Software Technologie
Refactoring
Thema 1.1
Nutzungsverhalten der Entwickler bzgl. des Refactoring

Nguyen Hung

Überblick

- Refactoring
  - Ziele des Refactoring
  - Refactoring bei Entwicklern
  - Nutzungsverhalten Fragen
- Forschungsvorgehensweise
  - Warum ist es wichtig Refactoring in der Praxis zu erforschen?
  - Allgemeine Forschungsvorgehensweise
  - Bisherige Forschungsvorgehensweise
  - Verbesserte Forschungsvorgehensweise
- 7 Forschungsfragen
  - Nutzungsverhalten und was können wir daraus lernen?

Refactoring

- Disziplin des Refactoring ist die Umstrukturierung des Codes, ohne dabei die funktionale Eigenschaft des Programm zu ändern.

Ziele des Refactoring

- Software Qulität verbessern
- Effizienz bei Codierung

Vorgehensmodelle mit vielfachen Iterationen fordert in der Software Entwicklung viel Refactoring.

Automatisierte Refactoring ist Schlüssel Technologie zur Erfolg.

Refactoring bei Entwicklern

- Entwickler nutzen Refactoring bei Codierung
  - Umbenennen von alle arten von Variablen.
  - Umbenennen von Funktionen oder Methoden
  - Umwandlung lokale Variablen in Propertie Felder

Refactoring bei Entwicklern

- Entwickler nutzen enorm Refactoring in der Softwareentwicklung
  - Manuelle Umstrukturierung
  - Automatisiert umstrukturieren durch Refactoring-Tool

Nutzungsverhalten

Wie werden Refactoring in der Praxis verwendet?

Welche Nutzungsverhalten haben Entwickler zu Refactoring?

Was können wir daraus für Refactoringssysteme oder Tools lernen?

Warum ist es wichtig Refactoring in der Praxis erforschen?

- Das Verständnis Refactoring in der Praxis
- Software Tools Systeme
  - Gestaltung der Refactoring-Tool
  - Nutzungsverhalten der Entwickler zu Refactoring werden in der Refactoring-Tool abgestimmt.
  - Vereinfachte Nutzung

- Unterstützt Entwickler
  - Lesbarkeit, Wartbarkeit
  - Effizientes Codierung

Software Qualität verbessern

Allgemeine Forschungsvorgehensweise

- Beobachtung
  - Nutzungsverhalten im Labor
  - Natürlichen Umgebung der Entwickler
- Daten manuelle und automatisierte Refactoring
  - Code Snapshot vor Refactoring
  - Code Snapshot nach Refactoring
- Sammlung der Daten
  - Basieren auf Vergleiche der Snapshot vor und nach der Refactoring in Version Control System (VCS) ergeben sich notwendigen Daten für die Auswertung.
    - Häufigkeit der Nutzung
    - Entwicklergruppe und genutzten Refactoring
- Statistik Auswertung

Bisherige Forschung

- Stützen die Auswertung auf Resultate von Daten
  - Basieren Snapshot Vergleich
- Die Daten sind nicht vollständig und genau
  - Viele Refactoring erreichen nicht VCS
    - Aus diesen Paper geht hervor, dass 37% der geänderten Code nicht VCS erreichen.
    - Mehrfach angewendete Refactoring überschreiben gegenseitig.
    - Manuelle und automatisierte Refactoring kann vereinzelt überlappen.
    - Folge der angewendeten Refactoring kann nicht erkannt werden.

Verbesserte Vorgehensweise

- Datensammlung
  - In natürlichen Umgebung werden alle 10 unten genannten Refactoring während der Arbeit kontinuierlich erkannt und festgehalten.

Scope	Refactoring
API-level	Encapsulate Field Rename Class Rename Field Rename Method
Partially local	Convert Local Variable to Field Extract Constant Extract Method
Completely local	Extract Local Variable Inline Local Variable Rename Local Variable

Welche Vorteile ergeben sich?

- Lückenlose Aufnahme alle angewendeten Refactoring
- Dabei werden geänderte, hinzugefügte oder gelöschte Codefragmente als Knoten in ein Abstract Syntax Tree (AST) Baum abgelegt.
- Reihenfolge der angewendete Refactoring kann nachvollzogen werden

Datenumfang

- Gesammelten Daten stammen
  - 23 Entwickler in ihren natürlichen Umgebung
    - 10 professional Entwickler
    - 13 Hochschule Absolvent Student
  - Gesamt Zeit 1520 Stunden
  - 5371 verwendete Refactorings

Metric	Group							
	Aggregated	Affiliation		Tool usage (hours)		Programming experience (years)		
		Students	Professionals	≤ 50	> 50	< 5	$5 - 10$	> 10
Participants	23	13	10	13	10	5	11	6
Usage time, hours	1,520	1,048	471	367	1,152	269	775	458
Mean	66	81	47	28	115	54	70	76
STDEV	52	54	44	16	38	46	52	62

7 Forschungsfragen

- Die Antworten der 7 Forschungsfragen sollen uns einen vollständigeres und genaueres Bild zu der Nutzungsverhalten der Entwickler zu Refactoring geben. Wobei die siebte für Thema 1.1 nicht von relevant ist.

