Code Quality Issues in Student Programs

Hieke Keuning

OUrsi 9 May 2017

Open University of the Netherlands
Windesheim University of Applied Sciences

Open Universiteit
www.ou.nl

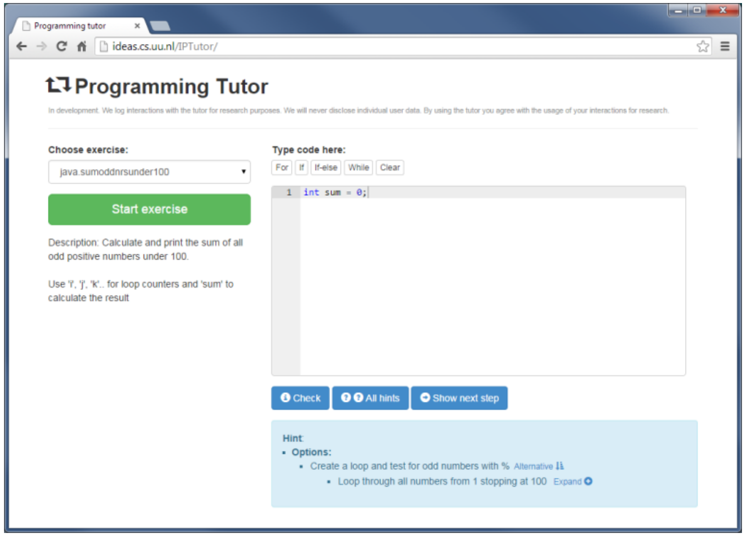
Windesheim

About me

04 – now	Lecturer Software Engineering	Windesheim
07 – 14	Student Master Computer Science	
15 – now	PhD candidate (NWO Doctoral grant for teachers) supervised by prof. dr. Johan Jeuring and dr. Bastiaan Heeren	

Master thesis [Keuning14]

Designing a programming tutor giving stepwise feedback using the IDEAS framework



IDEAS

PhD

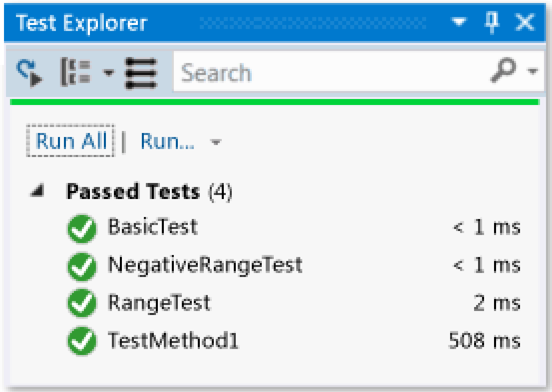
- Review of programming feedback
- Code quality in student programs
- Feedback for improving student code

Code Quality Issues in Student Programs

[Keuning17], to be presented @ITiCSE 2017: ACM Conference on Innovation and Technology in Computer Science Education

Problems with low code quality

- Affect software quality
- Students are unaware
- Not much attention in courses (more focus on correctness)



[www.codehunt.com]

Issues in low quality code

- Duplicates
- Too complex
- Too long (classes, methods)
- Unsuitable types
- …

```
if(! (a && !b) == true)
{
    System.out.print("Something else");
    System.out.print("the same");
} else {
    System.out.print("the same");
}
```

Studies on student code

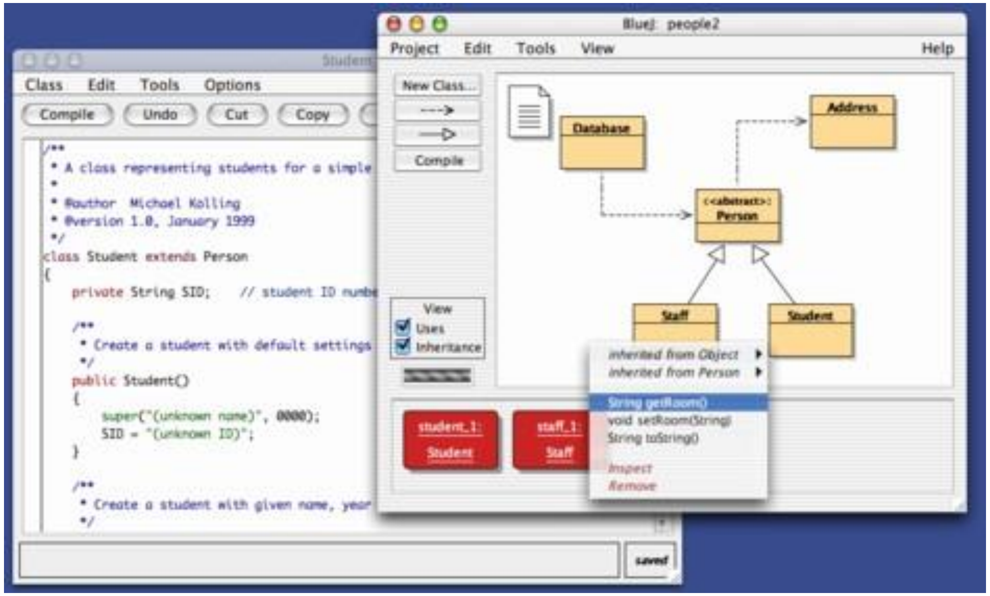
- Characteristics and code smells in kids' Scratch programs [Aivaloglou16]
- Some high-level metrics in student programs [Pettit15]
- Differences in quality between 1st and 2nd year students [Breuker11]

