Verwendung, Nichtgebrauch und Missbrauch von automatischer Restrukturierung

Michael Kriese

Johann Wolfgang Goethe-Universität Frankfurt am Main

m.kriese@gmx.net

December 8, 2013

Übersicht

Grundlagen

Was ist Restrukturierung (engl. Refactoring)?

Ein Prozess, der ein Softwaresystem so verändert, dass das externe Verhalten nicht geändert wird, der Code aber eine bessere innere Struktur erhält. [Kerievsky, 2005]

Grundlagen

Was ist Restrukturierung (engl. Refactoring)?

Ein Prozess, der ein Softwaresystem so verändert, dass das externe Verhalten nicht geändert wird, der Code aber eine bessere innere Struktur erhält. [Kerievsky, 2005]

Warum ist Refactoring wichtig?

Um übersichtlicheren Quellcode zu schreiben und Fehler in bestehenden Programmen zu vermeiden. [B. Daum, 2005]

Grundlagen

Was ist Restrukturierung (engl. Refactoring)?

Ein Prozess, der ein Softwaresystem so verändert, dass das externe Verhalten nicht geändert wird, der Code aber eine bessere innere Struktur erhält. [Kerievsky, 2005]

Warum ist Refactoring wichtig?

Um übersichtlicheren Quellcode zu schreiben und Fehler in bestehenden Programmen zu vermeiden. [B. Daum, 2005]

Wieso wird es also nicht genutzt?

Die Versuchsreihe

Die Versuchsreihe

Unterteilung

- Verwendung
- Nicht Benutzung
- Missbrauch

von Refactoring

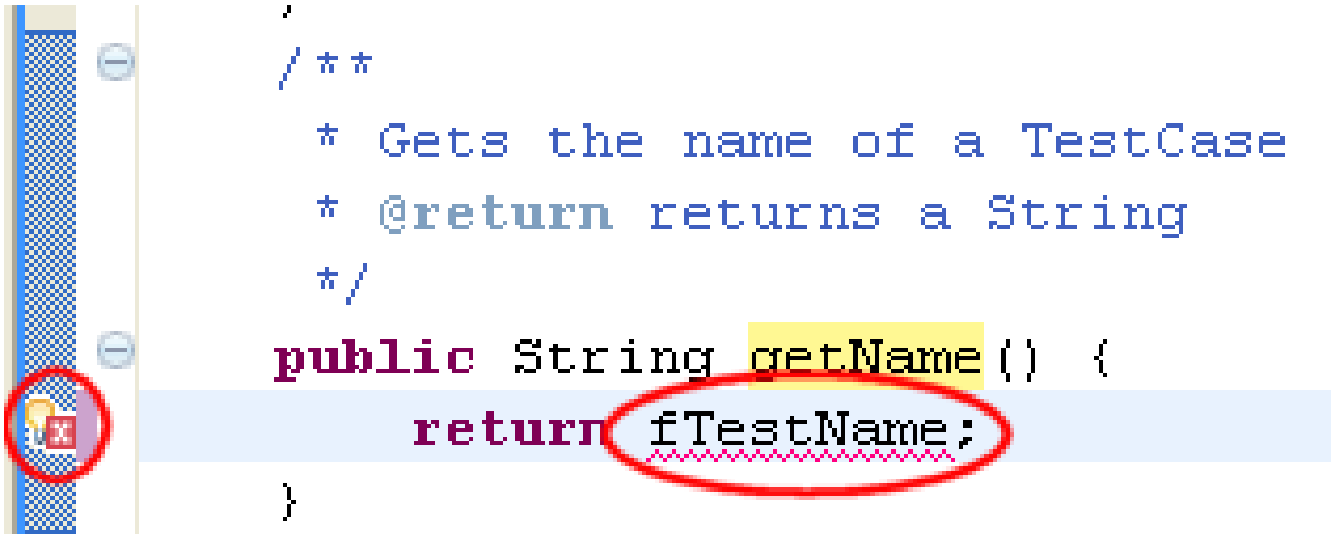
Die Versuchsreihe

Rahmenbedingungen

IDE:	Eclipse
Refactoring Werkzeuge:	Quick Assist, Quick Fix
Daten Sammeln:	CodingSpectator, CodingTracker, Befragungen