1. Welche Nutzungsanteil hat manuelle und automatisierte Refactoring?
2. Was sind die beliebtesten automatisierte und manuelle Refactoring?
3. Wie oft bevorzugt der Entwickler zu manuellen trotz Wissen über automatisierte Refactoring?

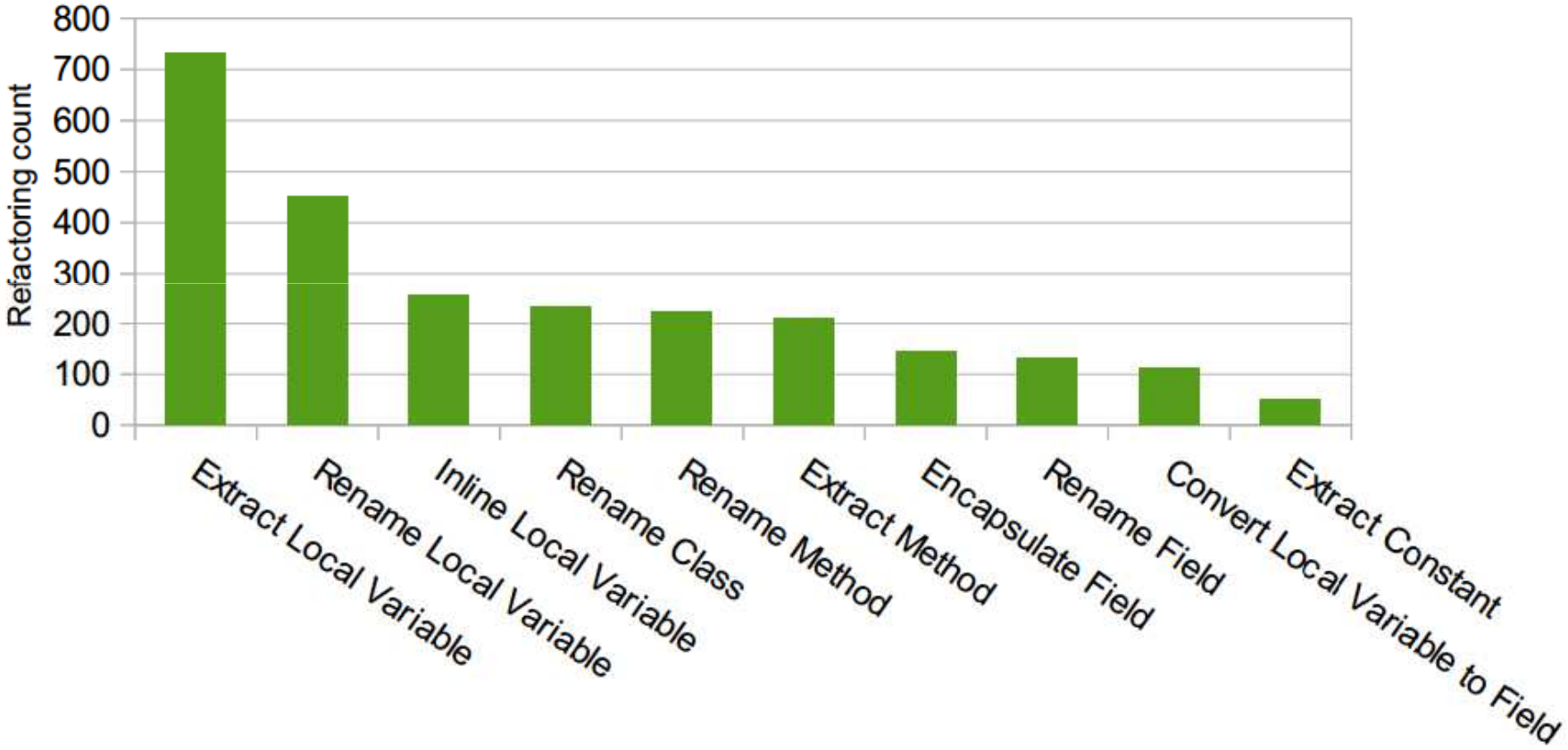
7 Forschungsfragen

4. Entscheidet die aufgewendete Nutzungszeit der Entwickler für manuelle oder automatisierte Refactoring?
5. Entscheidet die Aufwendungsgröße die Nutzung des automatisierte Refactoring?
6. Wie viele Refactoring wurde in Gruppe, also nacheinander verwendet?
7. Wie viele Refactoring erreichen nicht VCS?