Research questions

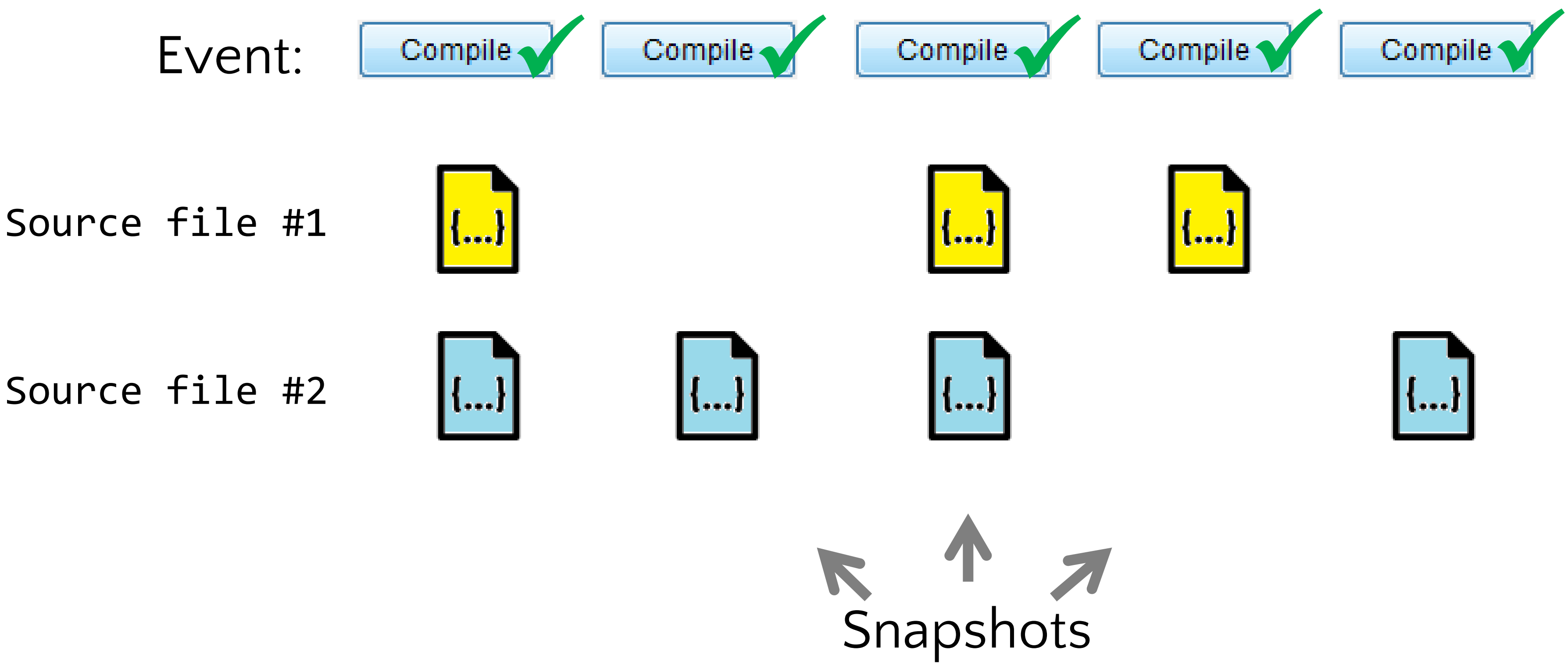
1. Which code quality issues occur?
2. How often are code quality issues fixed?
3. What are the differences in the occurrence of code quality issues between students who use code analysis extensions compared to students who do not?