- 24 Programmierer
- 9 Befragungsteilnehmer
- ca. 1268 Programmier-Stunden

Quick Fix und Quick Assist



- Beide über die Tastenkombination Strg + 1 erreichbar.
- Quick Fix ist ein Assistent der beim Lösen von Problemen im Kompiliervorgang hilft, indem er Vorschläge für die Lösung der Kompilier-Warnungen gibt.
- Quick Assist ist nicht an die Kompilierprobleme gebunden und kann somit zum Umbenennen von Methoden, Variablen etc. verwendet werden.

(vgl. [M. Vakilian, N. Chen, S. Negara, B. A. Rajkumar, B. P. Bailey, R. E. Johnson, 2012])

Table 1

DATA ABOUT THE USAGE OF AUTOMATED REFACTORINGS FROM 26 PROGRAMMERS FOR ABOUT 1268 HOURS OVER THREE MONTHS. THE CS SUBSCRIPT INDICATES THE DATA CAPTURED USING CODINGSPECTATOR, AND THE CT SUBSCRIPT INDICATES THE DATA CAPTURED USING CODINGTRACKER. IN THE COMPLEXITY COLUMN, S = SIMPLE, M = MODERATE, AND C = COMPLEX. CONFIG_{CS} IS THE AVERAGE CONFIGURATION TIME (SECONDS) OF 788 REFACTORINGS. LINES_{CT} AND FILES_{CT} ARE THE AVERAGE NUMBERS OF AFFECTED LINES AND FILES COMPUTED USING THE DATA AVAILABLE FOR 93% OF THE PERFORMED REFACTORINGS CAPTURED BY CODINGTRACKER. $\text{PR}(\text{P} \mid \text{W})_{CS} = \text{PR}(\text{PERFORMED} \mid \text{WARNING})$ AND $\text{PR}(\text{P} \mid \text{E})_{CS} = \text{PR}(\text{PERFORMED} \mid \text{ERROR})$. THE SYMBOL "-" INDICATES AN UNKNOWN OR UNDEFINED VALUE. PERFORMED_{CS} IS LESS THAN PERFORMED_{CT} FOR CHANGE METHOD SIGNATURE BECAUSE CODINGSPECTATOR DID NOT SUPPORT THIS REFACTORING FROM THE BEGINNING OF THE STUDY.

