Did the availability of refactoring functionality of IDEs result in refactorings being performed more frequently?

Abstract

Restructuring existing code during software development is a useful activity but it's often complained to be laborious. This includes fairly simple operations like changing a method's name as well as complex ones like converting an embedded class to a top level class. To support these activities latest IDEs (Integrated Development Environments) provide corresponding functionality to automate these tasks.
The study presented here investigates whether the introduction of refactoring support in IDE changed the behaviour of the programmers, especially if refactoring is being preformed more frequently due to tool support.
During the course "Empirical Evaluation in Informatics" at Freie Universität Berlin a survey started in June 2004. Readers of some selected mailing lists and a few companies were asked to fill out an online form.
Only programming language Java supported by the environments Eclipse and IntelliJ IDEA have been considered.
91 participants responded. 22 of them reported raised usage of refactorings, mostly renaming, moving classes and methods, and changing the parameter list of methods. No correlation was found between programming experience and amount of refactoring activity.

1 Introduction

Restructuring of existing code is not only necessary for long-lived, maintenance-intensive projects, but should be everyday practice for a software developer. This is usually denoted as "Refactoring" which is defined in [1] as:

„Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations. Each transformation (called a

'refactoring') does little, but a sequence of transformations can produce a significant restructuring. Since each refactoring is small, it's less likely to go wrong. The system is also kept fully working after each small refactoring, reducing the chances that a system can get seriously broken during the restructuring.“

Opposed to its positive effect on the quality of the code, refactorings are labour-intensive and error-prone. I should better be supported by developer tools. Vendors of IDEs (Integrated Development Environment) therefore provide support for particular refactoring activities in recent times. These functionalities perform operations like moving methods or adding parameters automatically by simply choosing a menu entry.

The study at hand investigates the question whether the availability of refactoring functionality in IDEs resulted in refactorings being performed more frequently. A questionnaire as a link to a web-based form has been sent to some mailing lists (see Appendix A) and selected companies.

Since the IDEs differ significantly in their amount and usability of the refactoring support for different programming languages, the survey concentrated on programming language Java and IDEs Eclipse, and IntelliJ IDEA. This permitted a more homogeneous result.

2 Implementation

The questionnaire was designed as a web based form, see appendix C. Mails and postings were sent via e-mail to selected mailing lists (appendix A), newsgroups, and some companies, see appendix B. E-Mail as well as survey were written in English and German with the option to switch the language in the form. Some mailing lists and the companies received the German version.

The form was built using PHP, and Phorm [2]. It has been on line for 17 days, from 8th to 25th of June 2004.

3 Questionnaire

The questionnaire contained three parts.

- General questions about the subject under investigation
- Questions about specific refactoring activities
- Questions regarding the attendee

3.1 General questions about the subject under investigation

This part contained a question about the meaning of “Refactoring”, and four questions concerning the IDE and its usage by the participants.

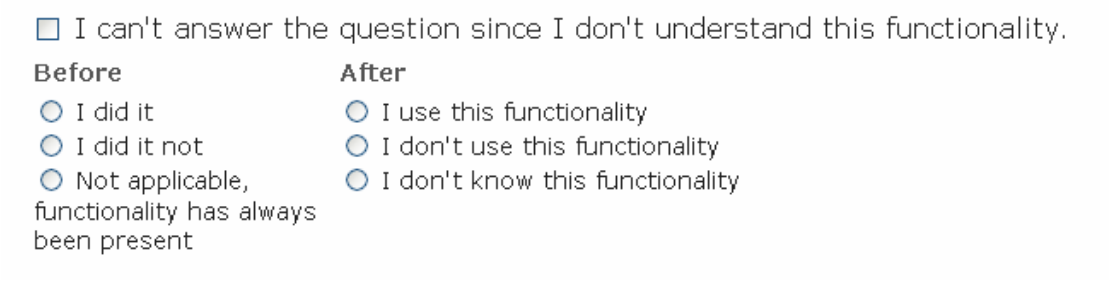
3.2 Questions about specific refactoring activities

This core part of the questionnaire contained 11 similar blocks of questions about following refactoring activities.

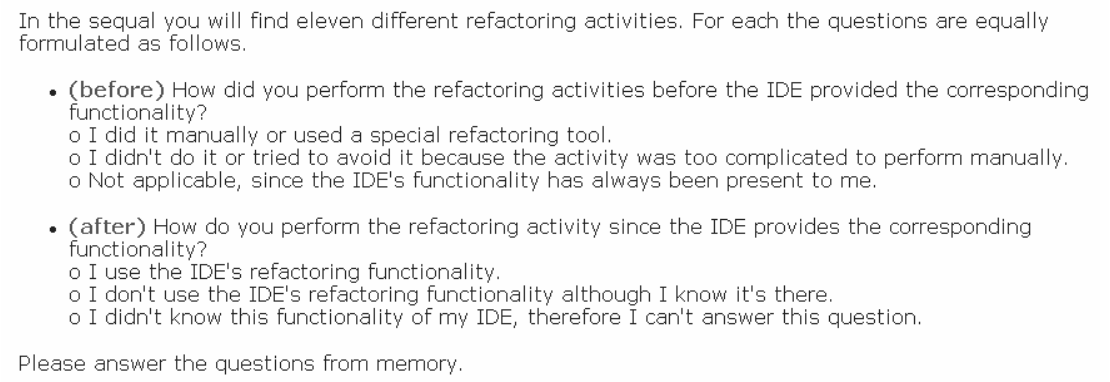
- Rename class, method, field (R)
- Move class (M)
- Add or remove parameters of a method (P)
- Convert anonymous class into named inner class (AI)
- Convert inner class into top-level class (IT)
- Move methods or fields of a class to sub-classes (D)
- Move methods or fields of a class to super-class (U)

- Create and use an interface on base of a class and its methods (EI)
- Extract method from code fragment (EM)
- Convert an expression to an additional parameter of a method (EP)
- Encapsulate field and create getter and setter methods (F)

Both Eclipse and IntelliJ IDEA provide corresponding functions in the latest versions. The menu entries were mentioned in the questionnaire. The questions for each of the activities have been the same.



