On the Effectiveness of Accuracy of Automated Feature Location Technique

Takashi Ishio

OSAKA UNIVERSITY

Shinpei Hayashi

Tokyo Institute of Technology

Hiroshi Kazato

NTT DATA INTELLILINK Corp.

Tsuyoshi Oshima

NTT Software Innovation Center

Feature Location

- A first step of software maintenance
  - Which modules/functions/methods implement a feature?
  - How they interact?

- Feature location process (for a Java program) [Wang, 2011]

  A feature description ➔ A list of relevant methods

  - Search keywords to find *seed* methods
  - Explore the *seed* methods and their neighbors

Limited Accuracy

- Industrial developers would like to locate a feature completely.
  - To make a plan for their maintenance task.
    - How to change, review and test the code
- Manual feature location is not so precise.
  - It is required only when no one knows the complete implementation of a feature.
  - In [Wang, 2011], both precision and recall are at most 75%.

Automated Feature Location Techniques

- Information retrieval (IR) is a popular approach.
  - Latent Semantic Indexing [Marcus, 2004]
  - + Dynamic Analysis [Poshyvanyk, 2007]
  - + Static Analysis [Eaddy, 2008]

Developer

Feaure Description

jEdit full-screen mode …

Search Result

1. JEditBuffer.setMode(String)
2. View.toggleFullScreen()
3. ViewOptionPane._init()
4. View.propertiesChanged()

…

ViewOptionPane._save()

Automated Feature Location Tool

Can developers validate a result?

- Manual validation problem in traceability area
  - [Cuddeback, 2010], [Kong, 2011], [Dekhtyar, 2012]
  - Analysts validated links between requirements and system tests.

REQ.	TEST1	TEST2	TEST3	…
Req1	X ✓	✓	✓	
Req2	✗	✓	X ✓	
Req3	X ✗	✗	✓	
Req4	X ✗	✗	✗	
…				

REQ.	TEST1	TEST2	TEST3	…
Req1	X			
Req2	X		X	
Req3		X		
Req4		X	X	
…				

Traceability Researchers' Observations

High

Recall

Low

REQ.	T1	T2	T3	...
R1	X		X	
R2	X	X		
R3		X	X	
R4	X	X	X	
...				

Remove invalid links
(and some true links)

REQ.	T1	T2	T3	...
R1	X		X	
R2	X			
R3				
R4		X	X	
...				

Accuracy decreased!

REQ.	T1	T2	T3	...
R1	X			
R2	X			
R3		X		
R4		X	X	
...				

Significantly improved

Identify missing links
(and some invalid links)

REQ.	T1	T2	T3	...
R1	X			
R2			X	
R3	X			
R4	X			
...				

REQ.	T1	T2	T3	...
R1	X		X	
R2				
R3				
R4		X		
...				

Low

Precision

High

Our Experiment

- Can developers locate a feature using a result of an automated feature location tool?
  - We asked subjects to locate a feature using a list of methods.
    - The accuracy of the lists is artificially controlled.
    - We measured precision, recall and F-measure.

Subjects and Dataset

- 20 subjects in three organizations
  - 8 students in Osaka University,
  - 8 students in Tokyo Institute of Technology,
  - 4 developers in R&D Division of NTT
  - Java experience: 2—16 years
- Dataset
  - Features and goldsets in Dit's Benchmarks [Dit, 2013]
  - We have added feature descriptions.
    - Feature requests in the dataset do not explain the added features.

