Exploring Students' Understanding of the Concept of Algorithm: Levels of Abstraction

Jacob Perrenet
Educational Service Centre
Technische Universiteit Eindhoven
P.O.Box 513, 5600MB Eindhoven
The Netherlands
+31-40-2473396
j.c.perrenet@tue.nl

Jan Friso Groote
Department of Computer Science
Technische Universiteit Eindhoven
P.O.Box 513, 5600MB Eindhoven
The Netherlands
+31-40-2475003
j.f.groote@tue.nl

Eric Kaasenbrood
Department of Computer Science
Technische Universiteit Eindhoven
P.O.Box 513, 5600MB Eindhoven
The Netherlands
+31-628189391
e.j.s.kaasenbrood@student.tue.nl

ABSTRACT

How do we know if our students are beginning to think like computer scientists? In this study we have defined four levels of abstraction in the thinking of computer science students about the concept of algorithm. We constructed a list of questions about algorithms to measure the answering level as an indication for the thinking level. This list was presented to various groups of Bachelor Computer Science students. The mean answering level increased between successive year groups as well as within year groups during the year, mainly from the second to the third level. Little relation was found between answering levels and test results on algorithm oriented courses. The study was inspired by the tradition of mathematics education research.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms

Algorithms.

Keywords

Computer science education, abstraction.

1. INTRODUCTION

In the Netherlands, Computer Science is not yet a mandatory subject in pre-university education and almost no tradition of Computer Science education research exists. This will probably change in the next decade. For the Computer Science Department of the Technische Universiteit Eindhoven a natural starting theme for educational research is the concept of *algorithm*. The educational program has a thoroughly mathematical character; the focus is on correctness by construction, reliability and the use of formal methods for design tools (the Eindhoven School). The modern development in software engineering, starting with system component construction before bringing the components together, has recently been incorporated. Still, compared to other Dutch Computing Science programs, learning to construct (elementary) algorithms is taught very thoroughly, using a high level of abstraction. So a natural choice for a focus of educational research is: *levels of abstraction in students' thinking about algorithms*.

2. LEVELS OF THINKING THEORY

Computer Science Education researchers like Hazzan [5] and Aharoni [3] investigate levels of abstraction in students' thinking about core concepts, starting with, for example, Data Structures concepts and Computability Theory concepts. Like them, we value the tradition of the close research field of mathematics education research. The mathematics education discipline is relatively mature [1] and many mathematical concepts are similar in nature to concepts in computer science [2], although sometimes the same concept has a different meaning (see Fant [4] for the very concept of algorithm as an example).

Hazzan's and Aharoni's research is inspired by the work of Skemp and his successors (see [7] for an overview). In their framework a level of abstraction has three interpretations [3], [5]:

1. Abstraction level as the quality of the relationships between the object of thought and the thinking person.
2. Abstraction level as a reflection of the so-called process-object duality.
3. Abstraction level as the degree of complexity of the concept of thought.

We prefer the second interpretation where a next level is reached when a student can interpret processes and relations between objects, as a new kind of objects. The process or relation becomes an object in itself.

In our opinion, when a student has reached a certain level of thinking, the lower levels still exist and are incorporated; lower levels can be evoked if necessary, in accordance with Hazzan [5]. Questions or problems do not necessary evoke the highest available level of thinking. Thus it is better to speak about the level of thinking in relation to a certain problem. Depending on the problem, a student with a high level of thinking has the possibility but not the necessity to react on the highest possible level.

3. RESEARCH DESIGN AND FINDINGS

We will describe our focus in the curriculum, the construction of an algorithm questionnaire and a set of scoring rules as an instrument to measure abstraction level. Our main hypotheses are:

1. *The abstraction level of successive year groups will increase.*
2. *The individual abstraction level will increase during a year's programme.*
3. *The individual grade on a subject test will correlate with the individual abstraction level.*

3.1 Design

Algorithm courses

Five algorithm-oriented Bachelor courses were selected in which students were intended to develop their understanding of the concept of algorithm, its construction and its analysis (Table 1).