To clarify the meaning of the choices an introduction was given.



The purpose was to identify following problems.

- Which refactoring activities haven't been performed before the IDE supported it?
- Which refactoring activities are actually used?
- Did the introduction of functions concerning refactoring in IDEs change the behaviour of programmers?

3.3 Questions regarding the attendee

Five questions targeted the experience of the participants, their job, employment years, number of programming languages (with applicable knowledge), general programming experience as well as Java knowledge.

4 Findings

4.1 Participants

94 answers were sent in, 91 of which were valid. Three forms have been sorted out because they were empty.
The valid forms were mostly complete, leaving just a few if any questions unanswered.
The survey took part from 8th to 25th June 2004. The input was spread over the days of the month as shown in Figure 4-1.

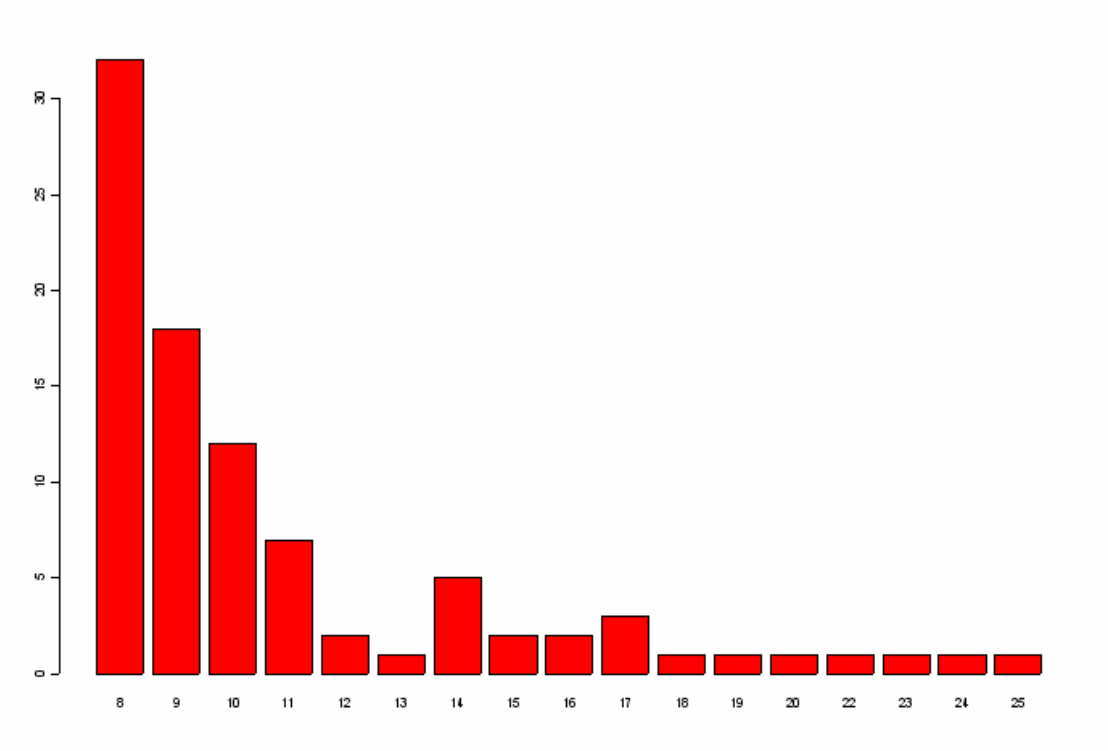


Figure 4-1: Number of answers per day

The main job of the attendees is distributed as shown in Figure 4-2.

