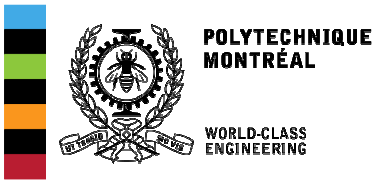
MOFAE – Multi-objective Optimization Approach to Framework API Evolution

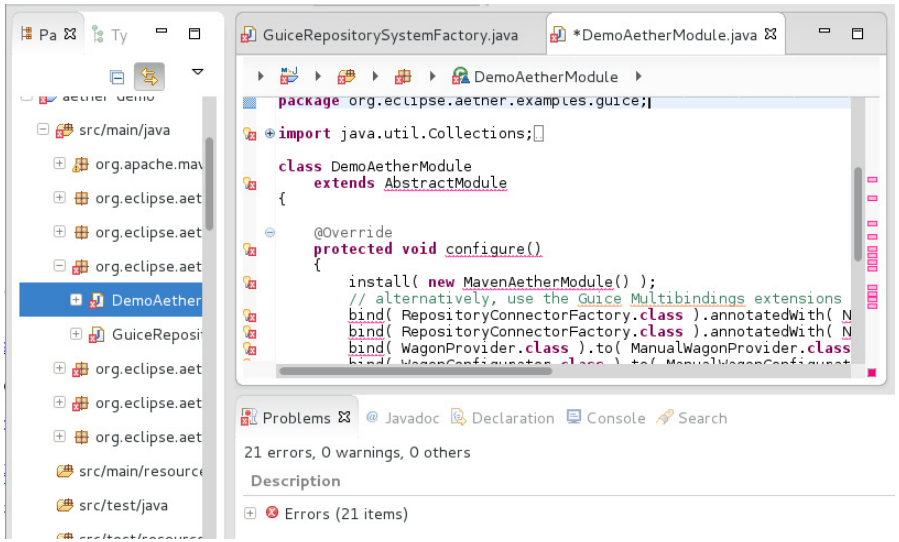
Wei Wu[1,2], Yann-Gaël Guéhéneuc[1], Giuliano Antoniol[2]
Ptidej Team[1], SOCCER Lab[2]
DGIGL, École Polytechnique de Montréal, Canada



Département de génie informatique et de génie logiciel

Framework API Evolution

- Observation
  - APIs change between two versions of a framework



Framework API Evolution

- Problem
  - No documents about how to replace the missing APIs
  - Manually searching for the replacements is time-consuming

Existing Approaches

- Call-dependency similarity
  - SemDiff [1], Schäfer et al. [2], Beagle [3], HiMa[4], AURA[5]
- Method signature text similarity
  - Kim et al. [6], Beagle [3], AURA [5]
- Software design model similarity
  - Diff-CatchUp [7]
- Software metrics similarity
  - Beagle [3]
- Comments similarity
  - HiMa [4]

Single Feature Approaches

- If the replacements are not similar in the feature, the approaches cannot report them.

```
BrowserJTree.BrowserJTree(...) is replaced by:
✓ VFSDirectoryEntryTable.VFSDirectoryEntryTable(...)
✗ BrowserView.BrowserView(...)
```