Table 1. Algorithm-oriented courses

Subject and study phase	**Contents**
Program Realization 1 Trimester 1.1	Embedding elementary algorithms into larger programs and encoding into a specific programming language.
Design of Algorithms 1 Trimester 1.2	Systematical construction of algorithms using the Guarded Command Language[1].
Design of Algorithms 2 Trimester 2.1	Programming from a formal specification with stepwise refinement to the use of simple standard algorithms.
Design of Algorithms 3 Trimester 2.3	Mathematically analyzing algorithmic run-time. Using advanced techniques of algorithmic design and a number of standard algorithms.
Complexity Trimester 3.1	Understanding algorithm complexity as well as problem complexity. Transforming problems within complexity classes.

Levels of Abstraction

Originally, we proposed three abstraction levels for the algorithm concept, the *program* level, the *object* level and the *problem* level. These abstraction levels were discussed in interviews with the lecturers of the five courses involved in the study. An additional, lower level, the *execution* level was suggested. The resulting four levels are defined as follows:

1. *Execution* level: the algorithm is a specific run on a concrete specific machine; its execution time is determined by the machine.
2. *Program* level: the algorithm is a process, described by a specific executable programming language; execution time depends on the input.
3. *Object* level: the algorithm is not connected with a specific programming language; it can be viewed as an object (versus process); while constructing an algorithm the data structure and the invariance properties are used; meta properties such as termination and 'patterns' (algorithmic modules) are relevant; execution time is considered in terms of magnitude of order as function of the input.
4. *Problem* level: the algorithm can be viewed as a black box; the perspective of thought is 'given a problem, which type of algorithm is suitable?'; problems can be categorized to suitable algorithms; a problem has an intrinsic complexity.

The lecturers were most outspoken about the difference between levels 2 and 3. They also expected these two levels of understanding to be most common among the Bachelor student group. This opinion was shared with the majority of other teachers involved in the subjects of Table 1: faculty instructors for assisting in solving problems, and student assistants for assistance at instruction and correction of assignments. They expressed their opinions during short interviews.

Other comments of the lecturers were that they agreed upon an interpretation of 'abstraction' as 'disregarding detail' or 'condensation'. Only some of them accepted the process-object interpretation, as given above. Some made extra abstraction level distinctions, such as the difference between knowing an algorithm for a certain task and being able to prove that a certain algorithm can complete a certain task in a certain order of time, or such as thinking about a sequential or a parallel machine performing an algorithm. There was disagreement on whether the ability to use pseudo code and an axiomatic notation signified a certain level of abstraction or a certain level of precision only. The discussions with the lecturers inspired the contents and the format of the questions for the students.

Construction of the Questionnaire and Measurement

The original questionnaire consisted of 11 items. The starting item is worded as follows:

0. Give your definition of 'algorithm'.

The other ten items ('proposition items') have another format. There is a general introduction: *Mark whether you agree or disagree with the following proposition and give a supporting argument. So, only 'agree' or 'disagree' is not sufficient. If necessary, the option 'both are possible' can be chosen, provided that an argument is given. Only if you cannot answer the question because of lack of knowledge, then choose for 'I don't know'.*

All ten items are followed by the four alternatives *Agree*, *Disagree*, *Agree and disagree are possible*, *I don't know*, and room for supporting argumentation.

The questionnaire was presented to the three Bachelor year groups in the first trimester after concluding their subject test on Program Realization 1, Design of Algorithms 2 or Complexity (Table 1). Further data were collected from the first year group at the end of the second semester after concluding the subject test of Design of Algorithms 1, and from the second year group at the end of the third trimester at Design of Algorithms 3. In total 398 questionnaires were collected (out of 461 students who completed the relevant courses).