Tasks

Feature	Goldset	Accurate (Better) List			Less Accurate (Worse) List		
	#meth	#Gold	Prec.	Recall	#Gold	Prec.	Recall
muCommander 1	32	10	**1.00**	0.31	8	0.80	0.25
muCommander 2	6	6	0.60	**1.00**	3	0.30	0.50
jEdit 1	13	10	**1.00**	0.77	4	0.40	0.31
jEdit 2	6	6	0.60	**1.00**	3	0.30	0.50
jEdit 3	10	10	**1.00**	**1.00**			

- A pair of an accurate list and a less accurate list
  - Each list includes 10 methods, selected using LSI [Gethers, 2012].
  - Larger features have high precision, smaller ones have high recall.
- A subject uses two accurate lists, two less-accurate lists.
- 30 minutes for each feature

Environment

- Eclipse enhanced with FL-Player plug-in

Java - jEdit/src/org/gjt/sp/jedit/buffer/DefaultFoldHandlerProvider.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Packa: jEdit, muCommander

DefaultFoldHandlerProvider.jav | JEditTextArea.java

```
/**
 * Returns an array containing the names of all
 * handlers.
 *
 * @since jEdit 4.0pre6
 */
public String[] getFoldModes()
{
    return folds.keySet().toArray(new String[fo
}

/**
 * Add a new FoldHander.
 *
 * @param foldHandler the new foldHandler
```

Outline: org.gjt.sp.jedit.buffe, import declarations, DefaultFoldHandlerI, folds : Map<Stri, getFoldHandler(, getFoldModes(), addFoldHandler(

Problems | Javadoc | Declaration | Task List | FL Player

jEdit 1747300 (203) - Enable customization of folding presentation

ID	Relevance	Name
1	Irrelevant	org.gjt.sp.jedit.textarea.DisplayManager.foldHandlerChanged()
2	Irrelevant	org.gjt.sp.jedit.buffer.FoldHandler.getFoldModes()
3	-	org.gjt.sp.jedit.buffer.DefaultFoldHandlerProvider.addFoldHandler(Foldl
4	Relevant	org.gjt.sp.jedit.buffer.JEditBuffer.propertiesChanged()
5	-	org.gjt.sp.jedit.buffer.DefaultFoldHandlerProvider.getFoldModes()
6	-	org.gjt.sp.jedit.textarea.TextArea.collapseFold()

Double-clicking opens a method declaration.

A subject chooses either relevant or irrelevant.

Research Questions

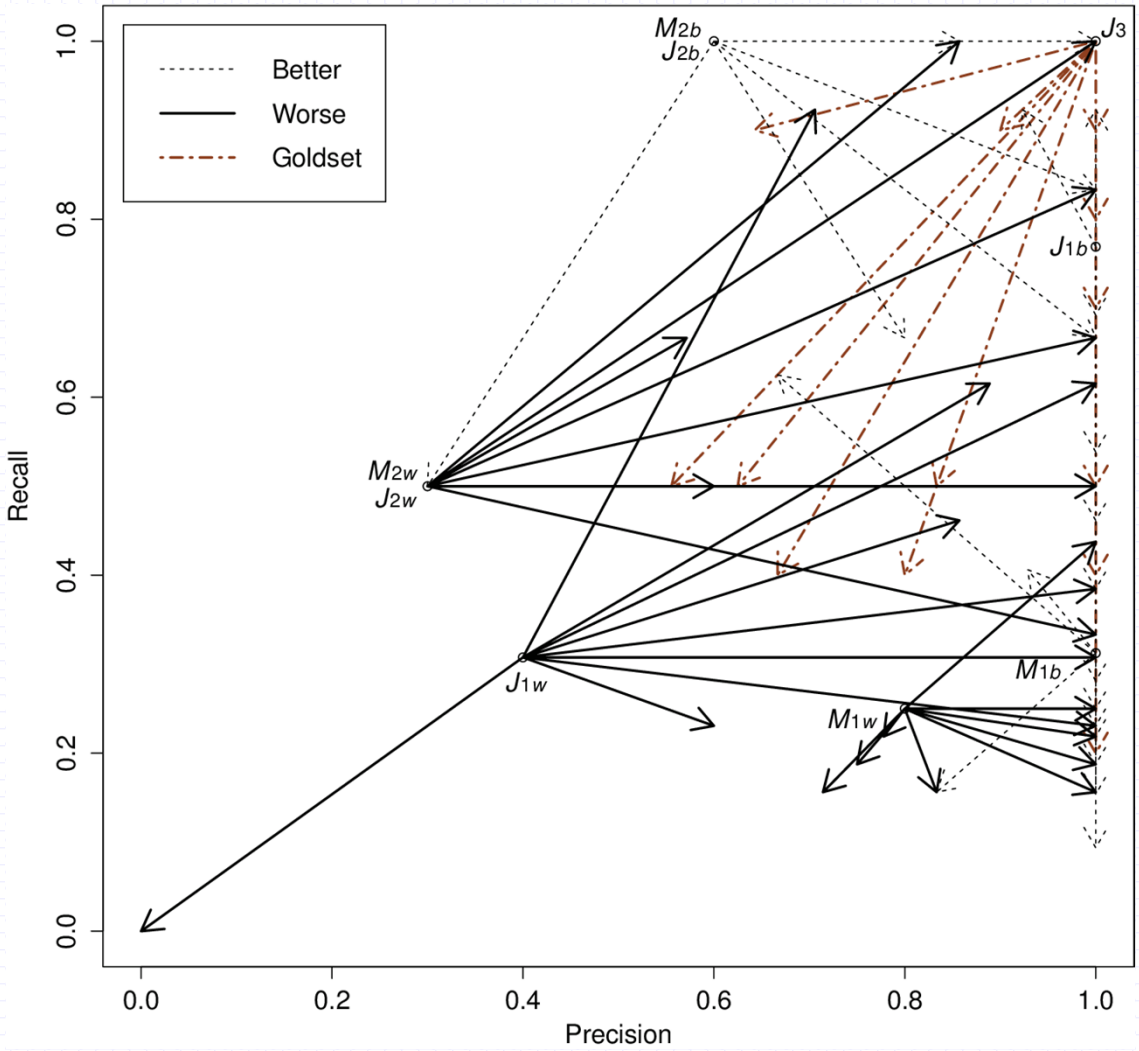
RQ1. Do better initial precision and recall engender better performance in feature location by developers?