Automated Refactoring	Complexity	Performed$_{CS}$	Performed$_{CT}$	Canceled$_{CS}$	Warning$_{CS}$	Error$_{CS}$	Fatal Error$_{CS}$	Quick Assist$_{CS}$	Preview$_{CS}$	Config$_{CS}$ (sec)	Lines$_{CT}$	Files$_{CT}$	Pr(P | W)$_{CS}$	Pr(P | E)$_{CS}$
Change Method Signature	C	45	49	8	1	9	0	-	0	8.1	7.48	2.31	1.00	1.00
Convert Anonymous Class to Nested	S	-	3	-	-	-	-	-	-	-	35.00	1.00	-	-
Convert Local Variable to Field	S	97	97	1	0	0	1	83	0	5.6	3.65	1.00	-	-
Encapsulate Field	M	-	225	-	-	-	-	-	-	-	10.44	1.10	-	-
Extract Class	C	-	15	-	-	-	-	-	-	-	120.75	2.25	-	-
Extract Constant	S	29	29	0	0	1	0	26	0	6.9	3.72	1.00	-	1.00
Extract Interface	M	2	2	4	2	0	0	-	1	25.6	36.00	1.50	0.00	-
Extract Local Variable	S	606	606	14	6	8	0	475	0	3.3	2.55	1.00	1.00	0.75
Extract Method	M	186	186	30	0	13	0	62	5	12.2	21.21	1.00	-	0.54
Extract Superclass	C	0	0	3	1	0	0	-	0	3.0	-	-	0.00	-
Generalize Declared Type	M	-	0	-	-	-	-	-	-	-	-	-	-	-
Infer Generic Type Arguments	C	-	7	-	-	-	-	-	-	-	1.29	0.57	-	-
Inline Constant	S	38	38	0	0	0	0	-	0	0.5	1.00	1.00	-	-
Inline Local Variable	S	182	182	1	0	0	0	73	0	0.4	3.26	1.00	-	-
Inline Method	S	63	63	8	0	3	1	-	0	1.5	9.97	1.13	-	0.67
Introduce Factory	M	-	0	-	-	-	-	-	-	-	-	-	-	-
Introduce Indirection	M	-	2	-	-	-	-	-	-	-	20.50	4.50	-	-
Introduce Parameter	C	-	46	-	-	-	-	-	-	-	4.74	1.52	-	-
Introduce Parameter Object	C	-	0	-	-	-	-	-	-	-	-	-	-	-
Move	C	147	147	8	0	3	3	-	10	5.5	12.01	2.19	-	1.00
Move Method	C	0	0	10	3	0	0	-	3	16.5	-	-	0.00	-
Move Static Member	M	6	6	1	1	0	0	-	0	13.0	45.20	3.20	1.00	-
Move Type to New File	S	-	7	-	-	-	-	-	-	-	54.50	1.00	-	-
Pull Up	C	9	9	0	5	0	0	-	1	8.9	11.78	2.89	1.00	-
Push Down	C	8	8	3	2	3	0	-	9	39.0	32.25	12.75	0.50	0.33
Rename Class	M	276	276	37	41	16	5	20	5	8.9	12.13	3.06	0.93	0.62
Rename Enumeration Constant	S	3	3	0	2	0	0	3	0	-	6.00	4.00	1.00	-
Rename Field	M	125	125	9	16	3	6	6	2	2.8	4.52	1.47	0.94	0.67
Rename Local Variable	S	465	465	14	4	16	6	20	0	2.3	3.37	1.00	0.50	0.62
Rename Method	M	260	260	17	33	16	9	15	0	7.4	3.45	2.20	0.94	0.69
Rename Package	M	12	12	0	4	0	0	0	0	6.8	4.67	2.75	1.00	-
Rename Type Parameter	S	6	6	0	0	0	0	0	0	-	2.17	1.00	-	-
Use Supertype Where Possible	C	0	0	7	0	0	0	-	1	5.6	-	-	-	-
		2565	2874	175	121	91	31	783	37	6.30	6.71	1.47	0.88	0.68
		Total Counts								Weighted Averages			Overall Pr	

Verwendung

Verwendung

- In dieser Studie wurden die ersten quantitativen Daten zur Nutzung von Quick Assist gesammelt.

Verwendung

- In dieser Studie wurden die ersten quantitativen Daten zur Nutzung von Quick Assist gesammelt.
- Programmierer stimmen sehr schnell dem automatischen Refactoring zu.
- 35 % der Zeit Quick Assist zum Ausführen von Refactoring.
- Alle Befragten kannten das Tool Quick Fix.
- Drei der Befragten war das Quick Assist Tool nicht bekannt.

Verwendung

Ergebnis:

- Würde es Programme geben, welche code smells finden und automatisches Refactoring als Lösung für das Problem anbieten würden, könnte dies die Programmierer dazu bewegen, öfters Refactoring anzuwenden.
- code smells = Quellcode, welcher schlecht lesbar sowie wartbar ist.