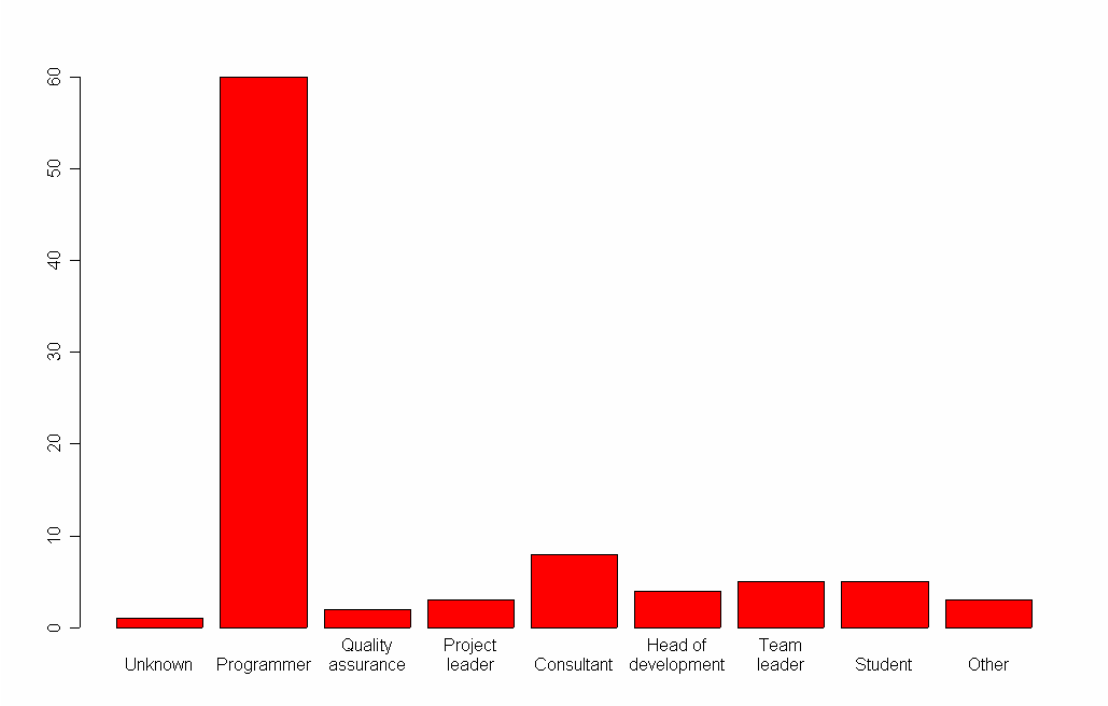


Figure 4-2: Jobs

The programming experience was surprisingly high, ranging from 2 to 35 years with a cluster at 20 years. The latter may be explained by the fact that long term experience in years has been rounded to the next multiply of ten. The mean is at around 15 years, the median at 14.7, see Figure 4-3.

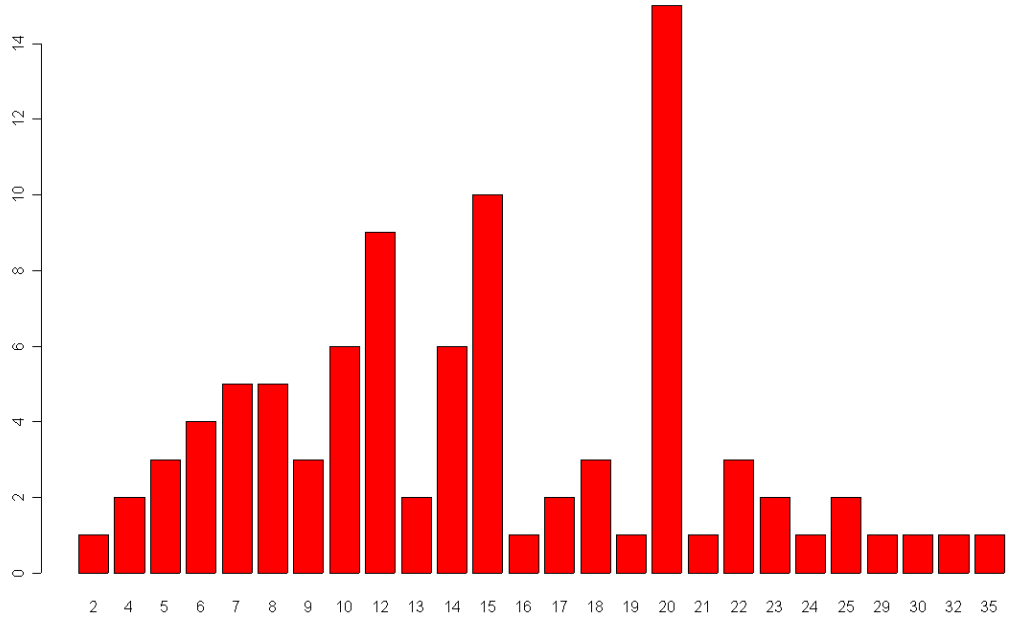


Figure 4-3: Programming experience

To get a better impression of the experience of the participants another survey's question was about the number of programming languages an attendee has "applicable skills" of. The mean was 4.6 with a median of 4. No correlation between programming experience and number of programming languages was observed (correlation coefficient 0.31).
The programming experience in Java was 5.2 years in the mean.
The question about which description best suited the participant's understanding of the term „refactoring" resulted in mainly two opinions:

Available Answer	Number of Choices
Change of software design	47
Modification of source code	43
Adding functionality	0
I don't know	0
- none -	1

As a summary, the attendees are predominantly experienced programmers. This may be due to the set of mailing list which were chosen for the survey. They were mainly specialists boards. Additionally, the term „refactoring" in the opening letter may attract experienced programmers mainly.

4.2 Distribution of IDE products

60 participants use Eclipse, 31 use IntelliJ IDEA, mainly the latest version (Eclipse 3.x: 39, IntelliJ IDEA 4.x: 26).
Most of the attendees (51) are using an IDE of the same vender longer than half a year. It seems that users stay with their chosen product line.
The quality of the refactoring functions of the IDE was important for choosing this specific product as depicted in Figure 4-4.