RQ2. Which option is more important: initial precision or recall?

RQ3. How do developers spend time to validate a list of methods?

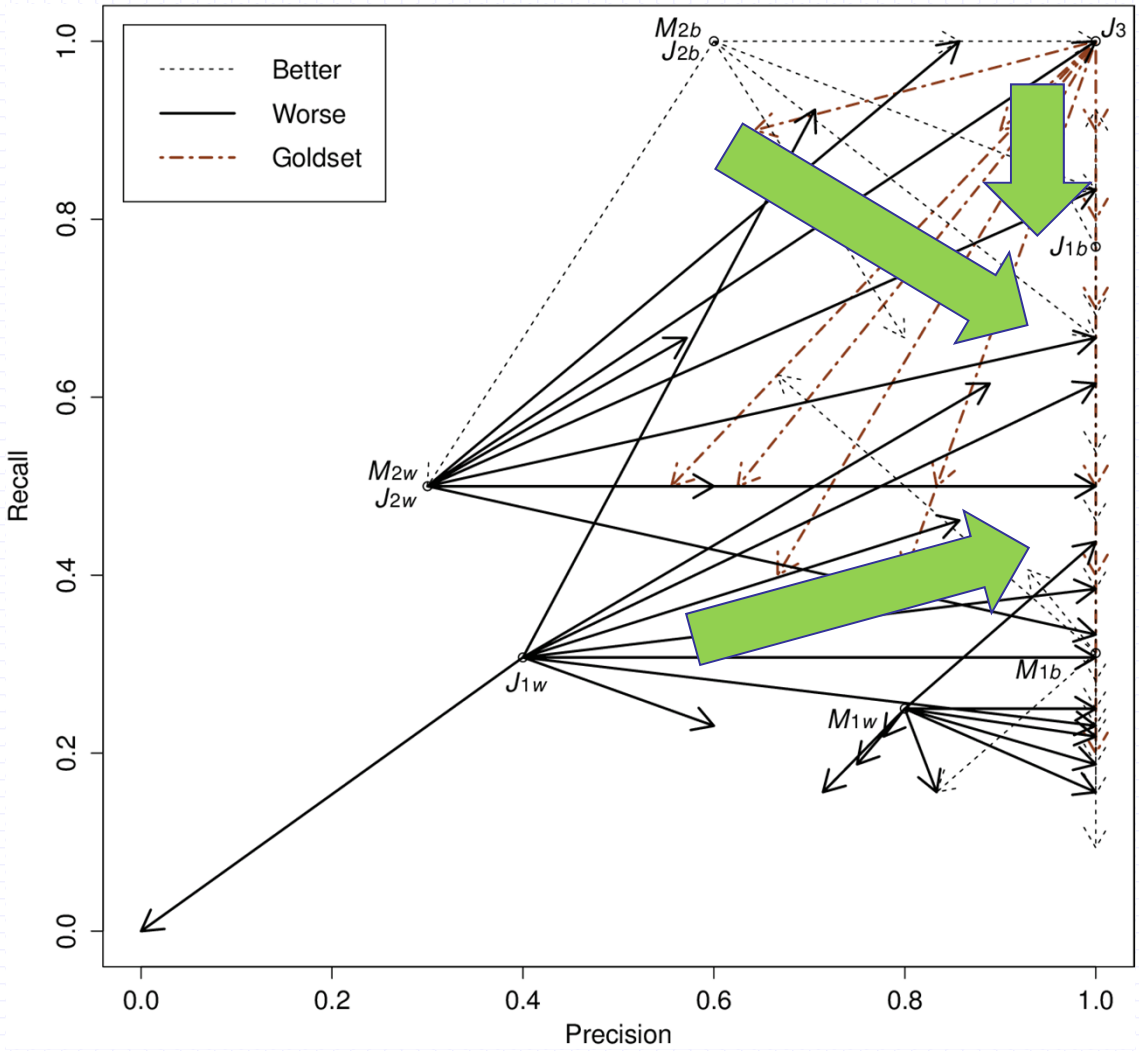
RQ4. How does a validated list differ from its initial list?

