Does a Programmer’s Activity Indicate Knowledge of Code?
Software Engineering Seminar 2010

Who? Jan Rüegg

From? 2007 Paper By
Thomas Fritz, University of British Columbia
Gail C. Murphy, University of British Columbia
Emily Hill, University of Delaware

When? 16th March 2010
Imagine You See Something Like This...

What?!?

```c
if (fabs (text->letter_spacing) > 0.1)
{
    PangoAttrList *attrs;
    PangoAttribute *attr;

    attrs = pango_layout_get_attributes(
            layout->layout);
    if (attrs)
        pango_attr_list_ref (attrs);
    else
        attrs = pango_attr_list_new ();

    attr = pango_attr_letter_spacing_new (
           text->letter_spacing * PANGO_SCALE);

    attr->start_index = 0;
    attr->end_index  = -1;
    ...
```
if (fabs (text->letter_spacing) > 0.1)
 {
    PangoAttrList  *attrs;
    PangoAttribute *attr;

    attrs = pango_layout_get_attributes(
            layout->layout);
    if (attrs)
        pango_attr_list_ref (attrs);
    else
        attrs = pango_attr_list_new () ;

    attr = pango_attr_letter_spacing_new ( 
            text->letter_spacing * PANGO_SCALE);
    attr->start_index = 0;
    attr->end_index = -1;
    ...

The Basic Question

How can we measure a Programmers Knowledge about source Code?

Hypothesis
The more Frequently and Recently a programmer has interacted with an element X, the higher is his Knowledge about X.
The Basic Question

How can we measure a Programmers Knowledge about source Code?

Hypothesis

The more Frequently and Recently a programmer has interacted with an element X, the higher is his Knowledge about X.
Outline

Example, Hypothesis

Motivation

DOI: Degree-Of-Interest

Subjects

Questionnaires

Significance of DOI

Problems

Improvements
Outline

Example, Hypothesis

Motivation
DOI: Degree-Of-Interest

Subjects
Questionnaires

Significance of DOI
Problems
Improvements
Outline

Example, Hypothesis

Motivation
DOI: Degree-Of-Interest

Subjects
Questionnaires

Significance of DOI
Problems
Improvements
Motivation
Who knows What?

Important for
▶ Expertise Recommendations
▶ IDE with Adaptive Interface
▶ Recommending group members

Mylyn Degree-Of-Interest Model
▶ Introduced 2005 by Mik Kersten and Gail C. Murphy
▶ Interest level in program elements
▶ Original goal: An Adaptive Eclipse Interface
Motivation
Who knows What?

Important for
▶ Expertise Recommendations
▶ IDE with Adaptive Interface
▶ Recommending group members

Mylyn Degree-Of-Interest Model
▶ Introduced 2005 by Mik Kersten and Gail C. Murphy
▶ Interest level in program elements
▶ Original goal: An Adaptive Eclipse Interface
Mylyn DOI Model
How Mylyn improves Eclipse...

How does it work?

- Each Java element has its DOI value that
- Increases with each select/edit/…
  different operations are differently weighted
- Decreases over time, when not accessed

Idea: Elements with higher DOI:
  Higher up in Eclipse lists…

Effect: Programmer spends less time searching for important items
Mylyn DOI Model
How Mylyn improves Eclipse...

How does it work?

Each Java element has its DOI value that
- Increases with each select/edit/…
  different operations are differently weighted
- Decreases over time, when not accessed

Idea
Elements with higher DOI:
Higher up in Eclipse lists…

Effect
Programmer spends less time searching for
important items
Empirical Study

Subjects
- 19 Industry Java Programmers
- Two IBM development laboratory locations
- Experience between one month to twenty years

Interactive Monitoring for five weeks
- Eclipse Mylyn project
- Modified to the needs of the study
- Show about one questionnaire per week

Evaluation
- Analysis of the collected data
- Interviews with programmers about the results
Empirical Study

Subjects

- 19 Industry Java Programmers
- Two IBM development laboratory locations
- Experience between one month to twenty years

Interactive Monitoring for five weeks

- Eclipse Mylyn project
- Modified to the needs of the study
- Show about one questionnaire per week

Evaluation

- Analysis of the collected data
- Interviews with programmers about the results
Empirical Study

Subjects
- 19 Industry Java Programmers
- Two IBM development laboratory locations
- Experience between one month to twenty years

Interactive Monitoring for five weeks
- Eclipse Mylyn project
- Modified to the needs of the study
- Show about one questionnaire per week

Evaluation
- Analysis of the collected data
- Interviews with programmers about the results
Questionnaires