Method

- Blackbox data set: 4 weeks of 2014-2015 from BlueJ
- Automated analysis with PMD



Blackbox data set



Total: 2,661,528 snapshots of 453,526 unique source files

PMD [pmd.github.io]

- Static analysis tool
- Detects bad coding practices
- Sample output:

```
C:\Sample.java:1: Possible God class (WMC=1231, ATFD=8, TCC=0.0)
C:\Sample.java:51: A high ratio of statements to labels in a switch statement.
Consider refactoring.
C:\Sample.java:511: A switch statement does not contain a break
C:\Sample.java:846: The default label should be the last label in a switch statement
C:\Sample.java:1034: Position literals first in String comparisons for
EqualsIgnoreCase
C:\Sample.java:2267: Avoid unnecessary comparisons in boolean expressions
C:\Sample.java:6617: Switch statements should have a default label
```

Categories [Stegeman16]

- Flow
- Idiom
- Expressions
- Decomposition
- Modularization
- Names
- Headers
- Comments
- Layout
- Formatting

First issue selection

From 26 sets (>280 issues) → 12 sets (170 issues), ran on data set of 439.066 code snapshots

Set	Issues seen	% of files with issues from set	Median %	Max %
Type resolution	4/4	26.04	3.96	20.1
Optimization	12/12	91.75	2.71	84.2
Unused code	5/5	26.86	2.50	16.2
Code duplication	3/3	4.99	2.28	5.0
Code size	13/13	13.69	1.40	8.2
Controversial	21/22	65.10	1.37	38.6
Import statements	6/6	10.61	1.02	8.5
Design	54/57	81.73	0.32	38.0
Unnecessary	8/8	10.25	0.11	9.6
Empty code	10/11	5.18	0.08	2.2
Coupling	3/5	41.98	0.04	39.7
Basic	23/24	2.52	0.02	1.3

Top 10 issues

Set	Issue	In % of files
Optimization	MethodArgumentCouldBeFinal	84.2
Optimization	LocalVariableCouldBeFinal	61.3
Coupling	LawOfDemeter	39.7
Controversial	DataflowAnomalyAnalysis	38.6
Design	UseVarargs	38.0
Design	UseUtilityClass	36.2
Design	ImmutableField	27.8
Type Res.	UnusedImports	20.1
Unused Code	UnusedLocalVariable	16.2
Controversial	AvoidLiteralsInIfCondition	14.0

Final set of 24 issues

Category	Some examples
Flow	CyclomaticComplexity PrematureDeclaration
Idiom	SwitchStmtsShouldHaveDefault AvoidInstantiatingObjectsInLoops
Expressions	ConfusingTernary SimplifyBooleanExpressions
Decomposition	NCSSMethodCount CodeDuplication
Modularization	TooManyMethods GodClass