RQ1 Nutzungsanteil manuelle vs automatisierte Refactoring?

- Mehr als die Hälfte der Refactoring wurden manuell durchgeführt
  - 11% manuelle Refactoring mehr als automatisierte Refactoring.
- Überwiegend manuell durchgeführte Refactoring
  - Convert Local Variable to Field
  - Extract Method
  - Rename Field
  - Rename Local Variable
  - Rename Method

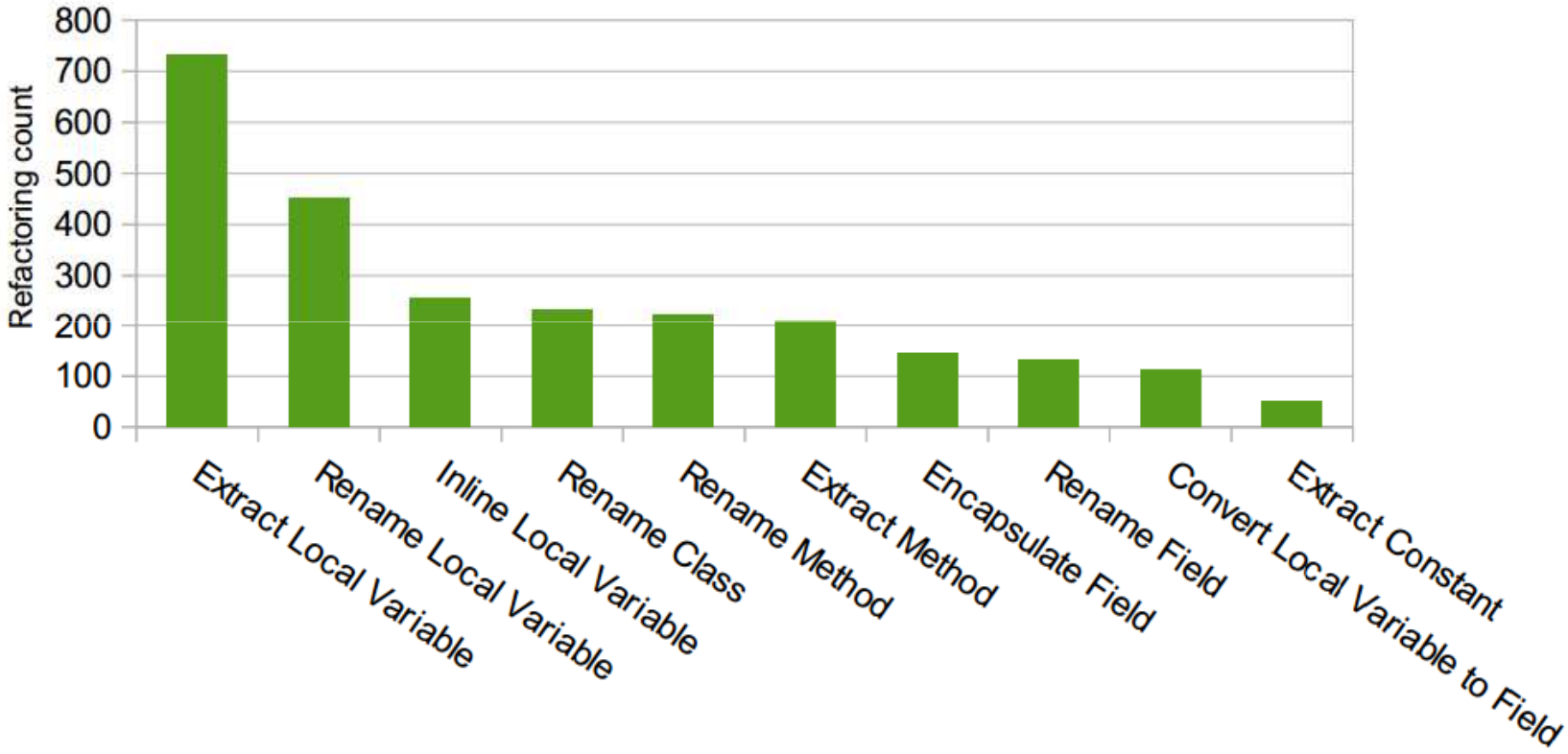
RQ1 Nutzungsanteil manuelle vs automatisierte Refactoring?



RQ 1 Nutzungsanteil manuelle vs automatisierte Refactoring?