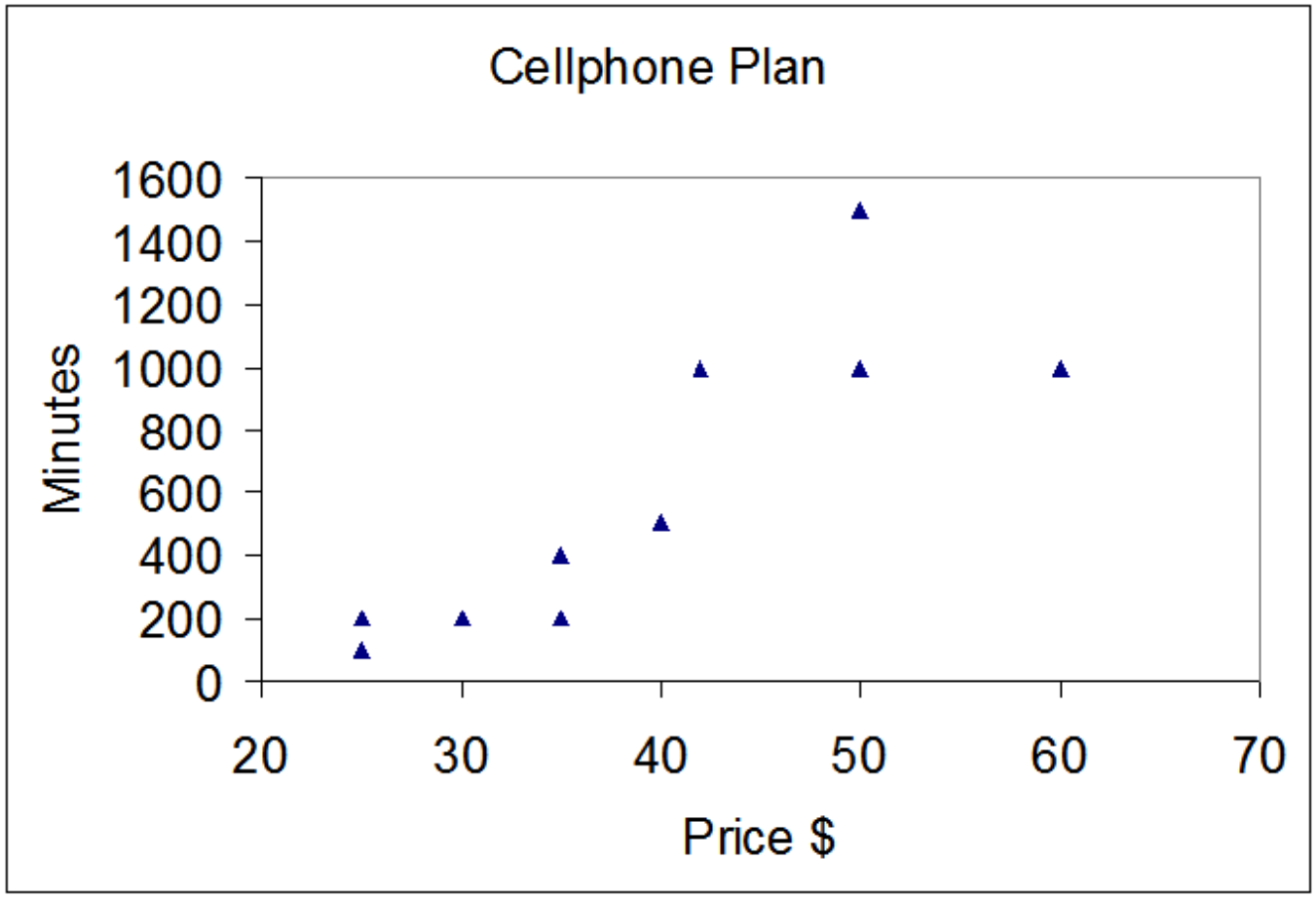
Hybrid Approaches

- Which feature to trust?
  - Prioritizing
  - Weighting system

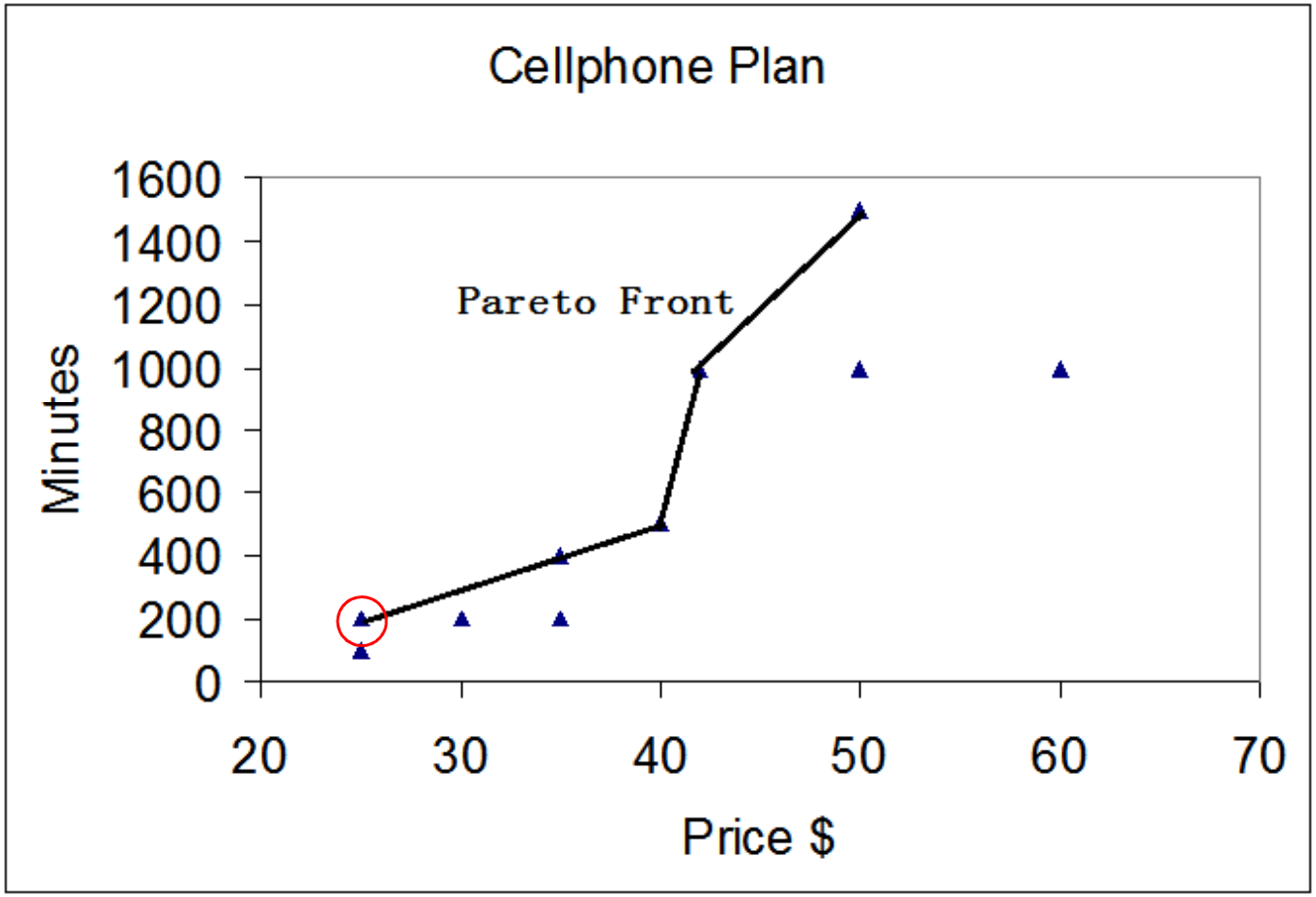
Multi-Objective Optimization

- Multi-objective optimization (MOOP) [8] is the process of finding solutions to problems with potentially conflicting objectives

How MOOP Works



How MOOP Works



MOFAE

- Recommendation system modeling framework API evolution as a MOOP problem

MOFAE

- Use some features as the objectives
- For a missing API
  - Select the Pareto front of the candidates
  - Sort the candidates on the Pareto front by the number of features in which the candidates are the most similar

Features Used by MOFAE

- Call dependency similarity
  - Confidence value and support
- Method comment similarity
  - Longest Common Subsequence (LCS)
- Method signature text similarity
  - LCS, Levenshtein Distance (LD) and Method-level Distance (MD)
- Inheritance tree similarity
  - Inheritance tree string LCS

Output of MOFAE

Target	DefaultBoxAndWhiskerDataset.getMaxOutlierValue(...)	
Objective	Inheritance Tree String LCS	Method Signature LD
1	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxRegularValue(...)	
2	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxOutlier(...)	
3	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxRegularValue(...)	
4	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxOutlier(...)	