Result



Initial precision/recall
→feature location result

Result

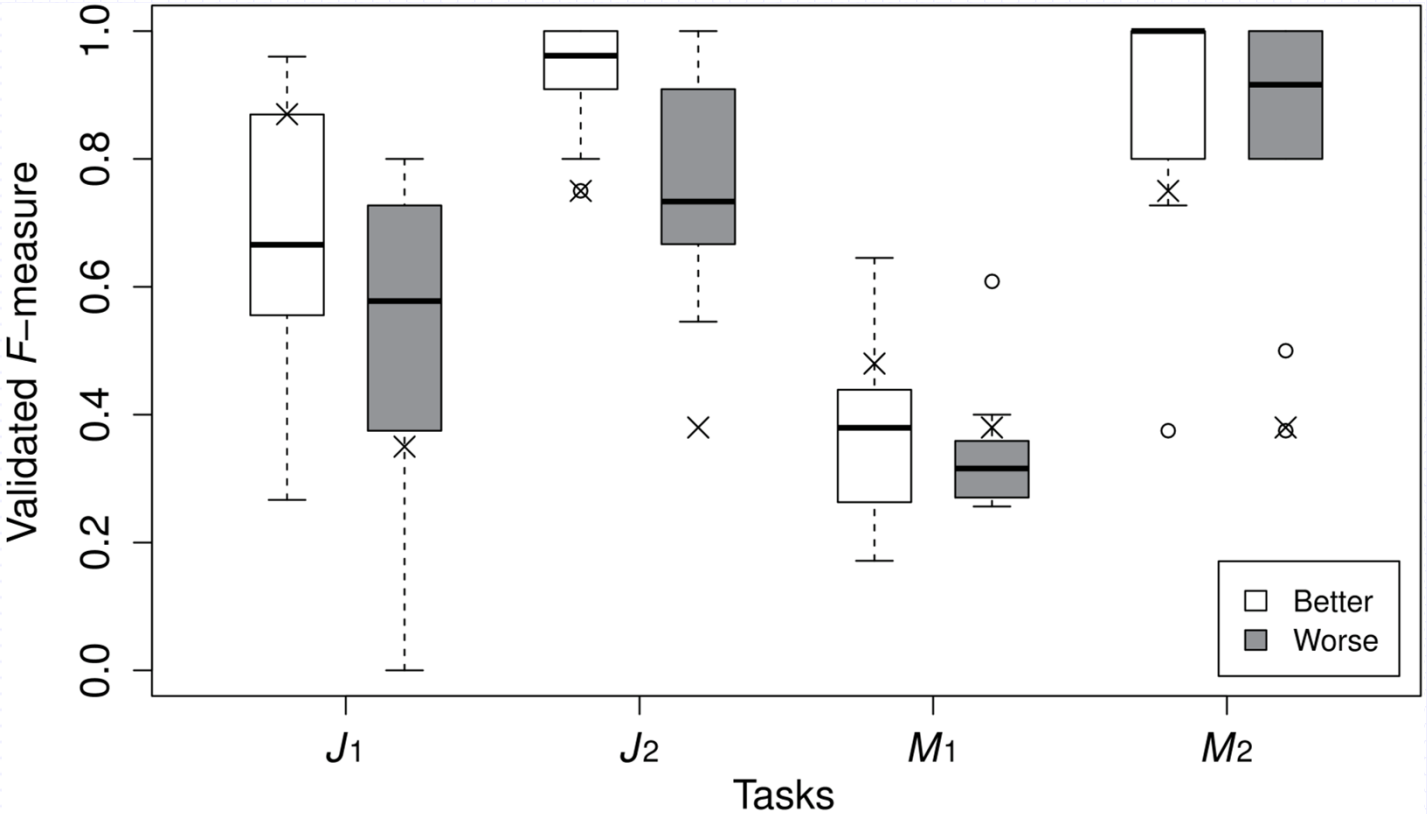


Subjects improved precision by removing false positives.

Subjects tend to decrease recall.

RQ1. Do better initial precision and recall engender