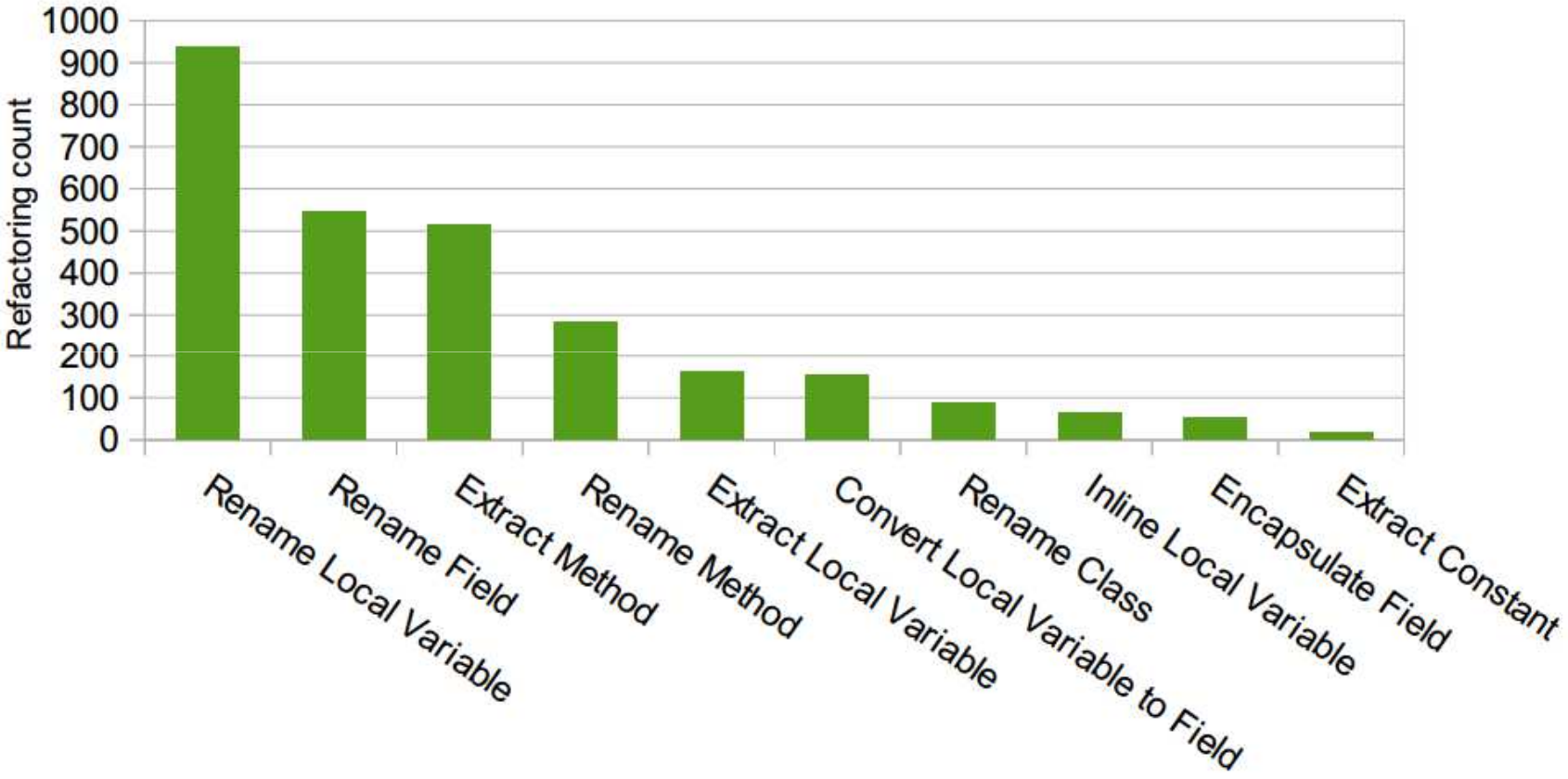
Teilnehmer Gruppe	Manuell über automatisierte	Mögliche Grund
< 5 Jahre Ex	28% mehr manuelle	Erhöhtes Wissen
5 – 10 Jahre Ex	12.1% Mehr automatisierte	Volles Wissen
Studenten	Kaum Nutzung automatisierte	Tools unwissend
> 10 Jahre Ex	Ca 1% wenige automatisierte	Volles Wissen. Gewöhnung an Manuelle, weil es vorher keine automatisierte gab.

RQ2 Die beliebtesten automatisierte und manuelle Refactoring?