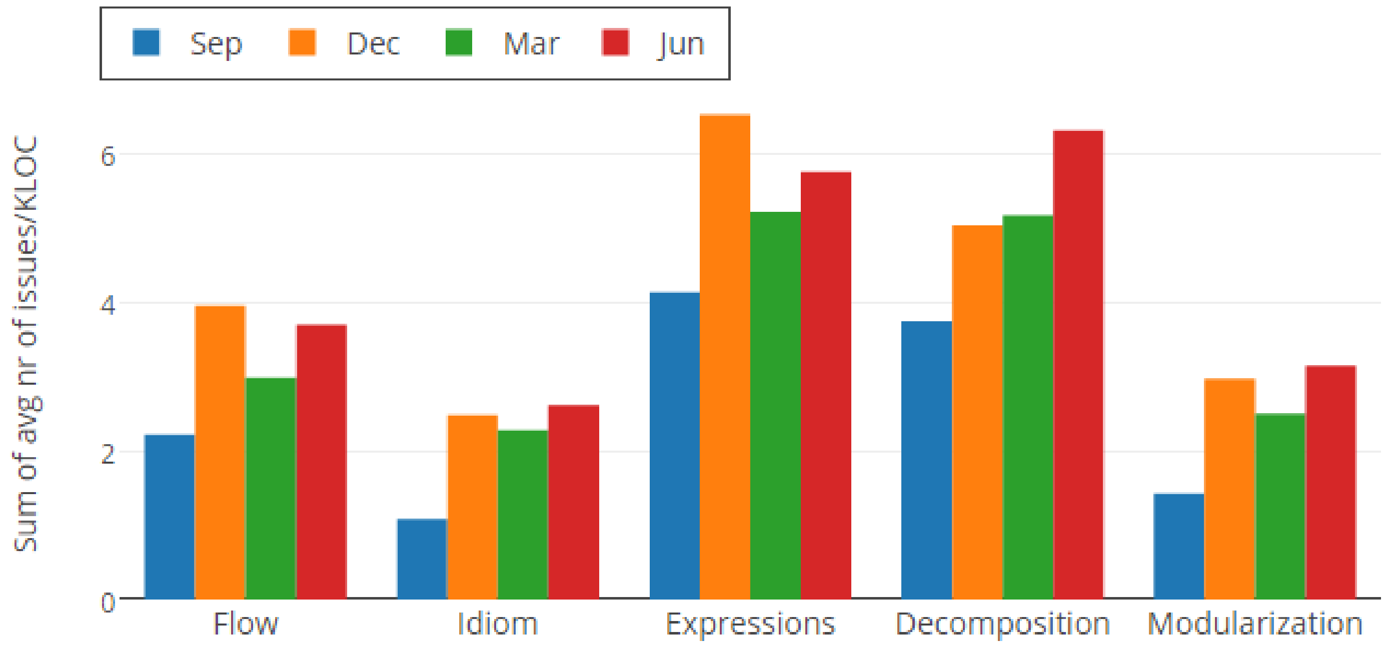
RQ1 Issue occurrence

I Per issue, the % of unique files in which the issue occurs,

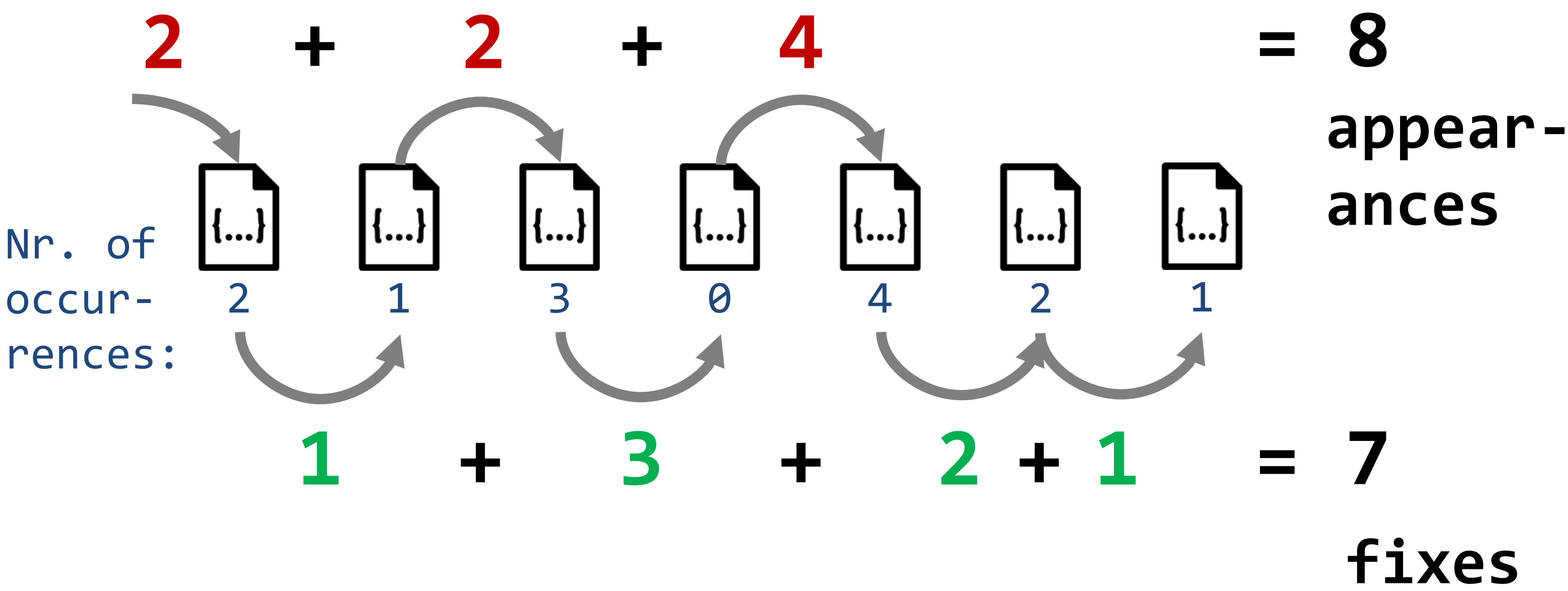
II the avg number of occurrences per KLOC