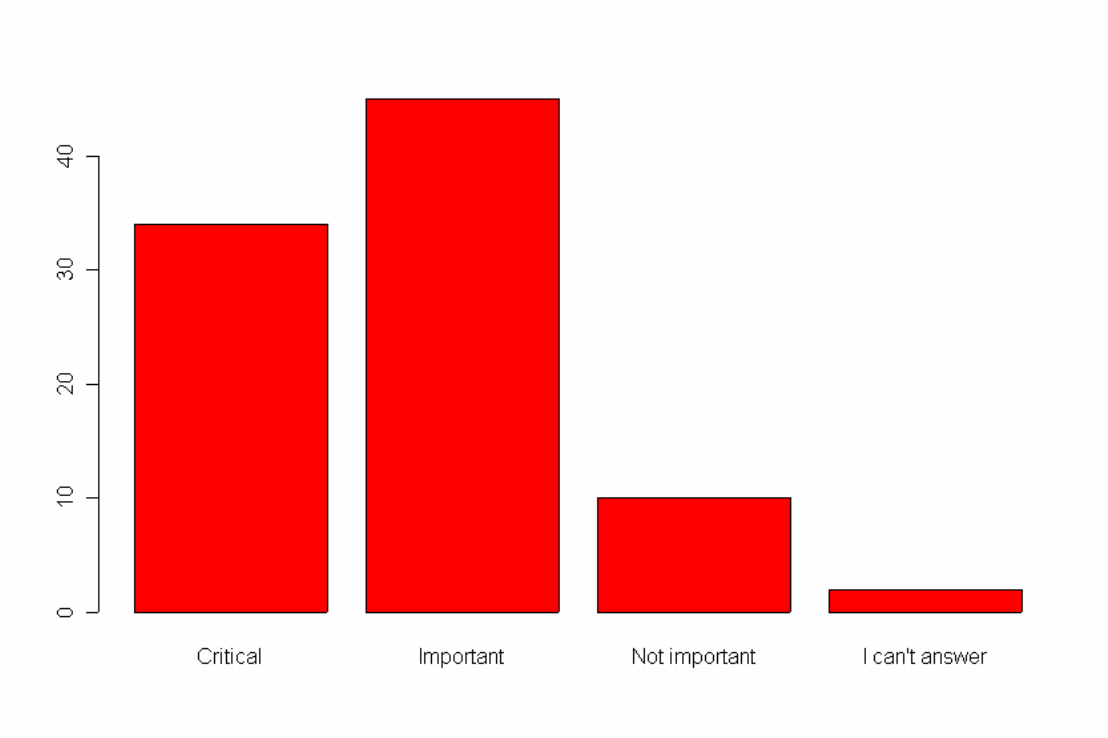


Figure 4-4: Refactoring functions as a criteria for choosing an IDE

4.3 Usage of refactoring functionality

Which of the eleven refactoring activities have been performed before the IDE supported them with specialized functionality?[1] For each participant the number of corresponding answers were counted over all eleven refactoring activities. In figure 4-5 the number of answers are specified horizontally, and vertically the number of participants which have this specific amount of answers. For example, three attendees used only one of the activities before their IDE supported this as a functionality whereas more then ten used all eleven before. From those three who never used any before, actually only one chose “I did it not” on all eleven activities. The other two always used the IDE for performing the activities, i.e. there is no “before” for them.

[1] An answer was only included if the IDE's functionality hasn’t always been present for the user anyway.

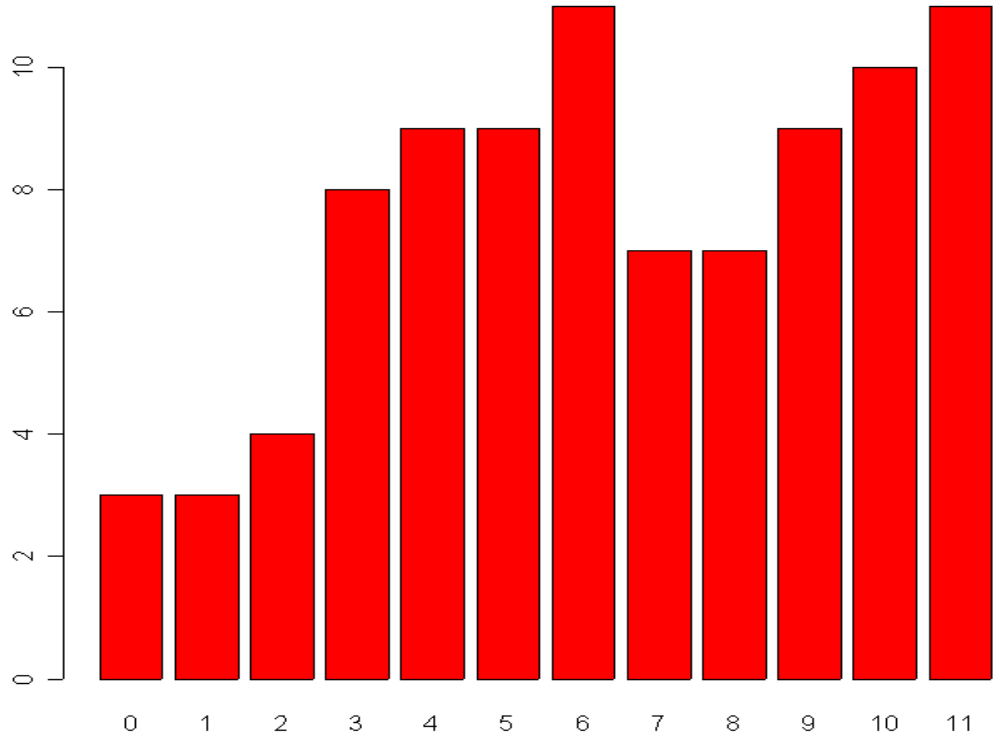


Figure 4-5: Number of refactoring activities prior to using the IDE

The refactoring activities mostly used before are

- (P): Add or remove parameters of a method (74)
- (R): Rename class, method, field (71)
- (EM): Extract method from code fragment (64)
- (M) Move class (63)

Following activities were only seldom used.

- (AI): Convert anonymous class into named inner class (28)
- (IT): Convert inner class into top-level (32)
- (EP): Create and use an interface on base of a class and its methods (37)

No correlation has been observed between programming experience and amount of usage of refactoring.

Examining which refactoring activities are performed regardless whether with or without the IDE or before or after the IDE was introduced, there are some obviously often and rarely used activities as shown in Figure 5-6. For each activity (denoted as presented in 3.2) the percentage of usage for all participants is drawn, i.e. renaming classes is an activity everybody does whereas converting an anonymous to an inner class is performed by only half of the attendees.