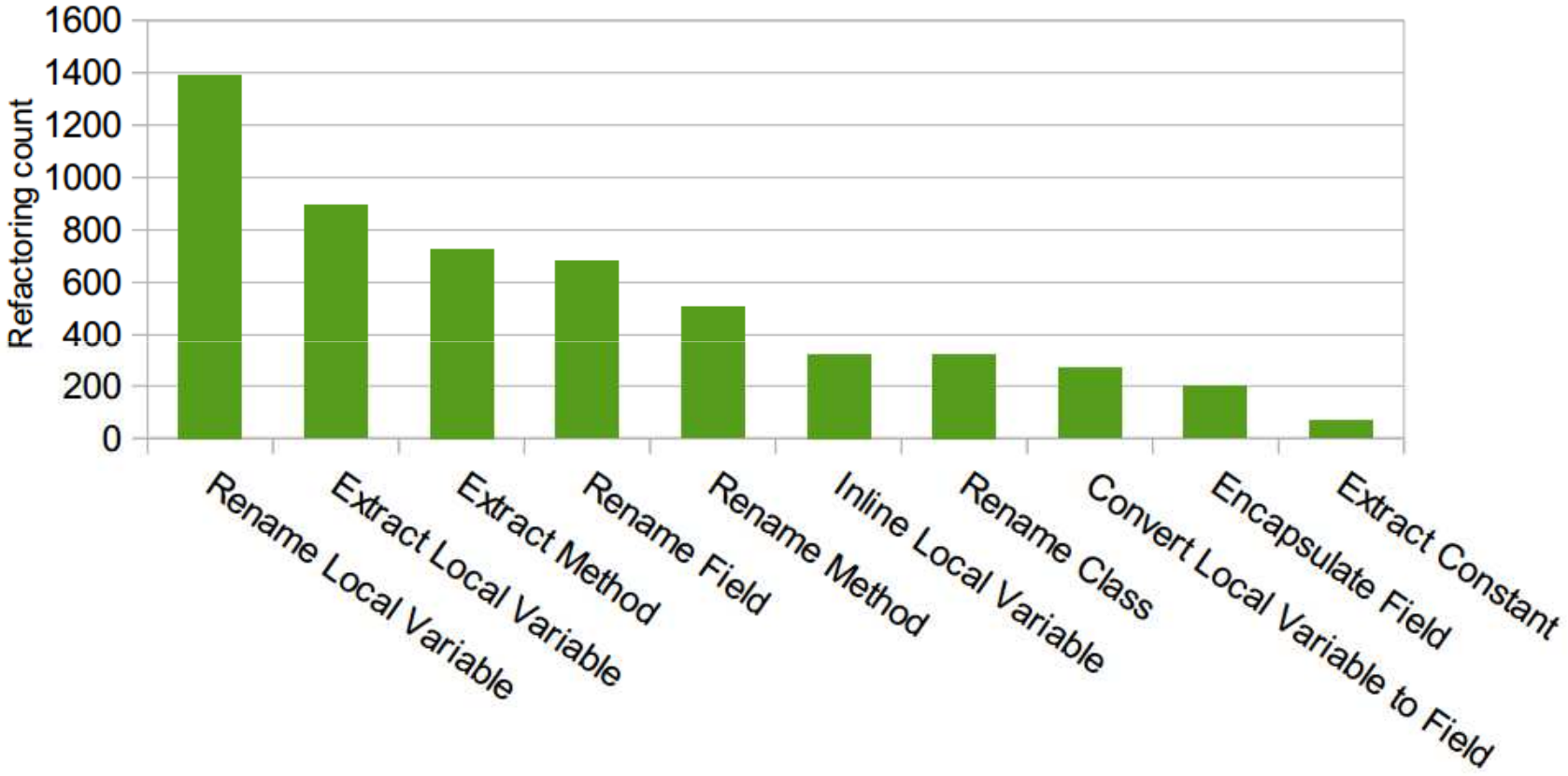
Beliebtheit automatisierte Refactoring

RQ2 Die beliebtesten automatisierte und manuelle Refactoring?



Beliebtheit manuelle Refactoring

RQ2 Die beliebtesten automatisierte und manuelle Refactoring?



Beliebtheit Refactoring im Allgemeinen

RQ2 Die beliebtesten automatisierte Refactoring

- Die Ergebnisse der Popularität bestätigen überwiegend mit der bisherigen Erkenntnissen von anderen Forschungen. Bis auf
  - Inline Local Variable
    - Von 7-te auf 3-te Rang
  - Encapsulate Field
    - Von 5-te auf 7-te Rang
- Der Gemeinsamen Nenner unter den top 5 der manuellen und automatisierte Refactoring mit jedoch unterschiedlichen Rang untereinander sind
  - Rename Local Variable
  - Rename Method
  - Extract Local Variable

RQ2 Die beliebtesten automatisierte Refactoring

- Die Kollidierung der Refactoring in automatisierte und manuelle Refactoring Daten reflektieren nicht die Beliebtheit der Allgemein.
- Jedoch stimmen die Aussage der Top 5 Refactoring für die erfahrene Entwicklergruppe
  - Mehr als 50 Stunden Nutzungserfahrung
  - 5 – 10 Jahre Erfahrung
  - Mehr als 10 Jahre Erfahrung.