A random sample of 5% of the first trimester (student) output was scored by two raters (the second and third author). The agreement between the raters at item level appeared to be too small, so differences and doubts were discussed and scoring rules were refined. This first sample was not used in further analysis. Another 5% sample still resulted in unsatisfactory agreement. However, after removal of four out of the ten proposition items, the degree of agreement between the two raters, measured by Cohen's Kappa, was .64. Also on the remaining list – item 0 and six proposition items - a third rater (the first author) scored

[1] See Kaldewaij [6].

satisfactorily consistently with the other two (.64 and .70). Cohen's Kappa is a measure that corrects for the influence of chance and a minimum of .60 is considered acceptable.

We will present the final list of six proposition items. Together with the item 0 (Give your definition of 'algorithm') this list was used as the questionnaire in further data collection and analysis.

1. An algorithm is a program, written in a programming language.
2. Two different programs written in the same programming language can be implementations of the same algorithm.
3. The correctness of an algorithm can generally be proven by testing the implementation on cleverly selected test cases.
4. A suitable quantity to measure the time needed for a certain algorithm to solve a certain problem is the time needed in milliseconds.
5. The complexity of a problem is independent of the choice of the algorithm to solve it.
6. For every problem it is possible that in the future algorithms are discovered which are more efficient by an order of magnitude than the algorithms known at present.

For items 0 to 4 a level score from 1 to 3 is possible, for items 5 and 6 a score from 1 to 4. In Table 2, as an example, the refined scoring rules are given for three items. As there is no information about the distances between the levels, we calculated a student's answer abstraction-level as the *median* of the series of item level scores. If for more than half of the item series no level was detectable (missing or unclear) then the data were considered to be insufficient to calculate a students' level.

3.2 Results

Students' Answering Levels

Generally, students took approximately fifteen minutes to answer the seven questions. From the answers the level could be calculated for almost 85% of the students filling in the questionnaire; the other students were too often not clear in their argumentation for the proposition items or they gave no argumentation at all.

The first hypothesis was, that *the abstraction level of successive year groups would increase*. Table 3 shows the percentages of answering level scores for the year groups 1, 2 and 3 at the end of the first trimester.

Table 3. Abstraction level of successive year groups

Year group in trimester	Student percentage with level score				Number of students
Level score	1.5	2	2.5	3	
Year group 1 in 1.1	4	50	6	40	67
Year group 2 in 2.1		21	7	72	58
Year group 3 in 3.1		8	2	90	72

Indeed, in higher years, the answering level generally is higher: Spearmann's rank correlation coefficient rho = 0.48, significant at 0.01 (two-tailed). Almost every student has a median answer level of 2 to 3; levels under 1.5 or above 3 are non-existent.

Our second hypothesis was, that *the abstraction level of the same group would increase during the year*. Table 4 shows the general answering level growth of the first year group and the second year group. The percentages are given of students with a higher level, the same level or a lower level at the second measuring moment, compared to the first.

Table 2. Examples of detailed answer level scoring

Item	Answer characteristics	Answer level
0	A process with only one input, and only on one machine	1
	A program in a specific language, runable with all possible input	2
	A series of steps (abstraction from programming language)	3
	Unclear or missing	x
1	+ An algorithm equals an execution equals a program, on a machine or on a virtual machine	1
	+ Implementation or effectuation in a programming language; program equals algorithm; executable on a machine or by hand	2
	- Implementation into differing languages	3
	± Implementation or effectuation etc is clearly mentioned (see above)	2
	± Implementation into differing languages	3
	?, +, -, ± Without an argument; unclear or missing	x
6	+ Computers will become faster	1
	+ By optimization of an actual program or by use of a better programming language	2
	+ By different architectures for computers / by optimization of a computational method (not related to a specific programming language); the solving algorithm determines the complexity, the algorithm is placed above the problem	3
	- One specific counter-example is given	3
	- The black-box approach emerges / upper bounds and lower bounds as well as problems are placed above problems; a problem possesses an intrinsic complexity	4
	± The arguments for disagreement should emerge clearly; a problem possesses an intrinsic complexity	4
	? (Without further explanation)	x
	? (The word 'complexity' is not understood)	x
	? (The word 'complexity' is understood, but the 'complexity of a problem' is not)	3
	?, +, -, ± (Without an argument, unclear or missing)	x
Key to symbols used above: + = Agree, - = Disagree, ± = Agree and disagree, ? = I don't know; x = No answer level detectable		