better performance in feature location by developers?

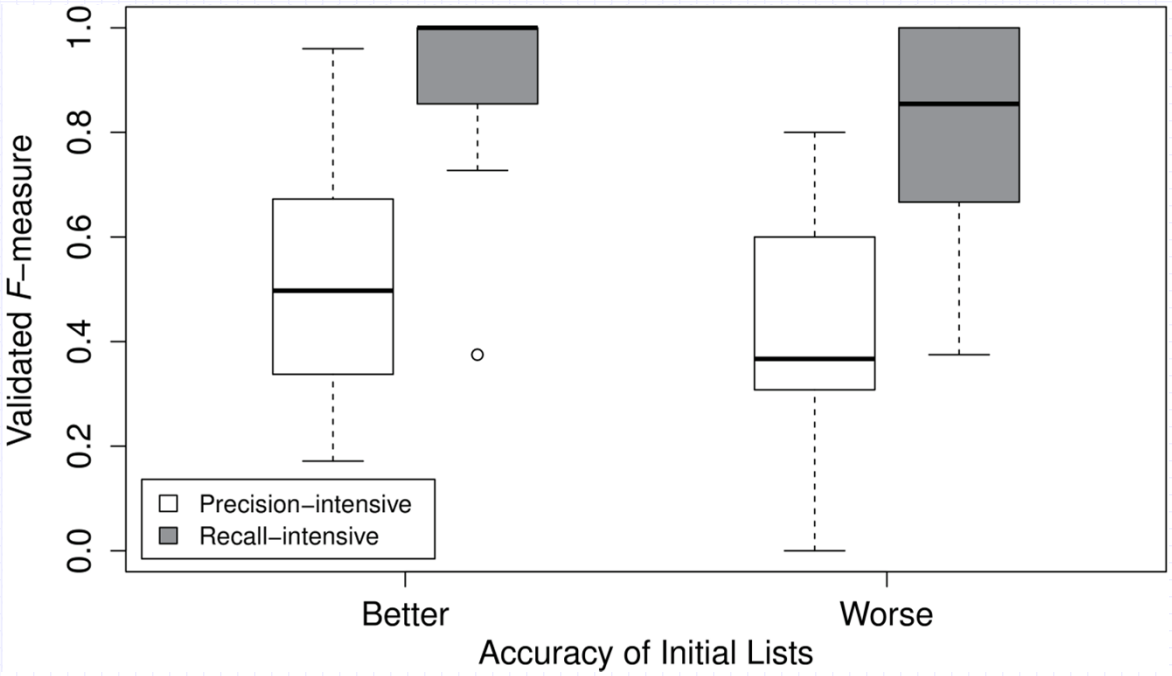
- Yes. Subjects given an accurate list performed better than subjects given a less accurate list.
  - An accurate feature location technique would be effective.