RQ3 Wie oft bevorzugt der Entwickler zu manuellen trotz Wissen über automatisierte Refactoring?

- Ein Grund für manuellen Refactoring ist meist das Unwissen über die Existenz des Tools.
- Um einen besseren Durchblick in diese Frage zu bekommen wird unserer Teilnehmer mit eine der 3 Eigenschaften zugeordnet

RQ3

- High full automation
  - Teilnehmer die immer automatisierte Refactoring durchführen
- High informed underuse
  - Teilnehmer die trotz Wissens von automatisierte Refactoring-Tool bis 50 % manuell durchführen
- General informed underuse
  - Teilnehmer die signifikant wenig automatisierte Refactoring nutzen

RQ3

Category	Group	Property		
		High full automation	**High informed underuse**	**General informed underuse**
Aggregated		Extract Constant Rename Class	Extract Method Rename Local Variable	All
Affiliation	Students	-Extract Constant -Rename Class	+Conv. Loc. Var. to Field +Extract Constant +Rename Method	
	Professionals	+Rename Method	-Rename Local Variable	-Extract Constant -Rename Class -Rename Method
Usage time	$\leq$ 50 hours	-Extract Constant +Rename Method		-Extract Constant -Rename Class
	$>$ 50 hours	-Rename Class	+Rename Method	-Extract Constant
Experience	$<$ 5 years	-Extract Constant -Rename Class		-Conv. Loc. Var. to Field -Extract Constant -Inline Local Variable
	5 – 10 years	-Extract Constant	-Rename Local Variable +Conv. Loc. Var. to Field +Extract Constant	-Extract Constant
	$>$ 10 years	-Extract Constant	+Encapsulate Field +Rename Method	