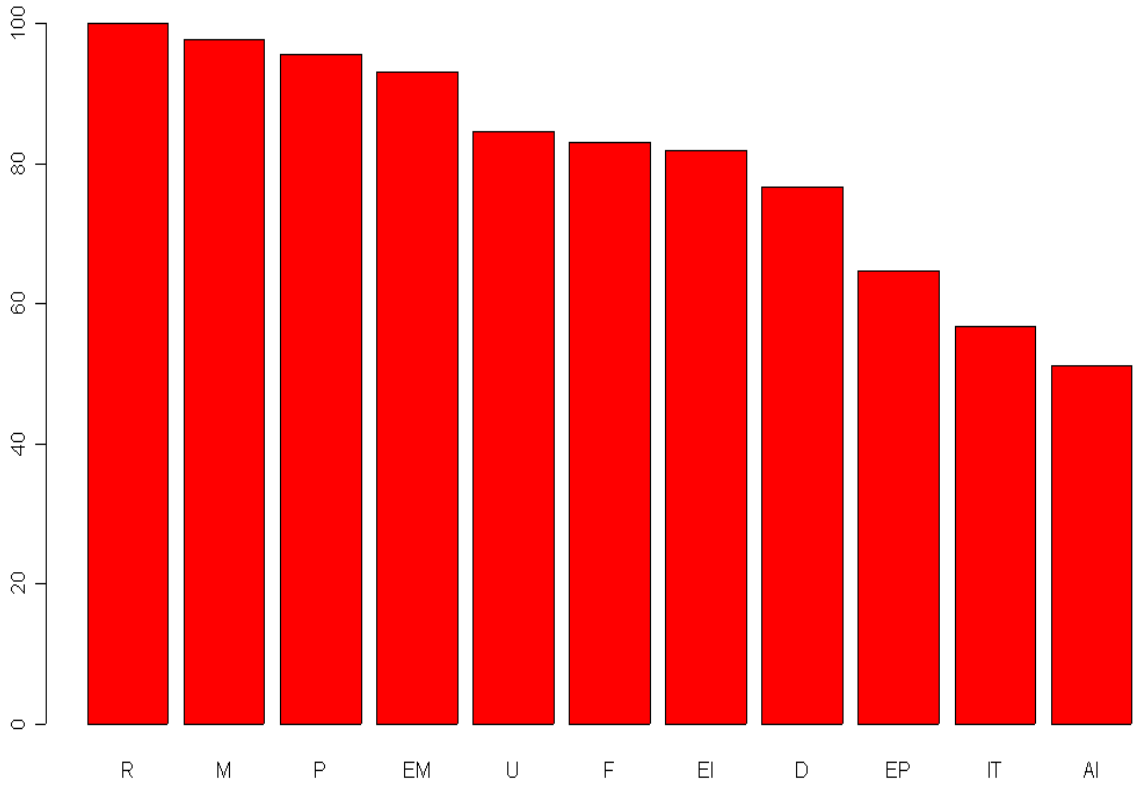


Figure 4-6: Usage of refactoring activities regardless of the means

See Figure 5-7 towards answering the main question of the survey: Did the availability of refactoring functionality of IDEs result in refactorings being performed more frequently? Horizontally specified is the number of answer combinations „I did it not“ in the “before” part and „I use this functionality” in the “after” part of each of the eleven refactoring blocks of the survey. Vertically drawn you see the number of attendees who have 0,1,2,… combinations of this kind. For example, for 40 participants the introduction of the IDE’s functionality didn’t change their usage of refactoring (either because they don’t use the IDE’s support or because they did it before anyway). One participant performs seven refactorings she/he didn’t perform before the IDE provided support for it.

If one defines „raised usage“ as performing 1/3rd of the refactoring activities *with* the IDE *but not* before (i.e. 4 refactorings of the 11), 22 participants reported raised usage. That’s about ¼ of all attendees. Some even perform more then five activities they didn’t perform before.

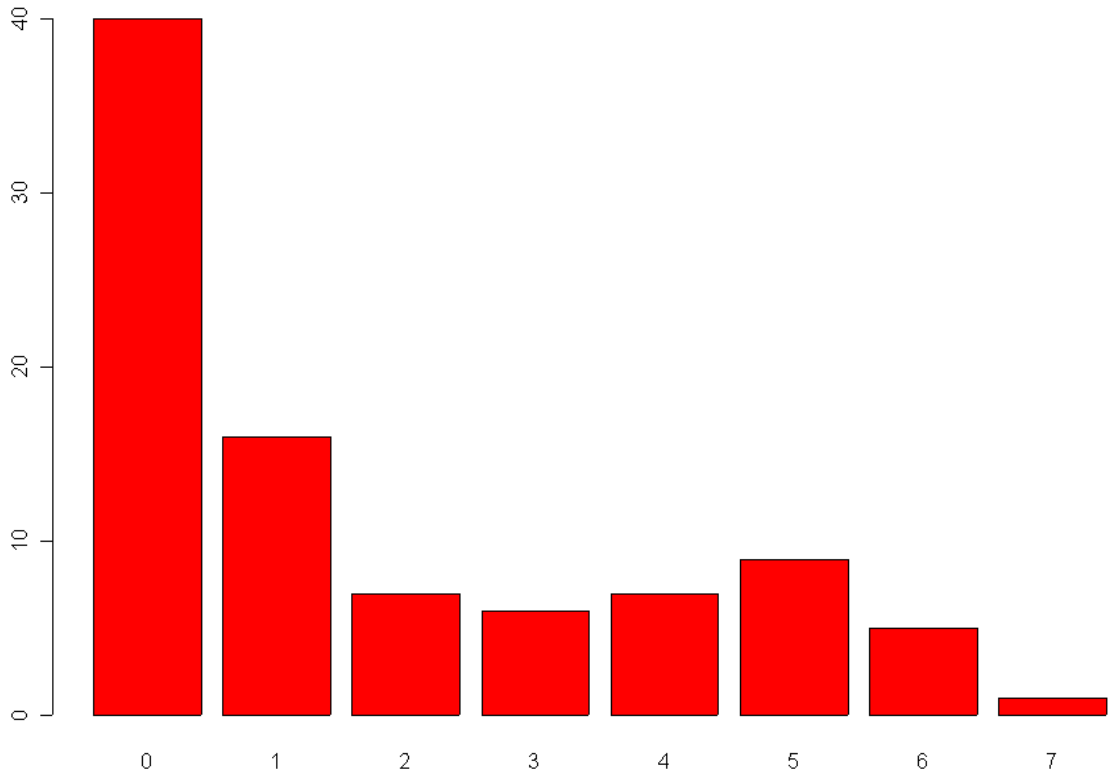


Figure 4-7: Raised refactoring usage

4.4 Summary and remaining questions

As depicted in figure 4-4, refactoring functionality is of high interest to users of IDEs. Yet the corresponding refactoring activities have been preformed by many of the participants before they used IDEs for this purpose. Only ¼ of the attendants seem to perform refactoring to a greater extend.
It would be interesting to know how far IDEs improved refactoring, especially

- How much has the effort doing refactoring been reduced using the IDE?
- Did the automatic support eliminated the injection of defects during refactoring?
- Did the programmer's attitude towards refactoring change?