Output of MOFAE

Target	DefaultBoxAndWhiskerDataset.getMaxOutlierValue(...)	
Objective	Inheritance Tree String LCS	Method Signature LD
1	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxRegularValue(...)	
2	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxOutlier(...)	
3	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxRegularValue(...)	
4	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxOutlier(...)	

Output of MOFAE

Target	DefaultBoxAndWhiskerDataset.getMaxOutlierValue(...)	
Objective	Inheritance Tree String LCS	Method Signature LD
1	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxRegularValue(...)	
2	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxOutlier(...)	
3	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxRegularValue(...)	
4	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxOutlier(...)	

Output of MOFAE

Target	DefaultBoxAndWhiskerDataset.getMaxOutlierValue(...)	
Objective	Inheritance Tree String LCS	Method Signature LD
1	19	-4
	~~DefaultBoxAndWhiskerXYDataset.getMaxRegularValue(...)~~	
2	19	-4
	DefaultBoxAndWhiskerXYDataset.getMaxOutlier(...)	
3	13	-3
	~~DefaultBoxAndWhiskerCategoryDataset.getMaxRegularValue(...)~~	
4	13	-3
	DefaultBoxAndWhiskerCategoryDataset.getMaxOutlier(...)	

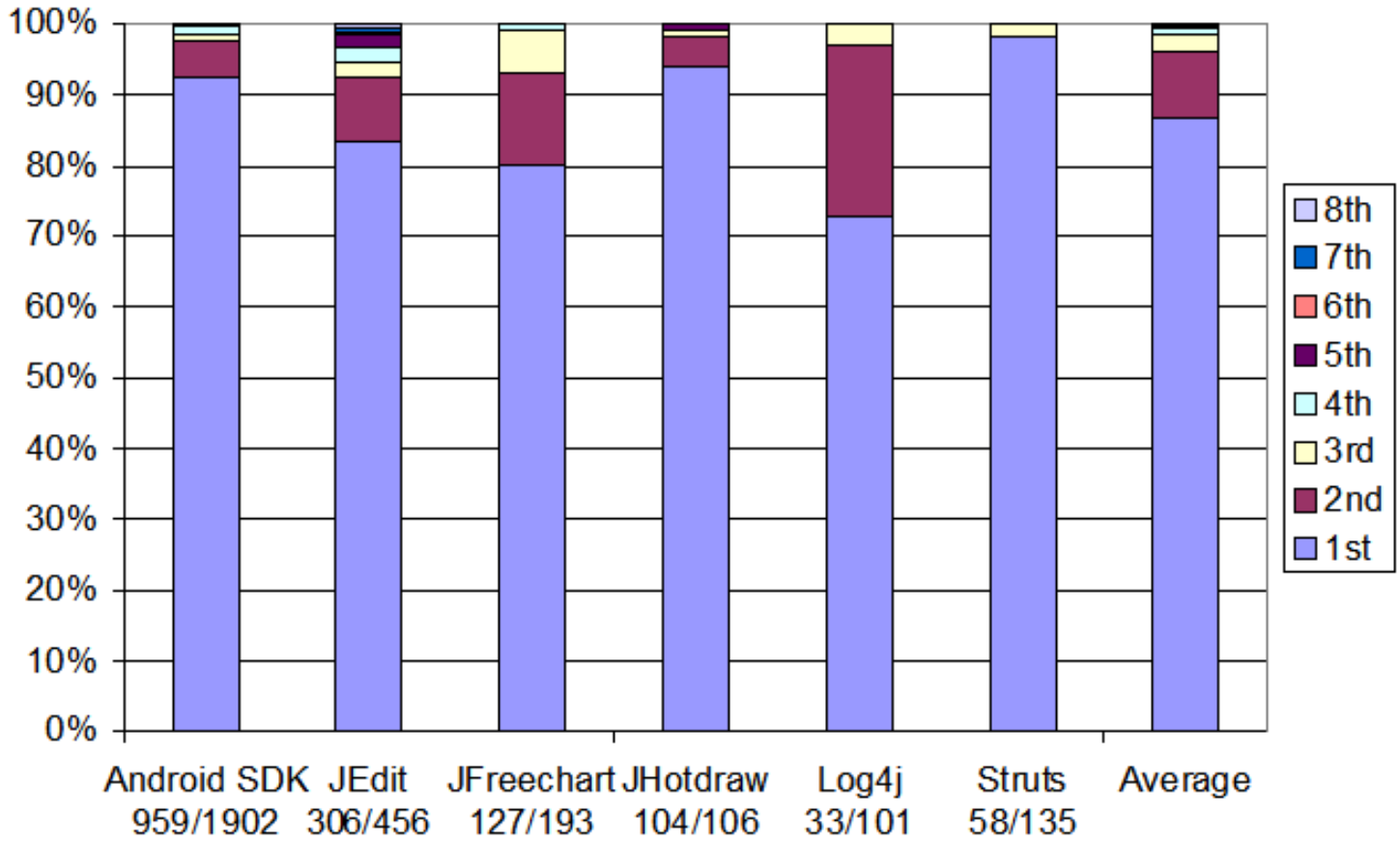
Evaluation

Subject Systems	Releases	# Methods
Android SDK	2.1_r2.1p2	20,516
	2.2.3_r2	21,214
JEdit	4.1	2,773
	4.2	3,547
JFreeChart	0.9.11	4,751
	0.9.12	5,197
JHotDraw	5.2	1,486
	5.3	2,265
Log4j	1.0.4	906
	1.1.3	1,110
Struts	1.1	5,973
	1.2.4	6,111

