method	Cosine Similarity
Search.setReverseSearch	0.63
Search.find	0.53
~~SearchMatcher.nextMatch~~	~~0.48~~
Search.getReverseSearch	0.47
…	…

IR: Information Retrieval **Dyn**: Dynamic Analysis

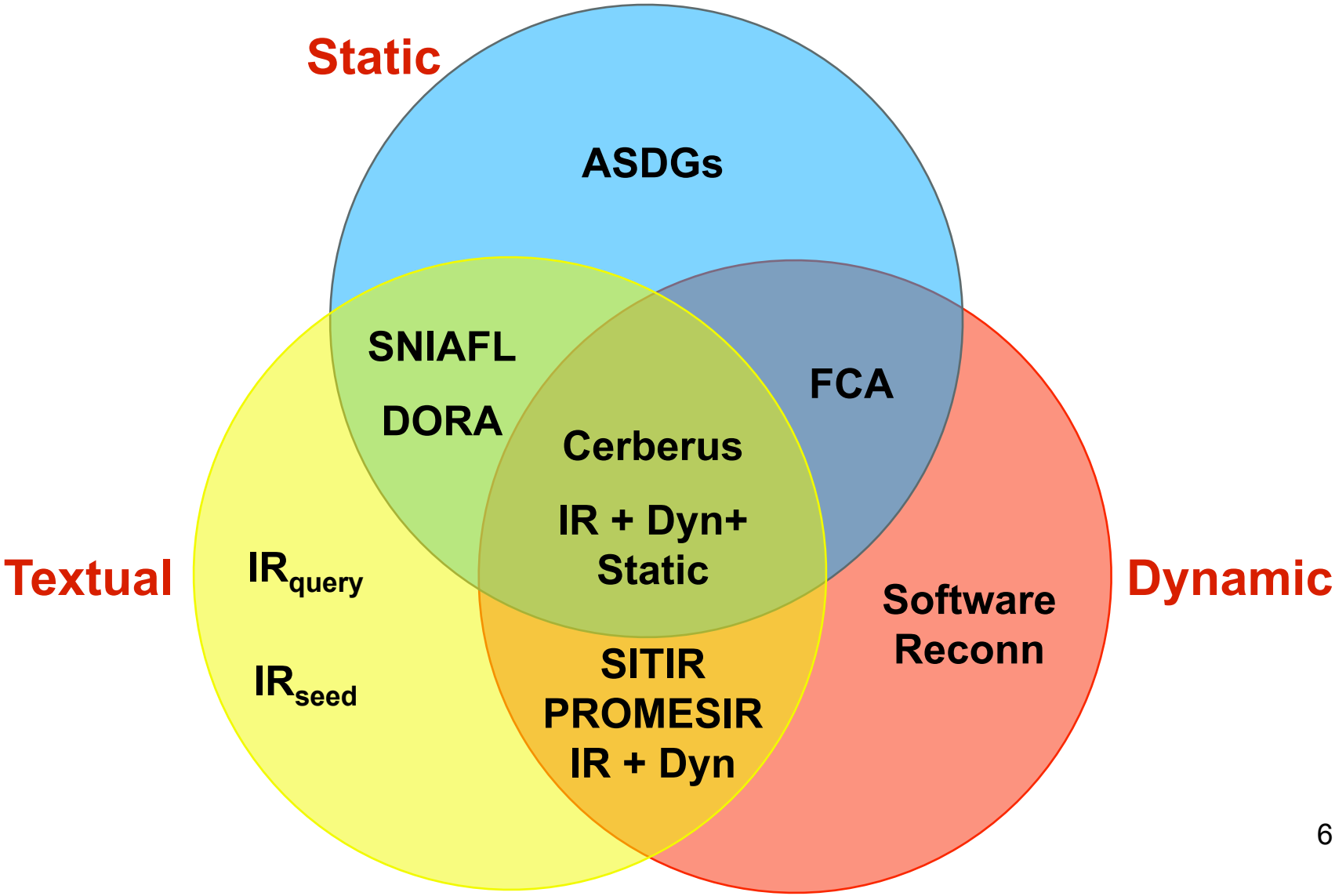
Textual, Dynamic, & Static Feature Location