5 References

[1] http://www.refactoring.com/
[2] http://www.phorm.com/

Appendix A: Mailing lists

- http://de.groups.yahoo.com/group/agile-ka/
- http://groups.google.de/groups?hl=de&lr=&ie=UTF-8&oe=UTF-8&group=comp.lang.java.softwaretools
- http://groups.google.de/groups?hl=de&lr=&ie=UTF-8&oe=UTF-8&group=comp.software-eng
- http://groups.google.de/groups?hl=de&lr=&ie=UTF-8&group=comp.lang.java.developer
- http://groups.yahoo.com/group/refactoring/
- http://www.jsurfer.de/modules.php?name=Forums&file=viewforum&f=1
- http://groups.google.com/groups?hl=de&lr=&safe=off&group=comp.lang.java.programmer
- http://groups.google.de/groups?hl=de&lr=&ie=UTF-8&group=comp.software.extreme-programming
- http://www.intellij.org/twiki/bin/view/Main/HowOftenDoYouRefactor

Appendix B: Invitation

Dear visitors of [mailing list],

the Institute of Computer Science of the "Freie Universitaet Berlin" is currently investigating the question

> "Did the availability of refactoring functionality of IDEs result in refactorings being performed more frequently?"

We would be pleased if you could participate in our survey and fill out (requiring about 15 minutes) the survey form available online at

http://www.inf.fu-berlin.de/inst/ag-se/teaching/survey/04-01/survey.php

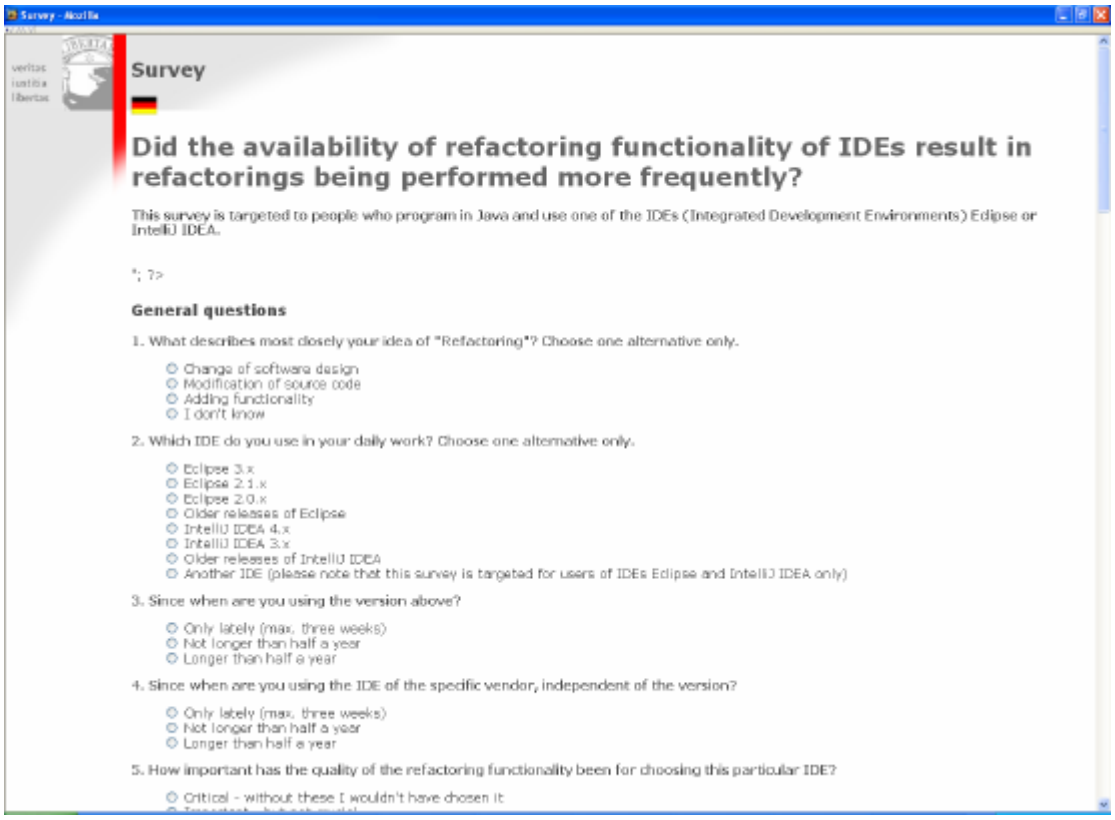
As a prerequisite, only users of the development environments (IDEs) IntelliJ IDEA or Eclipse for Java may attend.

In case you leave your e-mail address, we will send you the result of the survey. Your address will only be used for this purpose and will be deleted after completion of the study.

The survey will last until 22/6.

Many thanks for participating!

Appendix C: Questionnaire



Survey

Did the availability of refactoring functionality of IDEs result in refactorings being performed more frequently?

This survey is targeted to people who program in Java and use one of the IDEs (Integrated Development Environments) Eclipse or IntelliJ IDEA.