Table 4. Abstraction level growth during the year

Year group (trimesters)	% with increased level	% with the same level	% with decreased level	Number of students
1 (1.1 to 1.2)	48	44	8	36
2 (2.1 to 2.3)	34	56	10	48

Clearly most students reach a higher level or stay at the same level: according to the Wilcoxon Signed Ranks Test significant in both cases at .05 (two-tailed, with Z=-2.47 and Z=-2.27 successively).

Our third hypothesis was that *the individual grades on the various subject tests would correlate with the abstraction level*. Table 5 shows the correlations between grade and level for the five subject tests. Only for one subject a small but significant (at 0.05) Spearmann's rank correlation appears: for Complexity. For the other subjects correlations are non-significant and close to 0 (Design of Algorithms 1, 2, 3 and Program Realization 1).

Table 5. Correlation between answering level and test grade

Subject test	Rank correlation	Number of students
Program Realization 1	.14	58
Design of Algorithms 1	.05	63
Design of Algorithms 2	-.05	44
Design of Algorithms 3	.09	72
Complexity	.27*	72
* = significant at 0.05 (two-tailed)		

4. CONCLUDING DISCUSSION

Students' Answering Levels

In this study, we found various abstraction levels in Computer Science Bachelor students' answers to questions about algorithms, although within a certain range. Assumed levels were 1. execution level, 2. program level, 3. object level, and 4. problem level; the mean answering level almost always only varied from program level (2) to object level (3). One could assume that presenting the questionnaire to Master students would show a higher level. However, part of the explanation is in the characteristics of the items and the method of measurement. For only two of the seven items, an answer at the fourth level was possible and the mean answering level (actually the median) was calculated. For emergence of this highest level as the mean, more items should have been constructed with the possibility of answering at that level. Another possibility would have been to use another operationalization for the measurement, for instance defining the answering level of a student as the highest answering level that is reached at least at two items. On the other hand, the lowest level did not show, although this answering level was possible at every item. The students involved had followed one semester of computer science education or more. It would be interesting to do further research into the existence of the lower execution level at the actual start of the computer science program or at the end of pre-university education. In that case however the questionnaire would need some adaptation: the freshmen computer science vocabulary is limited.

The results showed level growth for successive year groups (first hypothesis). A year group was defined as the group taking a test belonging to the curriculum year. However, slow students in their second year sometimes have to take a first year test again. And quick students in their second year sometimes already take a third year test. More detailed data analysis, taking these circumstances into account to select purer data, could reveal clearer results.

The results also showed level growth within year groups (second hypothesis), although a large number of students stayed at the same answering level. Measurement of the same groups after a longer period could reveal clearer results. One could argue that filling in the same questionnaire twice could create a learning effect, explaining the level growth result as an artifact. This would certainly be true when feedback was given; this, however, was not the case. And because filling in the questionnaire generally would have been of marginal value for the students, compared to the value of the subjects test itself, we feel confident about the reality of the results. In any case, more frequent measurement of the same group of students would require the construction of parallel items.

Both positive results above should however be viewed with some caution. The assumption is that the level growth was (partially) caused by taking the algorithm courses. However, correlation is not the same as causation. Following other courses could have caused the level growth or simply the process of getting 'older and wiser'. Only experimental research gives a view on causation.

Unexpectedly, in general little relation was found between subject test results and answering level (third hypothesis), in fact only at the third year course Complexity. Maybe *precision* really is more important than abstraction level for success at these subjects, especially at the hard-core Eindhoven School subjects (Design of Algorithms 1 and 2), as some of the lecturers suggested. A curriculum change is prepared for the coming years with a revision of these subjects. It would be interesting to repeat the investigation in the revised curriculum. It would also be interesting to compare answering level with general performance.