Cat	Issue	I	II
M	LawOfDemeter	38.7	42.6
D	SingularField	8.2	3.8
F	CyclomaticComplexity	7.7	1.5
M	LooseCoupling	6.7	2.1
I	AvoidInstantiatingObjectsInLoops	6.3	1.6
E	AvoidReassigningParameters	5.7	1.7
F	ModifiedCyclomaticComplexity	5.2	0.8
M	TooManyMethods	5.0	0.3
D	Duplicate50	4.7	0.7
E	ConfusingTernary	4.4	0.7
D	NcssMethodCount50	3.9	0.3
E	PositionLiteralsFirstInComparisons	3.5	1.6
F	NPathComplexity	3.3	0.3
E	SimplifyBooleanExpressions	3.1	0.8
F	PrematureDeclaration	2.6	0.4
M	GodClass	2.1	0.1
F	EmptyIfStmt	2.0	0.3
E	SimplifyBooleanReturns	1.9	0.4
I	SwitchStmtsShouldHaveDefault	1.7	0.3
I	MissingBreakInSwitch	1.4	0.2
D	Duplicate100	1.3	0.1
E	CollapsibleIfStatements	1.3	0.2
M	TooManyFields	1.2	0.1
D	NcssMethodCount100	1.0	0.0

Issue occurrence over time



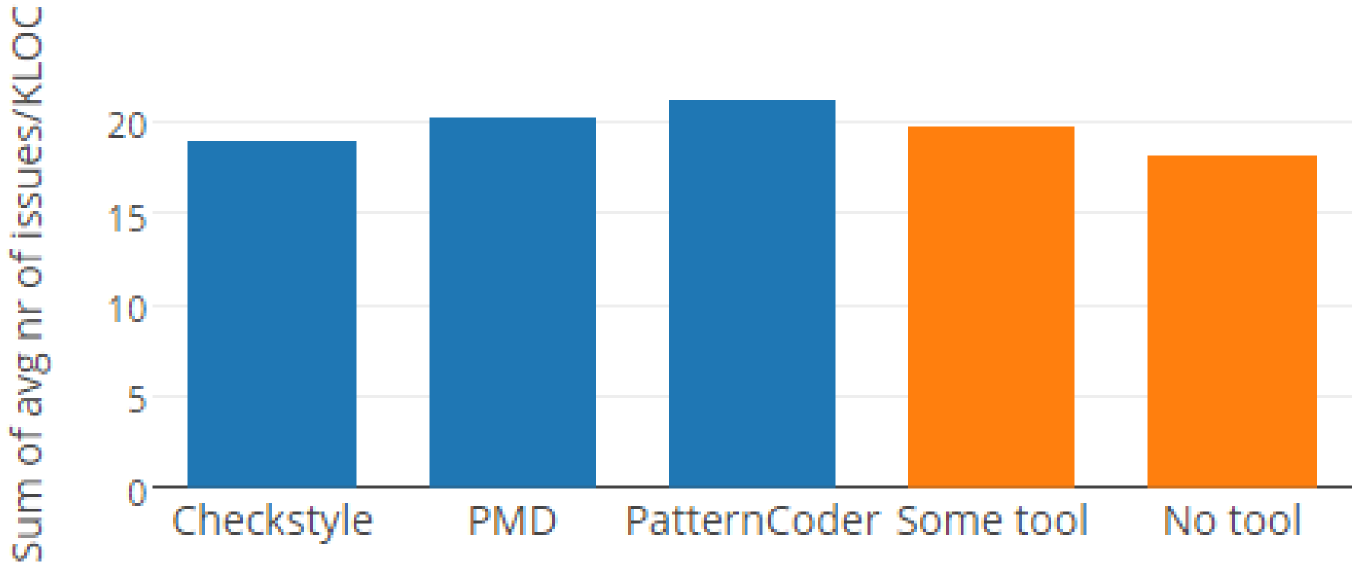
RQ2 Fixing



RQ2 Fixing

Cat	Issue	Appeared	Fixed	%
F	EmptyIfStmt	18,460	9,064	49.1
D	SingularField	103,004	30,152	29.3
F	PrematureDeclaration	21,008	5,891	28.0
D	Duplicate100	35,033	7,388	21.1
E	CollapsibleIfStatements	15,087	2,579	17.1
D	Duplicate50	91,951	15,520	16.9
E	AvoidReassigningParameters	76,359	10,023	13.1
I	MissingBreakInSwitch	9,594	1,033	10.8
F	NPathComplexity	20,549	2,129	10.4
E	ConfusingTernary	36,391	3,558	9.8
E	SimplifyBooleanReturns	12,612	1,162	9.2
E	SimplifyBooleanExpressions	48,965	4,347	8.9
F	ModifiedCyclomaticComplexity	56,822	4,475	7.9
I	AvoidInstantiatingObjectsInLoops	78,588	6,167	7.8
I	SwitchStmtsShouldHaveDefault	12,507	961	7.7
D	NcssMethodCount50	23,569	1,790	7.6
F	CyclomaticComplexity	85,426	6,240	7.3
D	NcssMethodCount100	6,178	410	6.6
E	PositionLiteralsFirstInComparisons	86,536	4,833	5.6
M	GodClass	9,575	437	4.6
M	LooseCoupling	57,039	2,056	3.6
M	TooManyFields	5,539	175	3.2
M	TooManyMethods	23,003	515	2.2