- Explore PDG from seed
  - Consider executed methods
  - Textual similarity threshold
- Approaches
  - $IR_{query} + Dyn_{marked} + Static$
  - $IR_{query} + Dyn_{full} + Static$
  - $IR_{seed} + Dyn_{marked} + Static$
  - $IR_{seed} + Dyn_{full} + Static$



IR: Information Retrieval **Dyn**: Dynamic Analysis

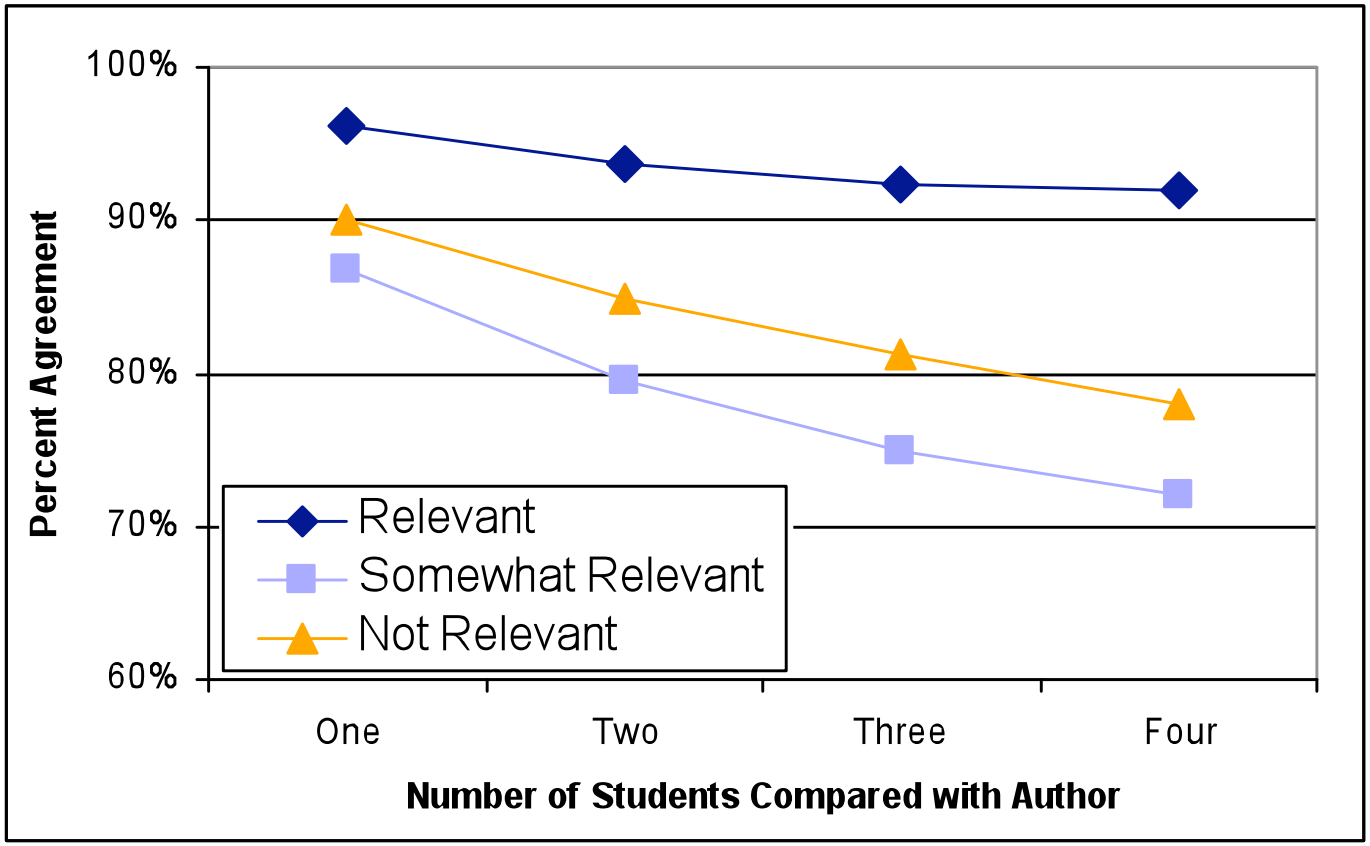
Related Work



Evaluation Options

- Artifact-based
  - Find methods from patch/bug fix
- Benchmark-based
  - Precision/recall based on existing data set
- Top N
  - Classify top 10 methods
  - Relevant, Somewhat Relevant, Not Relevant
  - Adapted guidelines

Sanity Check



Percent agreement among the volunteers and the authors for the *jEdit thick caret* feature.