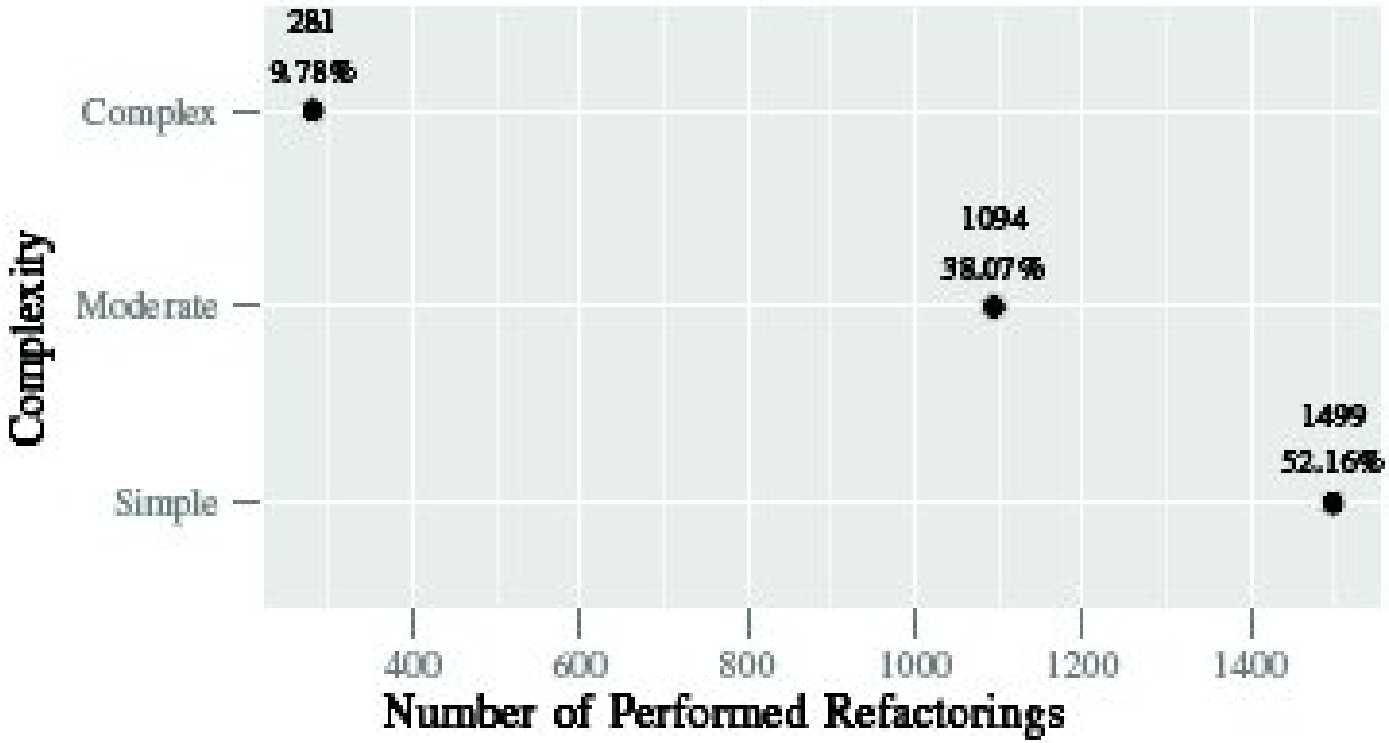
Nicht Benutzung

Nicht Benutzung

- Murphy-Hill hat bei einem Test der Versionskontrolle und Refactoring history festgestellt, dass 90 % der Refactorings von den Programmierern selbst durchgeführt werden.
  [G. C. Murphy, M. Kersten, and L. Findlater, 2006]
- In den Interviews sollte erstmals ein qualitativer Beweis erbracht werden, warum automatisches Refaktoring nicht genutzt wird.
  - Notwenigkeit
  - Sensibilisierung
  - Bezeichnung
  - Vertrauen (weniger wichtig)
  - Berechenbarkeit
  - Konfigurierung