RQ3 Extensions



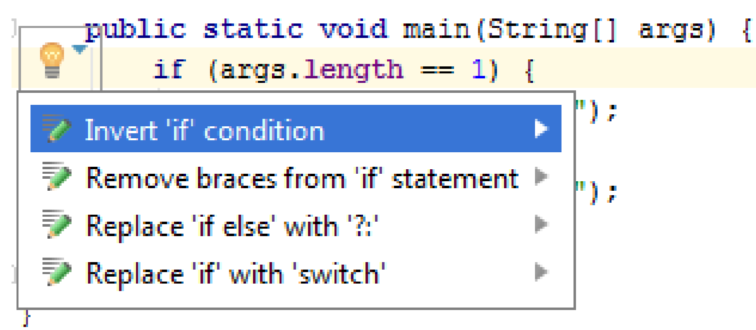
Conclusion

- Novice programmers develop programs with a substantial amount of code quality issues
- Do not seem to fix them, especially when related to modularization
- The use of tools has little effect

Recommendations and future work

- Spending more time on quality in courses
- Better understanding problems students & educators
- Improving suitability of quality tools for novices

```
public static void main(String[] args) {
    if (args.length == 1) {
        System.out.println("a");
    } else {
        System.out.println("b");
    }
}
```



ITiCSE Working group: Perceptions of Code Quality

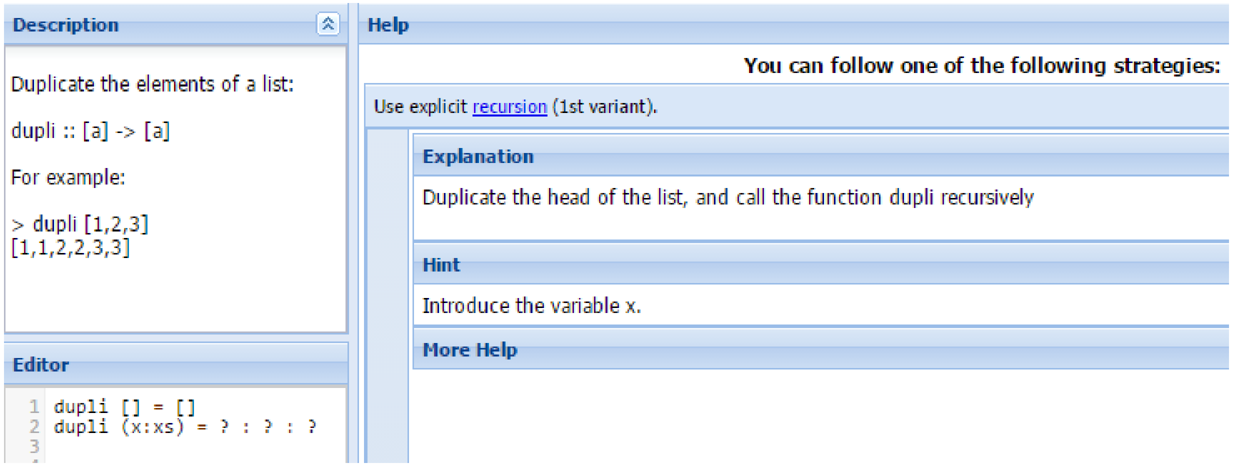
Intended **contributions**:

- Operational definitions of quality aspects that are considered important
- Examples of code that are considered ‘good’ or ‘bad’ with respect to some of the quality aspects

Method: Structured interviews with students, educators and professionals

Review of programming feedback

[Keuning16]



[Gerdes12]

The program requires **3** changes:

- In the return statement **return deriv** in **line 5**, replace **deriv** by **[0]**.
- In the comparison expression **(poly[e] == 0)** in **line 7**, change **(poly[e] == 0)** to **False**.
- In the expression **range(0, len(poly))** in **line 6**, increment **0** by **1**.

[Singh13]

[Moghadam15]

Feedback in programming tutors

Research questions

1. What is the nature of the feedback that is generated?
2. Which techniques are used to generate the feedback?
3. How can the tool be adapted by teachers?
4. What is known about the quality and effectiveness of the feedback or tool?