**Individual Questionnaires are generated**

- After 20000, 40000 and 60000 interactions
- Random questions from each of
  - 20% highest DOI
  - 20% lowest DOI
  - 20% in the middle
- Automatically generated and verified

**Typical Flow of Events**
1. Programmer works as usual with IDE
2. IDE monitors elements for about about a week
3. A questionnaire pops up, asking about high and low DOI elements
4. Programmer answers
Questionnaires

**Individual Questionnaires are generated**

- After 20000, 40000 and 60000 interactions
- Random questions from each of
  - 20% highest DOI
  - 20% lowest DOI
  - 20% in the middle
- Automatically generated and verified

**Typical Flow of Events**

1. Programmer works as usual with IDE
2. IDE monitors elements for about a week
3. A questionnaire pops up, asking about high and low DOI elements
4. Programmer answers
**Typical Questions**

What the Subjects were asked...

```java
public class Calculator {
    public Stack<StackItem> stack;

    public Calculator(double operand) {
        stack = new Stack<StackItem>();
        stack.push(new StackNumber(operand));
    }

    public double popOperand() {
        double result = peekOperand();
        stack.pop();
        return result;
    }

    ...
```
Typical Questions
What the Subjects were asked...

Q1) Do you know the types of the parameters that are passed to the invocation of constructor 'Calculator'?

Q2) Can you recall one method/constructor that calls method 'Calculator.popOperand()'?
Typical Questions
What the Subjects were asked...

```java
public class Calculator {
    public Stack<StackItem> stack;

    public Calculator(double operand) {
        stack = new Stack<StackItem>();
        stack.push(new StackNumber(operand));
    }

    public double popOperand() {
        double result = peekOperand();
        stack.pop();
        return result;
    }

    ...
GUI
How the Subjects answered...
GUI

How the Subjects answered...
GUI
How the Subjects answered...
Results

Significance of DOI
Results
Significance of Recency and Frequency
Results

Significance of Recency and Frequency
Results

Significance of Recency and Frequency
Problems
Threads to Validity

- Number of Subjects
- Technical Problems with IDE
- Only Java/Eclipse watched
- "Nature" of Questions
- Only IDE monitored
Improvements
What should be Added to the DOI

**Authorship**
Leads to more knowledge about authored program elements

**Code Stability**
Fewer code changes lead to longer knowledge about program elements

**Activity**
Short-term kinds of activity, like debugging, have a skewing effect
Summary

▶ DOI is a **good Indicator** for structural code Knowledge

▶ **Improvements** are still possible

*Authorship, Activity, . . .*

Outlook

Results can be used to create

*better Expertise Recommendation tools*

Should be repeated with

*more Programmers, Languages*
Summary

▶ DOI is a good Indicator for structural code Knowledge

▶ Improvements are still possible
   Authorship, Activity, . . .

Outlook

▶ Results can be used to create
  better Expertise Recommendation tools

▶ Should be repeated with
  more Programmers, Languages
**Typical Questions**
What the Subjects were asked...

**Q1)** Can you recall the name of one class or interface that is directly extended, implemented by 'TYPE', or can you recall the name of one class that directly extends the type 'TYPE'?

**Q2)** Do you know the types of the parameters that are passed to the invocation of method/constructor 'METHOD'?

**Q3)** Do you know two methods that are called by method/constructor 'METHOD'?

**Q4)** Can you recall one method/constructor that calls method/constructor 'METHOD'?
## Improvements

What should be Added to the DOI

<table>
<thead>
<tr>
<th>Authorship</th>
<th>More knowledge about authored program elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorship Duration</td>
<td>More time spent creating an element $\Rightarrow$ more knowledge about the element.</td>
</tr>
<tr>
<td>Code Stability</td>
<td>Fewer code changes $\Rightarrow$ longer knowledge about it</td>
</tr>
<tr>
<td>Code Patterns</td>
<td>More code patterns $\Rightarrow$ easier to infer knowledge</td>
</tr>
<tr>
<td>Role</td>
<td>Important element $\Rightarrow$ more knowledge</td>
</tr>
<tr>
<td>Task Locality</td>
<td>More interaction with other programmers’ code $\Rightarrow$ less DOI-to-knowledge correspondence.</td>
</tr>
<tr>
<td>Activity</td>
<td>Short-term kinds of activity, like debugging, have skewing effect.</td>
</tr>
<tr>
<td>Work Experience</td>
<td>More experience of programmer $\Rightarrow$ better DOI-to-knowledge correspondence.</td>
</tr>
</tbody>
</table>
Mylyn
What it looks like...

http://www.eclipse.org/mylyn/