Paired t-tests rejected the null hypothesis. (p=0.0089)

RQ2. Which is important: initial precision or recall?

- Recall. Subjects given a high-recall list performed better than subjects given a high-precision list.
  - False positives are easier to identify than false negatives.



Wilcoxon signed rank test rejected the null hypothesis. ($p<10^{-6}$)

Observations

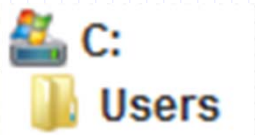
- Subjects took 20 minutes for each task.
  - No significant differences among subjects/tasks
- Subject located a complete implementation in 17 of 100 tasks.
  - Much better than a manual feature location experiment [Wang, 2011].
  - Subjects tend to remove certain methods from lists.
    - An example is shown in the next slide.

Example: jEdit's default icons

- A feature enables to choose either system icons or default icons used in a window. (Only system icons were available before the feature was introduced.)

```
//{{{ getIconForFile() method
public static Icon getIconForFile(VFSFile file,
    boolean expanded, boolean openBuffer)
{
    if (defaultIcons)
        return file.getDefaultIcon(expanded, openBuffer);
    return file.getIcon(expanded, openBuffer);
} //}}}
```

Default Icons



System Icons

- 13 of 20 subjects excluded *getDefaultIcon* from the list.
  - Because the code exists before the feature was introduced.
  - The benchmark regarded the method as a part of a feature, because the method was also affected by the feature.

Threats to Validity

- We have created method lists artificially.
  - The methods in the lists are picked up from a result of LSI, but they are not actual output of a tool.
- Each list included only 10 methods.
  - To conduct the experiment in the limited time.
  - “Top-10” is a typical usage of a search tool.
  - Industrial developers might use a longer list.