Berechenbarkeit

- komplex
- mittel
- einfach



Berechenbarkeit

- Prinzipiell können große automatische Refactorings viele Zeilen Code in mehreren Dateien verändern.
- 82 % nur maximal 6 Zeilen
- 84 % nur eine Datei
- Grund: Überwältigung, da so viel auf einmal am Code verändert wird

Bezeichnung

- Befragungsteilnehmer sollten anhand der Namen erklären, was denn die einzelnen automatischen Refactorings machen.
- Manche Begriffe zu technisch, schwer verständlich oder einfach missverständlich.
- Sie konnten nicht erklären, was sie machen und konnten es auch nicht korrekt erraten.

Ergebnis:

- Benennung der Werkzeuge eindeutiger machen
- Sozial Media Anbindungen
- Motivation: die Belastung der Menschen sowie die Fehler in großen Projekten kann reduziert werden

Missbrauch

Missbrauch

- Erste quantitativen und qualitativen Ergebnisse über die Nutzung der unsicheren Restrukturierungen bei Programmierern.

Missbrauch

- Erste quantitativen und qualitativen Ergebnisse über die Nutzung der unsicheren Restrukturierungen bei Programmierern.
- Ein Missbrauch von Automation bedeutet laut Definition, wenn sich Personen auf eine Automation zu sehr verlassen, statt es selbst zu machen. [R. Parasuraman, V. Riley, 1997]
- Beispiel: Eastern Fluges 401 in den Everglades (Florida)

Missbrauch

- Erste quantitativen und qualitativen Ergebnisse über die Nutzung der unsicheren Restrukturierungen bei Programmierern.
- Ein Missbrauch von Automation bedeutet laut Definition, wenn sich Personen auf eine Automation zu sehr verlassen, statt es selbst zu machen. [R. Parasuraman, V. Riley, 1997]
- Beispiel: Eastern Fluges 401 in den Everglades (Florida)
- Wenn eine Voraussetzung nicht erfüllt wird, gibt das Refactoring Tool eine Warnung:
  - Information (information)
  - Warnung (warning)
  - Fehler (error)
  - schwerwiegender Fehler (fatal error)
- Wenn es eine Warnung gibt ist das Verhalten des Quellcodes nicht mehr garantiert.

Missbrauch

- Es ist mit Aufwand verbunden ein Refactoring abzubrechen und neu zu konfigurieren.
- Die Befragten behaupteten, dass sie sich der Grenzen des Refactorings Tool bewusst sind und die Fehler, welche durch das Tool entstanden sind, schnell finden und einfach beheben können.
- Ein hohes Vertrauen in die Refactoring Tools gepaart mit wenig Vertrauen in die eigenen Programmierfähigkeiten, führen ebenfalls zu einem Missbrauch.

Ergebnis

- Anzahl der Warnungen zu reduzieren, damit sie auch Beachtung vom Programmierer erhalten.
- Fehlermeldungen verständlicher gestalten.

Fazit

Quellen

Joshua Kerievsky (2005),
Refactoring to patterns,
Addison-Wesley,

Berthold Daum (2005),
Java-Entwicklung mit Eclipse 3.1,
dpunkt.verlag GmbH,

M. Vakilian, N. Chen, S. Negara, B. A. Rajkumar, B. P. Bailey, R. E. Johnson, (2012)
Use, Disuse, and Misuse of Automated Refactorings,
University of Illinois at Urbana-Champaign,

R. Parasuraman, V. Riley (1997)
Humans and Automation,
http://stuff.mit.edu/afs/athena/course/16/16.459/www/parasuraman.pdf

G. C. Murphy, M. Kersten, and L. Findlater (2006)
How Are Java Software Developers Using the Eclipse IDE?
IEEE Software 2006.

Fragen???

Fragen???

Vielen Dank
für Ihre Aufmerksamkeit