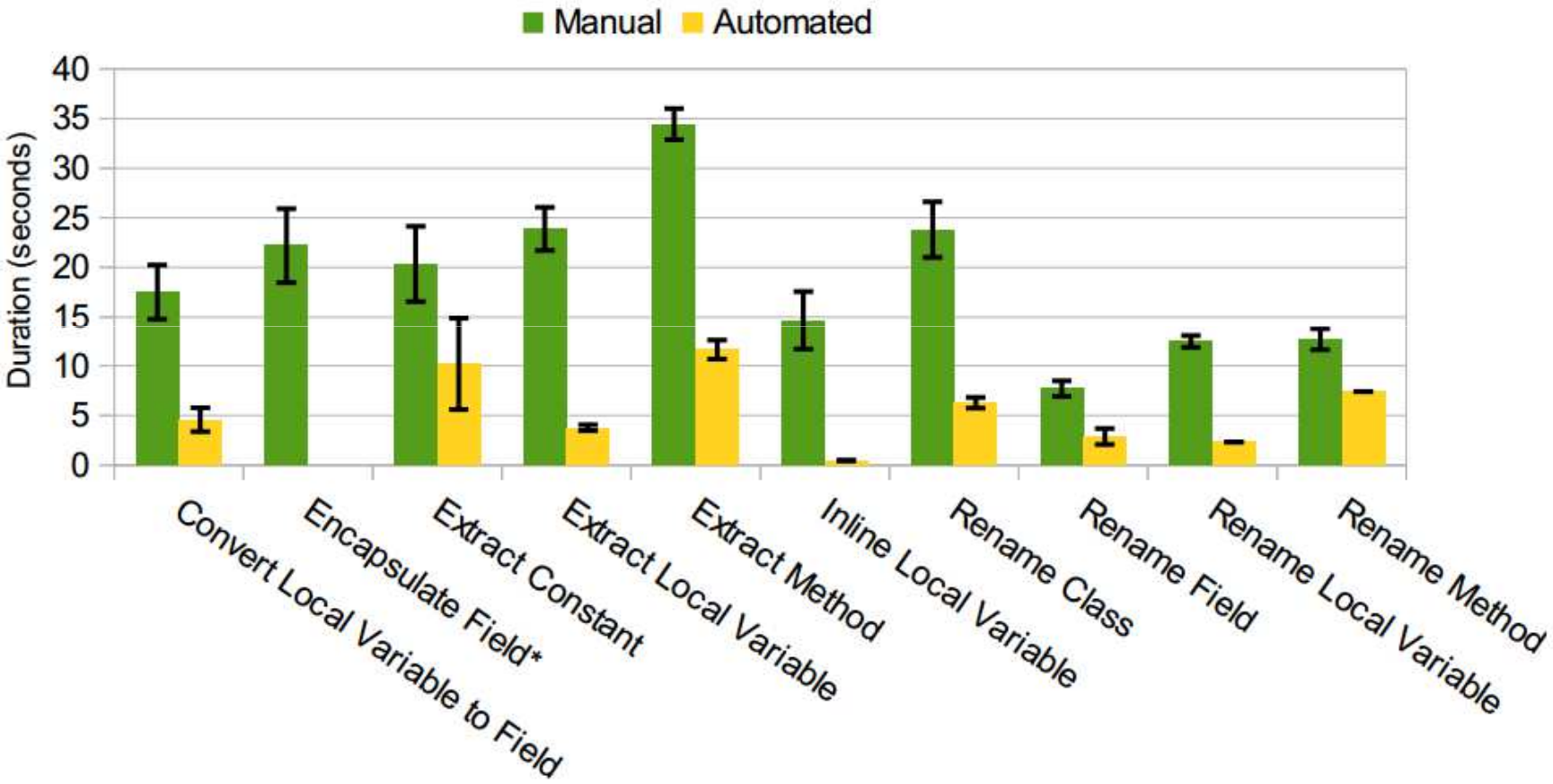
RQ3

- Refactoring Extract Constant und Rename Class werden meist in allen Entwicklergruppen durchgeführt.
- Refactoring Extract Method und Rename Local Variable werden so gut wie immer in allen Entwicklergruppen vermieden.
- Refactoring Conv. Loc. Var. To Field, Extract Constant und Inline Local Variable werden von Studenten komplett manuell ausgeführt.

RQ4 Spielt die zeitlichen eine Rolle bei der Refactoring-Wahl?

- Zeitlichen Aufwand der manuellen und automatisierte Refactoring werden aus timestamp der aufgenommenen Daten aufaddiert.
  - Fehlerrate durch Unterbrechungen oder jegliche Verzögerungen sind bei manuelle Refactoring berücksichtigt
  - Bei automatisierte Refactoring: Überprüfung der Refactoring wird nicht berücksichtigt

RQ4


Manual
Automated
Duration (seconds)
40
35
30
25
20
15
10
5
0
Convert Local Variable to Field
Encapsulate Field*
Extract Constant
Extract Local Variable
Extract Method
Inline Local Variable
Rename Class
Rename Field
Rename Local Variable
Rename Method

RQ4

- Die beobachteten manuellen Refactoring Zeitdauer sind durchschnittlich 15-25 Sekunden.
- Ein Entwickler wird bei der Wahl zwischen automatisierte und manuellen Refactoring die Differenz Zeit als Kalkulation berücksichtigen.
  - Je größer der Differenz desto mehr zieht der Entwickler die automatisierte Refactoring vor
- Ein eindeutiges Verhalten lässt sich rein von Statistik nicht nachvollziehen. Neben der Zeit Kalkulation, lässt sich erkennen, dass kleine zeitlichen Aufwand mehr zu manuellen Refactoring