Subject Systems

	jEdit 4.3pre16	Eclipse 2.1
Domain	Text editor	Development environment
Size	105KLOC; ~910 classes; ~5,530 methods	2.3MLOC; 7K classes, 89K methods
Features/ Bugs	Thick caret, Edit history text, Reverse regex search, Angle bracket matching	Double click drag, Unified Tree, Incremental Search, Repeated error message
nl-queries	Words from feature request	Words from bug report
method-queries	Text of random method from patch	Text of random method from bug fix
Scenarios	Description of feature in request	Steps to reproduce bug

Results

	jEdit			Eclipse		
Technique	**Relevant**	**Somewhat**	**Not**	**Relevant**	**Somewhat**	**Not**
IR_{query}	12.5%	15%	72.5%	22.5%	12.5%	65%
IR_{seed}	12.5%	20%	67.5%	12.5%	22.5%	65%
$IR_{query} + Dyn_{marked}$	30%	20%	50%	25%	5%	70%
$IR_{query} + Dyn_{full}$	15%	22.5%	62.5%	25%	12.5%	67.5%
$IR_{seed} + Dyn_{marked}$	20%	15%	65%	27.5%	25%	47.5%
$IR_{seed} + Dyn_{full}$	15%	27.5%	57.5%	27.5%	35%	42.5%
$IR_{query} + Dyn_{marked} + Static$	30%	17.5%	52.5%	30%	12.5%	57.5%
$IR_{query} + Dyn_{full} + Static$	12.5%	25%	62.5%	30%	12.5%	57.5%
$IR_{seed} + Dyn_{marked} + Static$	17.5%	17.5%	65%	30%	15%	55%
$IR_{seed} + Dyn_{full} + Static$	12.5%	30%	57.5%	27.5%	22.5%	50%
Average	**17.5%**	**21.25%**	**61.25%**	**24.75%**	**19.5%**	**55.75%**

Summary of Observations

- No feature location technique is universally successful at finding near-complete feature implementations
- *Method-queries* perform as well as *nl-queries*
- *Marked* traces outperform *full* traces
- Complete Results
  - www.cs.wm.edu/~denys/data/icpc09/

References

- Eaddy, M., Aho, A. V., Antoniol, G., and Guéhéneuc, Y. G., "CERBERUS: Tracing Requirements to Source Code Using Information Retrieval, Dynamic Analysis, and Program Analysis", in Proc. of ICPC'08, Amsterdam, The Netherlands, 2008.
- Eisenbarth, T., Koschke, R., and Simon, D., "Locating Features in Source Code", *TSE*, vol. 29, no. 3, March 2003, pp. 210 - 224.
- Hill, E., Pollock, L., and Vijay-Shanker, K., "Exploring the Neighborhood with Dora to Expedite Software Maintenance", in Proc. of ASE'07, November 2007, pp. 14-23.
- Liu, D., Marcus, A., Poshyvanyk, D., and Rajlich, V., "Feature Location via Information Retrieval based Filtering of a Single Scenario Execution Trace", in Proc. of ASE'07, November 5-9 2007.
- Marcus, A., Sergeyev, A., Rajlich, V., and Maletic, J., "An Information Retrieval Approach to Concept Location in Source Code", in Proc. of WCRE'04, Delft, The Netherlands, Nov. 9-12 2004, pp. 214-223.
- Poshyvanyk, D., Guéhéneuc, Y. G., Marcus, A., Antoniol, G., and Rajlich, V., "Feature Location using Probabilistic Ranking of Methods based on Execution Scenarios and Information Retrieval", *TSE,* vol. 33, no. 6, June 2007, pp. 420-432.
- Rajlich, V., "Modeling Software Evolution by Evolving Interoperation Graphs", *Annals of Software Engineering*, vol. 0, 2000, pp. 235-248.
- Robillard, M. P., Shepherd, D., Hill, E., Vijay-Shanker, K., and Pollock, L., "An Empirical Study of the Concept Assignment Problem", McGill University June 2007.
- Wilde, N. and Scully, M., "Software Reconnaissance: Mapping Program Features to Code", *Software Maintenance: Research and Practice*, vol. 7, 1995, pp. 49-62.
- Zhao, W., Zhang, L., Liu, Y., Sun, J., and Yang, F., "SNIAFL: Towards a Static Non-interactive Approach to Feature Location", *TOSEM*, vol. 15, no. 2, 2006, pp. 195-226.