Reliability and Validity

We succeeded in the construction of a series of items plus a reliable scoring protocol for the measurement of the mean answering abstraction level for the concept of algorithm. Looking back on the measurement, we think a measurement output of 85% is acceptable, given the fact that many students have problems with clearly expressing their thoughts in writing (or maybe with clear thinking itself) at the subject tests, as any corrector knows.

Compared to the method of the related research of Hazzan [5] and Aharoni [3], who interviewed students and analyzed their work, in our study many more students were involved but with less data per student. The larger number of students makes statistical analysis possible, for example to establish reliability as one of the important aspects of a measurement. On the other hand, an important question remains for our study: What about the validity? Did we really measure *level of abstraction*? Did we not simply measure the students' memorization of standard definitions? One could argue that, if we had solely used a closed multiple-choice format in our questionnaire. However, we did not score the multiple-choice *answers*, but we analyzed the students' *argumentations* for their choices. This method fits in the tradition of mathematics education and its research: not to look at answers only. But a follow-up could improve the validity of the results indeed. One way would be to construct more items, and also a solution for ‘the problem of jargon’ would be welcome: An example of this problem is the impossibility of measuring the answering level of the items involving ‘complexity’, when it is unclear whether the technical meaning of the term ‘complexity’ is understood or not (items 5 and 6). Another manner to improve the validity would be to combine the measurement used by us with methods such as used by Hazan and Aharoni - interviewing students and analyzing written work and to associate the results; in that case for a smaller, heterogeneous group of students.

Generalizability

Because of the abstract Eindhoven School character of the curriculum it is hard to generalize the results to Dutch or general computer science university education. Comparative measurement at another university would be necessary. A somewhat risky hypothesis would be that the abstraction level of the Computer Science students at the Technische Universiteit Eindhoven, at least in the second and third year, would be higher.

Implications for Didactics and Didactical Research

Our results are too premature to conclude that teachers should take their students’ algorithmic thinking level into account more. Further research into algorithmic thinking levels would be required - at other institutions and at other moments in the curriculum – as well as research into *other* computer science concepts, and the relations *between* thinking levels for different concepts. We advise to use little data from many students as well as much data from few students. Knowing more about the students’ thought processes can improve teaching aimed at supporting students’ own knowledge construction. For computer science education, there still are worlds to discover!

5. ACKNOWLEDGMENTS

We would like to thank the students and the staff of the Computer Science Department of the Technische Universiteit Eindhoven for their cooperation in this research.

Special thanks goes to our colleagues Jos Arts, Irene Vanderfeesten and Christina Morgan for their helpful comments on an earlier version of this paper.

6. REFERENCES

[1] Almstrum, V.L., et.al. Import and Export to/from Computing Science Education: The Case of Mathematics Education Research. *Proceedings ITiCSE,* Aarhuus, 2002, 193-194.

[2] Almstrum, V.L., et.al. Transfer to/from Computing Science Education: The Case of Science Education Research. *Proceedings SIGCSE,* Reno, 2003, 303-304.

[3] Aharoni, D. Cogito, Ergo, Sum! Cognitive Processes of Students Dealing with Data Structures. *Proceedings SIGCSE,* Austin, 2000, 26-30.

[4] Fant, K.M. A Critical Review of the Notion of the Algorithm in Computer Science. *Proceedings ACM,* Indianapolis, 1993, 1-6.

[5] Hazzan, O. Reducing Abstraction Level when Learning Computability Concepts. *Proceedings ITCsE,* Aarhuus, 2002, 156-160.

[6] Kaldewaij, A. *Programming: The Derivation of Algorithms.* Prentice Hall International, UK, 1990.

[7] Tall, E. & Thomas, T. (Ed.). *Intelligence, Learning and Understanding in Mathematics; a tribute to Richard Skemp*. Post Pressed, Flaxton, 2002.