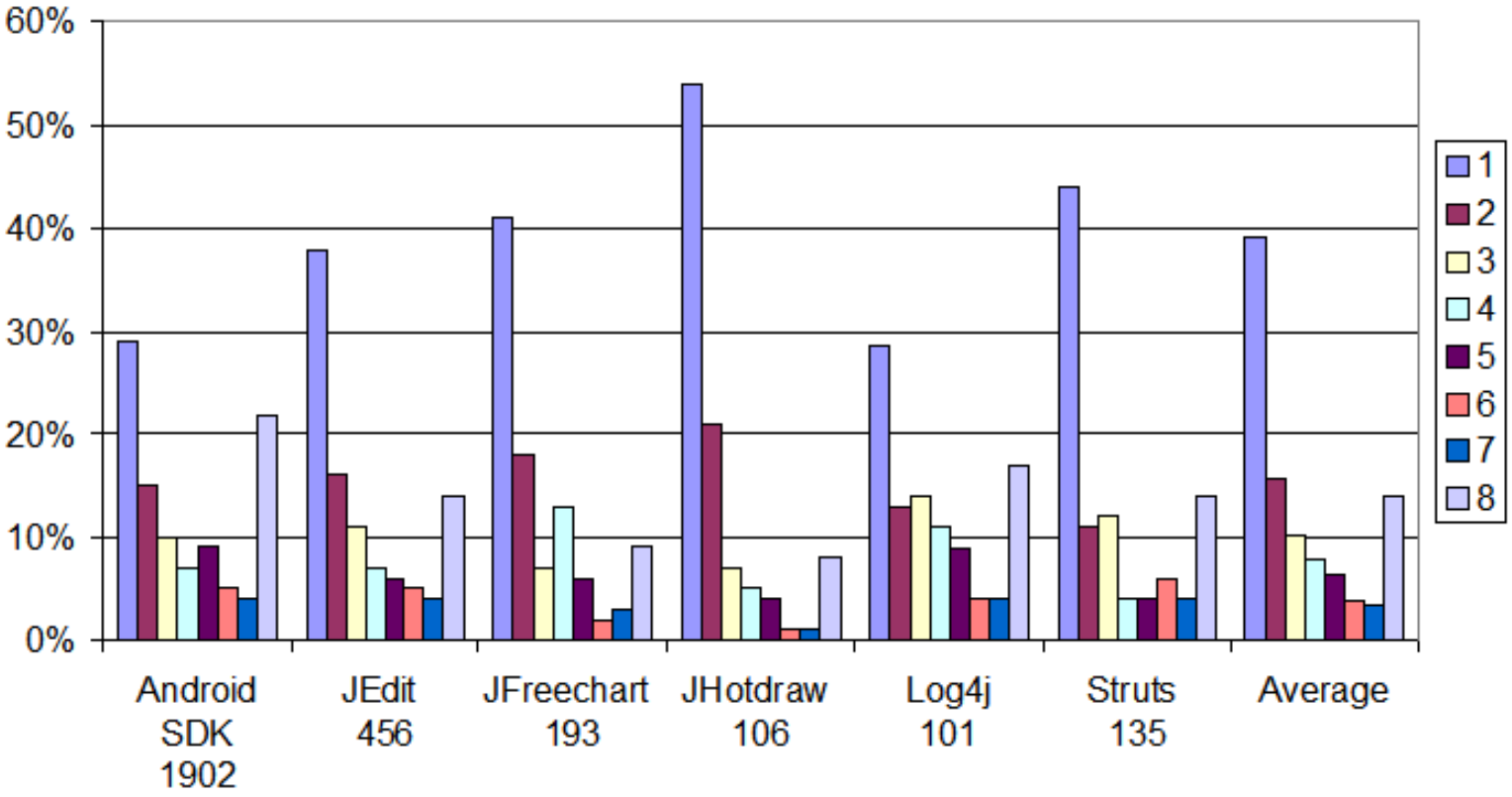
Comparison

System	Indicators	Number of Correct Replacement			
		MOFAE	Halo	AURA	Kim et al.
Android	# Correct Rule	959	914	826	772
	MOFAE Improvement	-	5%	16%	24%
JEdit	# Correct Rule	306	289	247	217
	MOFAE Improvement	-	6%	24%	41%
JFreechart	# Correct Rule	127	113	95	88
	MOFAE Improvement	-	12%	34%	44%
JHotdraw	# Correct Rule	106	99	96	81
	MOFAE Improvement	-	7%	10%	31%
Log4j	# Correct Rule	33	30	21	33
	MOFAE Improvement	-	6%	57%	0%
Strutus	# Correct Rule	58	58	56	56
	MOFAE Improvement	-	0%	4%	6%
Average Improvement		-	6%	24%	24%
Total Improvement		18%			

Correct Recommendation Position



Recommendation List Size



Limitations

- Semi-automatic
- Depends on the features

Conclusion

- MOFAE can detect correct replacements for 18% more missing APIs than previous approaches
- 87% correct recommendations are the first, 99% correct recommendations are in top three
- Average size 3.7, median size 2.2, maximum size 8

References

- [1] B. Dagenais and M. P. Robillard. Recommending adaptive changes for framework evolution. TOSEM 2011.
- [2] T. Schäfer, J. Jonas, and M. Mezini. Mining framework usage changes from instantiation code. ICSE 2008.
- [3] M. W. Godfrey and L. Zou. Using origin analysis to detect merging and splitting of source code entities. TSE 2005.
- [4] S. Meng, X. Wang, L. Zhang, and H. Mei. A history-based matching approach to identification of framework evolution. ICSE 2012.
- [5] W. Wu, Y.-G. Guéhéneuc, G. Antoniol, and M. Kim. Aura: a hybrid approach to identify framework evolution. ICSE 2010.
- [6] M. Kim, D. Notkin, and D. Grossman. Automatic inference of structural changes for matching across program versions. ICSE 2007.
- [7] Z. Xing and E. Stroulia. API-evolution support with Diff-CatchUp. TSE 2007.
- [8] Y. Sawaragi, H. Nakayama, and T. Tanino. Theory of multiobjective optimization. Academic Press, 1985.

MOFAE – Multi-objective Optimization Approach to Framework API Evolution

Thank You!

POLYTECHNIQUE MONTRÉAL

WORLD-CLASS ENGINEERING

Département de génie informatique et de génie logiciel