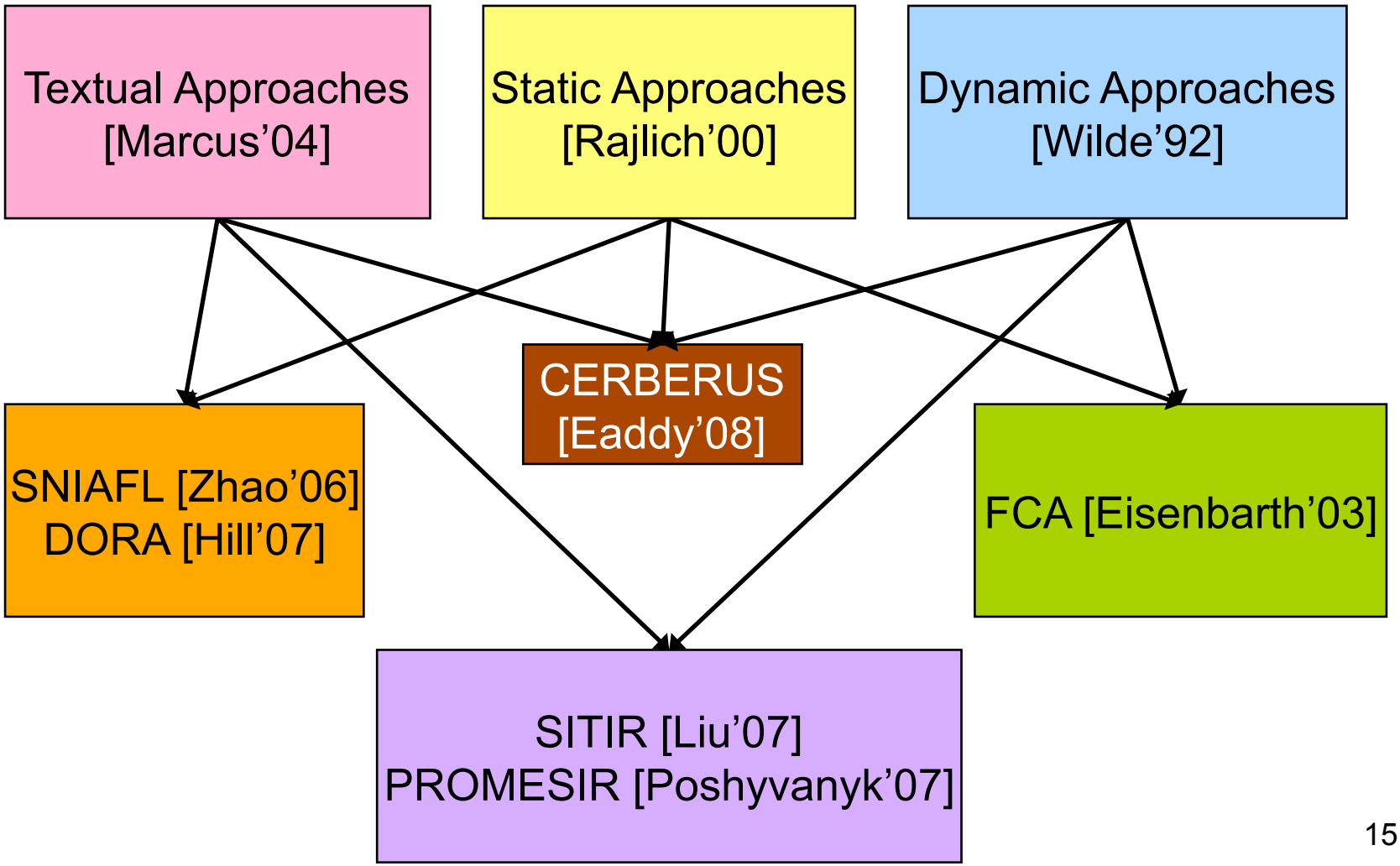
Threats to Validity

- Subjectivity of evaluation
  - Formalized how to determine relevance
  - Compared results with four volunteers for one feature
- Query construction & seed selection
  - Query terms from change requests/bug reports
  - Seeds randomly selected from patches
- Dynamic analysis
  - One scenario per feature
- Small scope of investigation
  - Two systems, four features each

Guidelines

- Method names that are similar to the words in the feature's description are good indicators of possibly relevant code, but the method's source code should be inspected to ensure the method is actually relevant to the feature.
- Determine if the method is relevant to the feature by asking "*Would it be useful to know that this method is associated with the feature if I had to modify the feature in the future?*"
- If most of the code in the method seems relevant to the feature, classify the method as *Relevant*. If some code within the method seems relevant but other code in the method is irrelevant to the feature, classify the method as *Somewhat Relevant*. If no code within the method seems relevant to the feature, classify it as *Not Relevant*.
- If unable to classify the method by reviewing its code, explore the method's structural dependencies, i.e. what other methods call it and are called by it. If the method's dependencies seem relevant, then the method probably is also.

Related Work



Locating features in jEdit

- Version 4.3pre16
  - 105KLOC; ~910 classes; ~5,530 methods
- Patches submitted for new feature requests
  - Global option for thick caret
  - Ability to edit history text
  - Reverse regex search
  - Add angle bracket matching