*; ?>

General questions

1. What describes most closely your idea of "Refactoring"? Choose one alternative only.

- ○ Change of software design
- ○ Modification of source code
- ○ Adding functionality
- ○ I don't know

2. Which IDE do you use in your daily work? Choose one alternative only.

- ○ Eclipse 3.x
- ○ Eclipse 2.1.x
- ○ Eclipse 2.0.x
- ○ Older releases of Eclipse
- ○ IntelliJ IDEA 4.x
- ○ IntelliJ IDEA 3.x
- ○ Older releases of IntelliJ IDEA
- ○ Another IDE (please note that this survey is targeted for users of IDEs Eclipse and IntelliJ IDEA only)

3. Since when are you using the version above?

- ○ Only lately (max. three weeks)
- ○ Not longer than half a year
- ○ Longer than half a year

4. Since when are you using the IDE of the specific vendor, independent of the version?

- ○ Only lately (max. three weeks)
- ○ Not longer than half a year
- ○ Longer than half a year

5. How important has the quality of the refactoring functionality been for choosing this particular IDE?

- ○ Critical - without these I wouldn't have chosen it

5. How important has the quality of the refactoring functionality been for choosing this particular IDE?

- ○ Critical - without these I wouldn't have chosen it
- ○ Important - but not crucial
- ○ Not important
- ○ I can't answer this question

Questions about specific refactoring activities

In the sequal you will find eleven different refactoring activities. For each the questions are equally formulated as follows.

- (before) How did you perform the refactoring activities before the IDE provided the corresponding functionality?
  - o I did it manually or used a special refactoring tool.
  - o I didn't do it or tried to avoid it because the activity was too complicated to perform manually.
  - o Not applicable, since the IDE's functionality has always been present to me.
- (after) How do you perform the refactoring activity since the IDE provides the corresponding functionality?
  - o I use the IDE's refactoring functionality.
  - o I don't use the IDE's refactoring functionality although I know it's there.
  - o I didn't know this functionality of my IDE, therefore I can't answer this question.

Please answer the questions from memory.

6. Rename class, method, or field

Menu entry in Eclipse: "Rename..."
Menu entry in IntelliJ IDEA: "Rename..."

☐ I can't answer the question since I don't understand this functionality.

Before	After
○ I did it	○ I use this functionality
○ I did it not	○ I don't use this functionality
○ Not applicable, functionality has always been present	○ I don't know this functionality

7. Move class

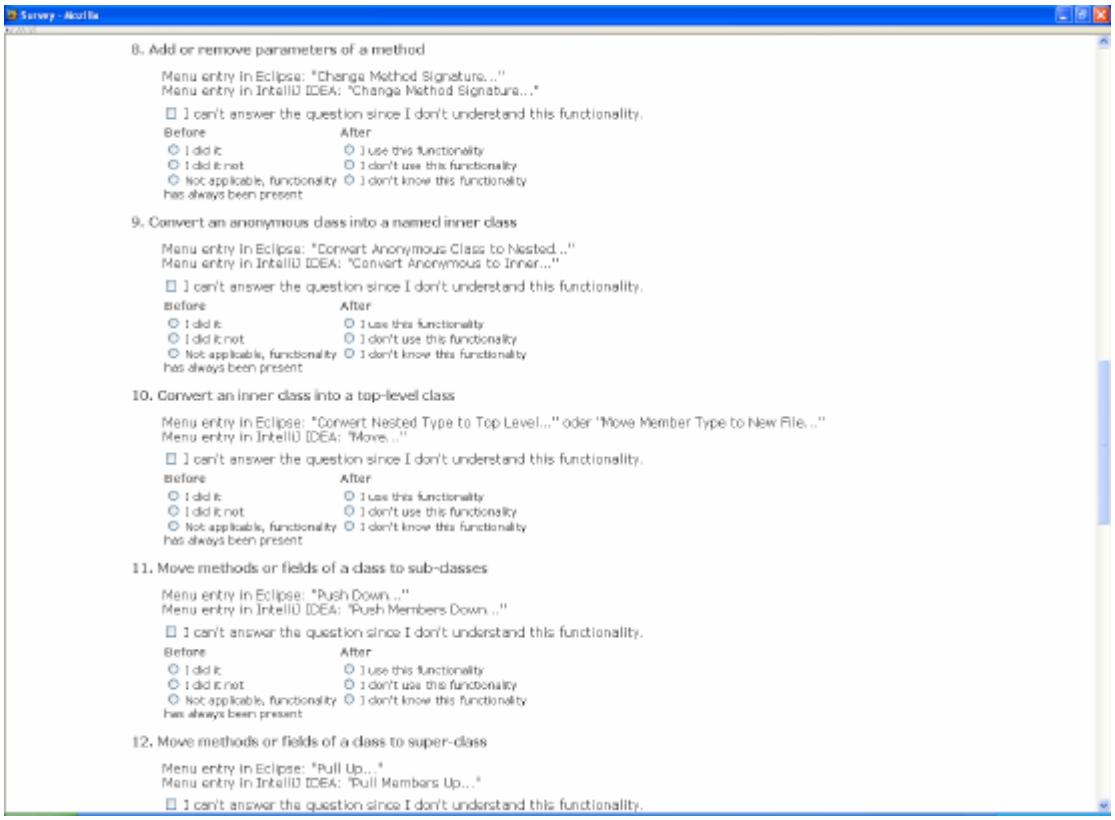
Menu entry in Eclipse: "Move..."
Menu entry in IntelliJ IDEA: "Move..."