Systematic Literature Review

Find relevant tools:

- 17 review papers
- Database search
- ‘Snowballing’

- Selections & discussion mostly by 2 authors
- Strict criteria

RQ1	
KTC	Knowledge about task constraints
TR	Hints on task requirements
TPR	Hints on task-processing rules
KC	Knowledge about concepts
EXP	Explanations on subject matter
EXA	Examples illustrating concepts
KM	Knowledge about mistakes (○ basic or ● detailed)
TF	Test failures
CE	Compiler errors
SE	Solution errors
SI	Style issues
PI	Performance issues
KH	Knowledge about how to proceed (◐ hint, ◑ solution or ● both)
EC	Bug-related hints for error correction
TPS	Task-processing steps
KMC	Knowledge about meta-cognition

Coding labels RQ1

RQ2		RQ3		RQ4	
MT	Model tracing	ST	Solution templates	ANC	Anecdotal assessment
CBM	Constraint-based modelling	MS	Model solutions	ANL	Analytical assessment
AT	Automated testing	TD	Test data	EM-LO	Empirical - Learning outcome evaluations
SA	Basic static code analysis	ED	Error data	EM-SU	Empirical - Surveys
PT	Program transformations	SM	Student model	EM-TA	Empirical - Technical analysis
IBD	Intension-based diagnosis				
EX	External tools				

Coding labels RQ2-4

Name, reference	Language	Ex. class	RQ1 Feedback type												RQ2 Technique								RQ3 Adaptability						RQ4 Evaluation				
			KTC		KC		KM					KH		KMC																			
			TR	TPR	EXP	EXA	TF	CE	SE	SI	PI	EC	TPS		MT	CBM	AT	BSA	PT	IBD	EX	Other	ST	MS	TD	ED	SM	Other	ANC	ANL	EM-LO	EM-SU	EM-TA
(Chen04)	Imp/OO	C3					●										●							●	●	●			●				
(Sant09)	Imp/OO	C3					●			●							●				●				●				●				
(Chang00)	Imp/OO	C2							●											●			●	●						●	●		
(Fischer06)	Imp/OO	C3	●				●	○		○							●							●	●						●		
(Naur64)	Imp/OO	C3					●				○						●								●				●				
(Bettini04)	Imp/OO	C3					●	●									●						●	●	●						●	●	
(Harris04)	Imp/OO	C3					●	●									●								●							●	
(Jackson00)	Imp/OO	C3					●			●							●	●						●	●				●				
(Hong04)	Log	C3							●			◐	◐						●	●				●									●
(Coffman10)	Web	C3					●										●						●		●								
(Isaacson89)	Imp/OO	C3					○	○									●								●								
(Sztipanovits08)	Web	C3					○										●								●								
ACT Programming Tutor (APT)	Multi	C2		●			○		●				●		●		●													●	●		
ADAPT	Log	C3		●	●	●			●				●						●	●				●									●
APOGEE	Web	C3					●										●						●		●							●	
APROPOS2	Log	C3							●			●					●	●	●	●													●
ASAP	Imp/OO	C3					●										●								●							●	
ASSYST	Imp/OO	C3					○			●	●						●	●						●	●				●				
AnalyseC	Imp/OO	C3							●			◑						●	●					●									●
Ask-Elle	Fun	C3			●		●	●					●		●		●		●					●	●			●				●	●
AutoGrader	Imp/OO	C3					●			●							●				●		●	●	●			●	●				
AutoLEP	Imp/OO	C3					●	●	●								●	●	●					●	●						●	●	●
Automatic Marker for Sakai	Imp/OO	C3					●	●									●								●							●	
BIP	Imp/OO	C3	●	●	●		●	●				◐					●	●						●	●			●			●		
BOSS	Multi	C3					○										●						●		●					●		●	
Bridge	Imp/OO	C2							●			●	●		●															●			
Ceilidh	Multi	C3					●	●	○	○	○						●	●			●		●	●	●			●				●	
CourseMarker/CourseMaster	Multi	C3			●		●	●	●	●	●						●	●			●		●	●	●			●			●	●	
DISCOVER	Imp/OO	C2						●	●			◐	●		●									●				●		●	●		
ELM-PE/ELM-ART (II)	Fun	C3			●	●	●		●			●	●				●		●	●							●			●	●	●	●
ELP	Imp/OO	C2					●		●			◑					●	●	●				●	●	●			●				●	
EduComponents	Multi	C3					●	●									●							●	●							●	
GAME (2, 2+)	Multi	C3					○	●	○	○							●	●						●	●								●
HOGG	Imp/OO	C3					●										●						●	●	●				●				
HabiPro	Imp/OO	C2				●						●		●								●	●	●			●	●			●	●	
INCOM	Log	C3	●	●					●			◐				●			●									●			●	●	●
INTELLITUTOR (II)	Imp/OO	C2							●			◐						●	●														●
ITEM/IP	Imp/OO	C3		●			○										●							●	●						●		
InSTEP	Imp/OO	C2					●	●	●			◐					●	●					●		●						●		
JACK	Imp/OO	C3	●				●	●	●								●	●	●				●		●	●		●			●	●	●
JITS	Imp/OO	C2					○	○				●					●	●	●				●		●					●	●	●	
Kassandra	Imp/OO	C3					●										●							●	●							●	
LAURA	Imp/OO	C2							●										●					●									●
Ludwig	Imp/OO	C3					●	●		●							●	●						●	●			●				●	
MarmoSet	Multi	C3					●		●								●				●				●							●	
Mooshak	Imp/OO	C3					○	●									●								●				●				
Online Judge	Imp/OO	C3					○	○			○						●								●				●				
PASS	Imp/OO	C3					●										●								●							●	
PASS	Imp/OO	C2					●		●			◐					●		●	●				●									●
PATTIE	Imp/OO	C3		●	●				●				●		●							●					●		●				
PROUST	Imp/OO	C3							●			◐							●	●													●
Praktomat	Imp/OO	C3					●	●									●								●							●	
ProPL	Imp/OO	C2	●						●			●	●									●									●		
Quiver	Imp/OO	C3					●	●									●						●	●	●				●				
RoboLIFT	Imp/OO	C3					●										●						●		●						●		
RoboProf	Imp/OO	C2					●										●							●	●						●		
SAC	Imp/OO	C3					○	●									●								●				●				
SIPLeS-II	Imp/OO	C3							●			◑						●	●					●				●					●
Scheme-robo	Fun	C3					●		○		○						●	●						●	●							●	
TRY	Multi	C3					●										●								●								
The LISP tutor	Fun	C2				●			●			●	●		●															●	●		
Virtual Programming Lab	Multi	C3					●			●							●				●				●							●	
Web-CAT	Multi	C3					●		●	●							●				●			●	●			●			●	●	
WebToTeach	Imp/OO	C3		●			●										●								●				●				
WebWork-JAG	Imp/OO	C3	●				●										●								●							●	
autograder	Imp/OO	C3					●	●									●							●	●							●	
datlab	Imp/OO	C3					●		●								●					●		●								●	●
submit	Imp/OO	C3					●	●		●							●	●							●							●	
xLx	Imp/OO	C3					●	●									●							●	●								