- Queries
  - *nl-queries* – words from feature request
  - *method-queries* – method from patch
- Scenarios
  - Developed from description in feature request

Locating features in Eclipse

- Version 2.1
  - 2.3MLOC; 7K classes, 89K methods
- Features associated with bugs
  - Double-click-drag to select multiple words broken
  - UnifiedTree should ensure file/folder exists
  - Add support for Emacs-style incremental search
  - Repeated error message when deleting and file in use
- Queries
  - *nl-queries* – words from bug report
  - *method-queries* – method from bug fix
- Scenarios
  - Steps to reproduce bug in bug report

jEdit Results

Technique	Relevant	Somewhat Relevant	Not Relevant
IR_{query}	12.5%	15%	72.5%
IR_{seed}	12.5%	20%	67.5%
$IR_{query} + Dyn_{marked}$	30%	20%	50%
$IR_{query} + Dyn_{full}$	15%	22.5%	62.5%
$IR_{seed} + Dyn_{marked}$	20%	15%	65%
$IR_{seed} + Dyn_{full}$	15%	27.5%	57.5%
$IR_{query} + Dyn_{marked} + Static$	30%	17.5%	52.5%
$IR_{query} + Dyn_{full} + Static$	12.5%	25%	62.5%
$IR_{seed} + Dyn_{marked} + Static$	17.5%	17.5%	65%
$IR_{seed} + Dyn_{full} + Static$	12.5%	30%	57.5%

Eclipse Results

Technique	Relevant	Somewhat Relevant	Not Relevant
IR_{query}	22.5%	12.5%	65%
IR_{seed}	12.5%	22.5%	65%
$IR_{query} + Dyn_{marked}$	25%	5%	70%
$IR_{query} + Dyn_{full}$	25%	12.5%	67.5%
$IR_{seed} + Dyn_{marked}$	27.5%	25%	47.5%
$IR_{seed} + Dyn_{full}$	27.5%	35%	42.5%
$IR_{query} + Dyn_{marked} + Static$	30%	12.5%	57.5%
$IR_{query} + Dyn_{full} + Static$	30%	12.5%	57.5%
$IR_{seed} + Dyn_{marked} + Static$	30%	15%	55%
$IR_{seed} + Dyn_{full} + Static$	27.5%	22.5%	50%

Acknowledgements

- David Coppit
- Maksym Petrenko
- Andrian Marcus
- Václav Rajlich
- Students of CS780 at W&M