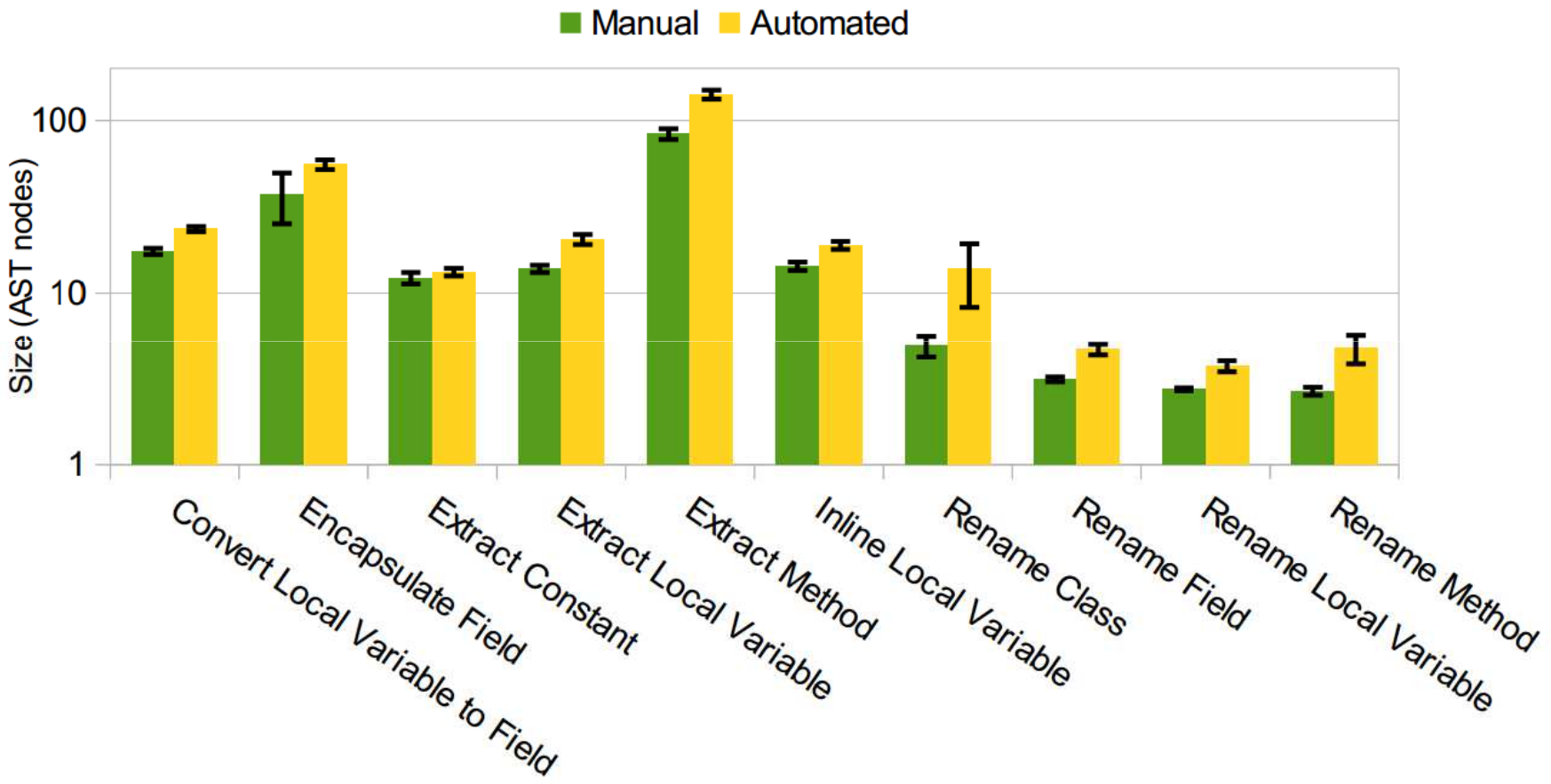
RQ4

Refactoring	M-Zeit	A-Zeit	Diferenz Zeit
Re. Loc. Var.	13	3	10
Re. Field	7	3	4
Extract Method	34	11	23
Re Method	13	7	6
Extrac Loc. Var.	24	4	20
Convert Loc. Var. To Field	17	5	12
Re Class	24	11	13
Inline Loc. Var.	14	1	13
Encapsulate Field	22	-	-
Extract Constant	20	10	10

RQ5 Entscheidet die Aufwendungsgröße die Nutzung des automatisierte Refactoring?

- Ähnlich wie bei Zeitaufwendung wird hier die Aufwandsgröße zusammen addiert und analysiert.
- Intuitive würde man behaupten, dass je Größe der Refactoring desto neigt der Entwickler für automatisierte Refactoring.
  - Doch die Beobachtung widerspricht diese ersten Aussage.

RQ5


Manual
Automated
Size (AST nodes)
100
10
1
Convert Local Variable to Field
Encapsulate Field
Extract Constant
Extract Local Variable
Extract Method
Inline Local Variable
Rename Class
Rename Field
Rename Local Variable
Rename Method

RQ6 Wie viele Refactoring wurde in Gruppe, also aufeinanderfolgend verwendet?

- Die Erkennungslgorithmus der zusammengehörigen aufeinanderfolgenden Refactoring werden basieren auf AST Daten mit der Greedy-Algorithmus selektiert
  - Refactoring Pattern, dass in unterschiedlichen Task ausgeführt wurde.
  - Nach der aufeinanderfolgenden Refactoring in nahezu gleichen Zeitfenster
  - …

RQ7

- Das erkennen der aufeinander gehörigen Refactoring ist meisten ein Zeichen für hochwertigen Refactoring-Pattern.
  - Dies kann man als Erkenntnis auffassen für nächsten Refactoring-Tool updates mehr Aufmerksam widmen.
- Die Beobachtung zeigen, dass Entwickler insgesamt zu hochwertigen Refactoring nutzen. Außer die Anfänger Entwickler Gruppe.

Fazit aus der 7 Teilantworten der 7 RQ

- Trotz der enormen Nutzung des automatisierte Refactoring
  - Wird Refactoring wird es noch nicht zu 50 % bei Entwicklern verwendet
  - Es gibt unerklärtes Unterinanspruchsnahme der automatische Refactoring
  - Berührungsängste oder schwerzugängliche Tools für Novizen

Fragen

Vielen Dank