First results: 102 papers on 69 tools [Keuning16]

Review conclusions, for now

- Very few tools give feedback with ‘knowledge on how to proceed'
- Feedback is not that diverse, mainly focused on mistakes
- Teachers cannot easily adapt tools
- Overall, quality of tool evaluation is poor

Conclusions & my future work

- Use results from review & data analysis for further research of automated feedback
- Develop a tool that helps students improving code
- Experiment with students using the tool

hw.keuning@windesheim.nl

References

- **[Aivaloglou16]** Efthimia Aivaloglou and Felienne Hermans. 2016. How Kids Code and How We Know: An Exploratory Study on the Scratch Repository. In Proc. of ICER.
- **[Breuker11]** Dennis Breuker, Jan Derriks, and Jacob Brunekreef. 2011. Measuring Static Quality of Student Code. In Proc. of ITiCSE.
- **[Gerdes12]** Alex Gerdes. 2012. Ask-Elle: a Haskell Tutor, PhD thesis.
- **[Keuning14]** Hieke Keuning, Bastiaan Heeren, and Johan Jeuring. 2014. Strategy-based feedback in a programming tutor. In Proc. of CSERC.
- **[Keuning16]** Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2016. Towards a systematic review of automated feedback generation for programming exercises. Proc. of ITiCSE.
- **[Keuning17]** Hieke Keuning, Bastiaan Heeren, and Johan Jeuring. 2017. Code Quality Issues in Student Programs. To appear in Proc. of ITiCSE. online
- **[Moghadam15]** Joseph Moghadam, Rohan Roy Choudhury, HeZheng Yin, and Armando Fox. 2015. AutoStyle: Toward Coding Style Feedback At Scale. In Proc. of Learning @ Scale.
- **[Pettit15]** Raymond Pettit, John Homer, Roger Gee, Susan Mengel, and Adam Starbuck. 2015. An Empirical Study of Iterative Improvement in Programming Assignments. In Proc. of SIGCSE.
- **[Singh13]** Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. 2013. Automated feedback generation for introductory programming assignments. ACM SIGPLAN Not. 48(6).
- **[Stegeman16]** Martijn Stegeman, Erik Barendsen, and Sjaak Smetsers. 2016. Designing a Rubric for Feedback on Code Quality in Programming Courses. In Proc. of Koli Calling.