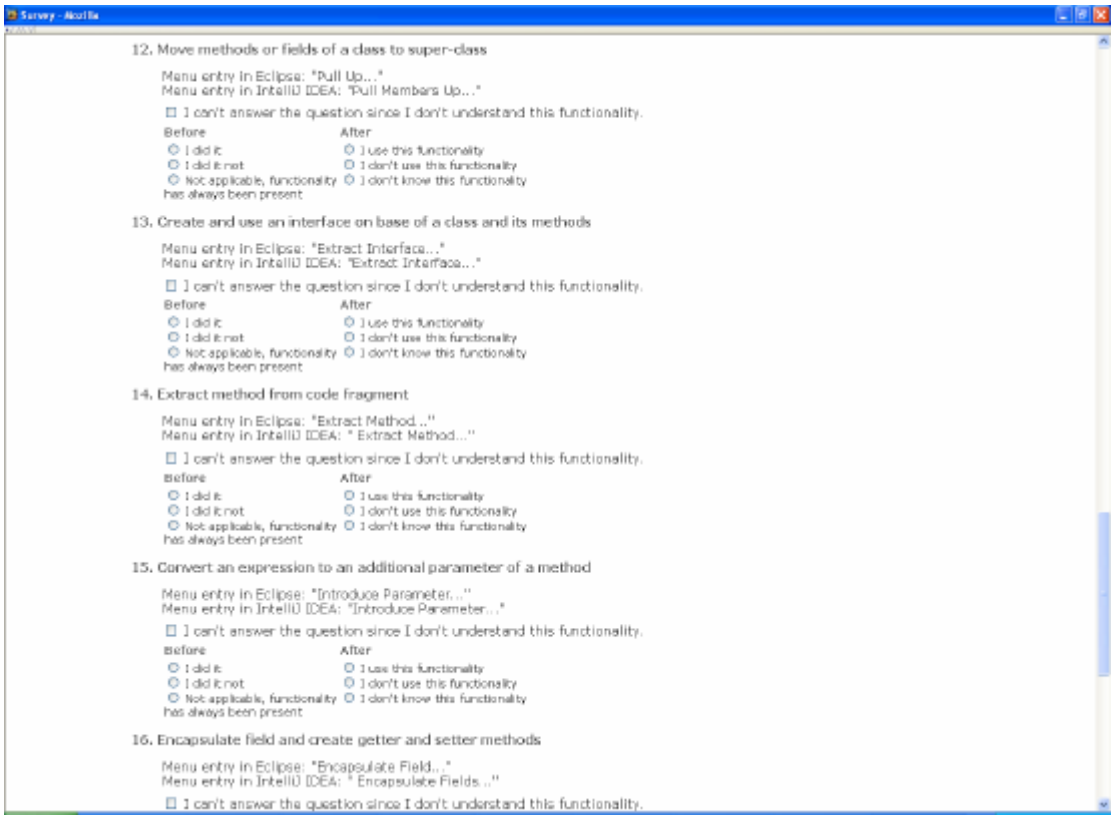
☐ I can't answer the question since I don't understand this functionality.

Before	After
○ I did it	○ I use this functionality
○ I did it not	○ I don't use this functionality
○ Not applicable, functionality has always been present	○ I don't know this functionality

8. Add or remove parameters of a method

Menu entry in Eclipse: "Change Method Signature..."
Menu entry in IntelliJ IDEA: "Change Method Signature..."


Survey - Mozilla
8. Add or remove parameters of a method
Menu entry in Eclipse: "Change Method Signature..."
Menu entry in IntelliJ IDEA: "Change Method Signature..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
9. Convert an anonymous class into a named inner class
Menu entry in Eclipse: "Convert Anonymous Class to Nested..."
Menu entry in IntelliJ IDEA: "Convert Anonymous to Inner..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
10. Convert an inner class into a top-level class
Menu entry in Eclipse: "Convert Nested Type to Top Level..." oder "Move Member Type to New File..."
Menu entry in IntelliJ IDEA: "Move..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
11. Move methods or fields of a class to sub-classes
Menu entry in Eclipse: "Push Down..."
Menu entry in IntelliJ IDEA: "Push Members Down..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
12. Move methods or fields of a class to super-class
Menu entry in Eclipse: "Pull Up..."
Menu entry in IntelliJ IDEA: "Pull Members Up..."
I can't answer the question since I don't understand this functionality.


Survey - Mozilla
12. Move methods or fields of a class to super-class
Menu entry in Eclipse: "Pull Up..."
Menu entry in IntelliJ IDEA: "Pull Members Up..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
13. Create and use an interface on base of a class and its methods
Menu entry in Eclipse: "Extract Interface..."
Menu entry in IntelliJ IDEA: "Extract Interface..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
14. Extract method from code fragment
Menu entry in Eclipse: "Extract Method..."
Menu entry in IntelliJ IDEA: " Extract Method..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
15. Convert an expression to an additional parameter of a method
Menu entry in Eclipse: "Introduce Parameter..."
Menu entry in IntelliJ IDEA: "Introduce Parameter..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
16. Encapsulate field and create getter and setter methods
Menu entry in Eclipse: "Encapsulate Field..."
Menu entry in IntelliJ IDEA: " Encapsulate Fields..."
I can't answer the question since I don't understand this functionality.


Survey - Mozilla
16. Encapsulate field and create getter and setter methods
Menu entry in Eclipse: "Encapsulate Field..."
Menu entry in IntelliJ IDEA: "Encapsulate Fields..."
I can't answer the question since I don't understand this functionality.
Before
I did it
I did it not
Not applicable, functionality has always been present
After
I use this functionality
I don't use this functionality
I don't know this functionality
Questions about you
17. My job is... (only choose your main activity)
Programmer
Quality assurance
Project leader
Consultant
Head of development
Team leader
I am a student
Other
18. I am working in the domain of software development since (in full years):
please select
19. I have applicable skills in following number of program languages:
please select
20. I am programming since (in full years):
please select
21. I am programming in Java since (in full years):
please select
If you would like to get a copy of the survey result, please leave your e-mail address:
I am finished. Send it off!