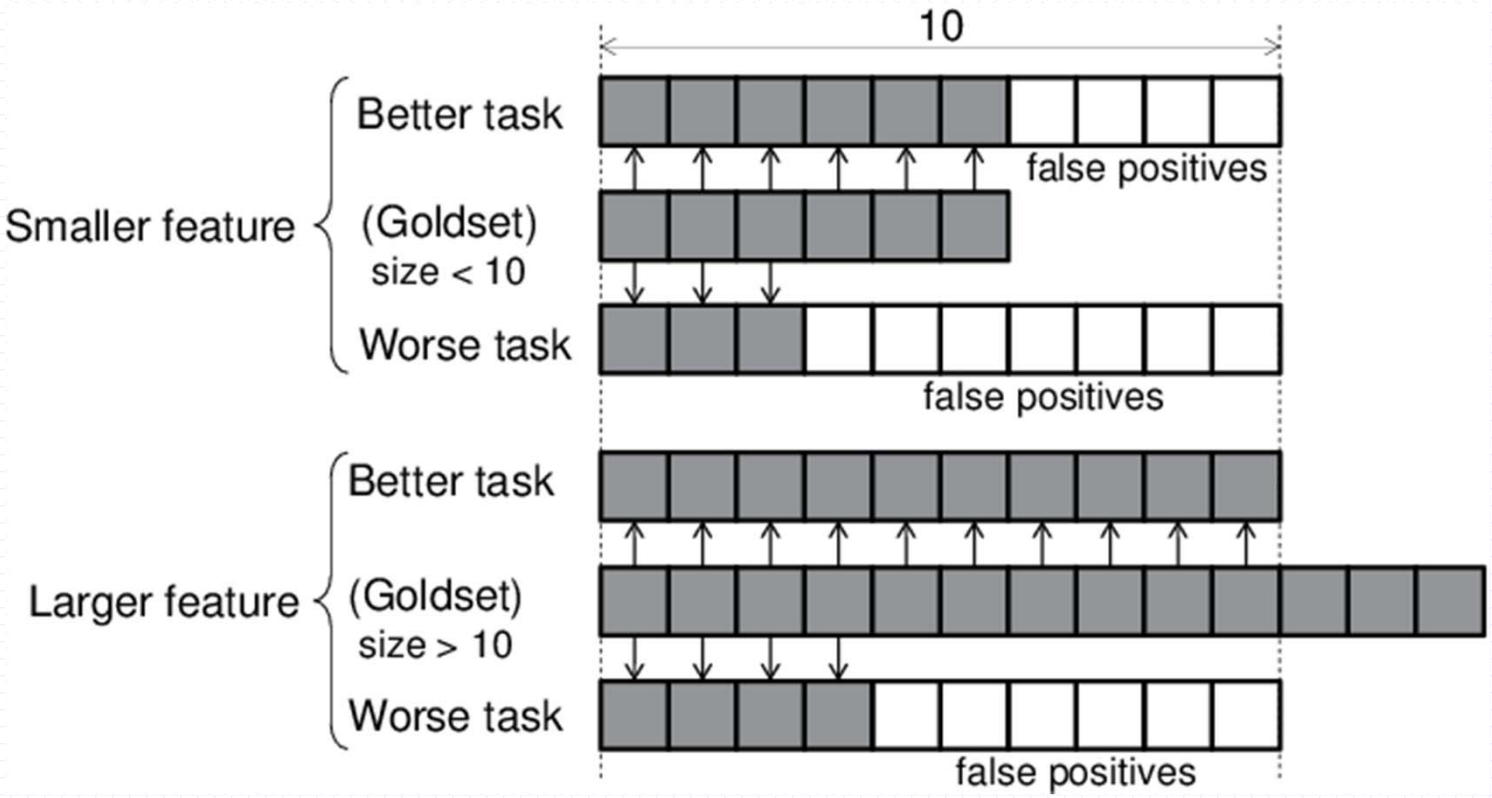
Concluding Remarks

- A controlled experiment of feature location tasks
  - Accurate feature location would be effective.
  - Recall is more important than precision.
  - Many subjects removed some relevant methods from lists.
    - A clearer feature description or some additional support may be important to avoid the problem.
    - Our next work is to analyze how feature descriptions affect feature location tasks.
- Our dataset is online. http://sel.ist.osaka-u.ac.jp/FL/
  - It has been derived from Dit's and Gethers' dataset.
  - Feature descriptions, tasks and an Eclipse plug-in.

APPENDIX

Artificially created lists

- We first determined the number of relevant methods to be included in the list.



Curation of Goldset

- Some list of goldset methods included non-existent methods.
  - It existed at the modified time, but removed until the release.
  - We manually removed such methods from the goldset.

Feature Description Refinement

- Some long description in the original dataset does not explain the feature itself.
  - A fullscreen mode for jEdit <u>would be very nice</u>. <u>Especially on netbooks with limited screen-size it is useful</u>, to get rid of the titlebar and window-borders.

Additional Feature Description

- Two paragraphs explain the basic behavior and the behavior of the feature.
  - The editor window of jEdit has a menu bar, a tool bar, and borders.
  - The new feature enlarges an editor window to full screen and hides its menu bar, a tool bar, and borders when the F11 key is pushed. Pushing the F11 key again in full screen mode transforms the window to a regular window mode. The General Options dialog allows users to enable menu, tool, and status bars in full screen mode.