Teacher's Assessment in Programming – Comparing Teachers' Individual Judgement Criteria in a Programming Course

Lars Björklund and Charlotta Nordlöf

In schools around the world the part of technology education related to programming is increasing. There is a lot to learn about teacher's assessment and grading of students in assignments that are related to technology, particularly in programming. Simon (2012) analyzed introductory programming exams and found that a large percentage of the marks were awarded for the product and the coding skills of the student and less for other skills such as design, debugging, testing, or explaining and documenting. To be able to give formative support teachers should also be able to assess the process in the classroom; students tinkering, creating, debugging, persevering, and collaborating. The aim of this paper is therefore to examine teachers' individual criteria, explicit, tacit and subjective criteria, when they grade and assess students in technology tasks related to programming. We interviewed 6 teachers in Sweden, representing lower secondary school, upper secondary school and university (teacher and engineering education). A Repertory Grid Technique interview combined with a Comparative Judgement grading was used to examine teachers' individual criteria in assessment. The expected outcomes from the methods are captured criteria that are implicit and based on teachers' experience, sometimes seen as teachers' gut feelings. Two types of criteria were found; product criteria assessing the program and process criteria assessing the ongoing process. We compared these criteria with an instrument measuring the development of creativity designed for Art education. We claim that the use of process criteria will help the teacher and the students in developing programming skills.

Keywords: Technology education, assessment, programming, computational thinking, creativity, process criteria.

Introduction

Technology is a school subject of a wide range, that appears different in countries around the world (e.g. A. Jones, Buntting, & de Vries, 2013). In this paper, the focus is not on the whole subject of technology, but on the subject content related to programming. A common theme in schools around the world, including Sweden, is that the part of technology education related to programming is increasing. For example, in Sweden subject content related to programming was implemented in a revision of the curricula of the Technology school subject in compulsory school (Skolverket, 2017). Another example is New Zealand, where the technology area in the curricula has recently been revised with the aim to strengthen the digital technologies and to give all students the possibility to develop their digital capability. This revision was integrated in the start of the school year 2020 (Ministry of Education, 2018). In view of the increased amount of digital ingredients such as programming in technology education internationally, there is a lot to learn about teacher's assessment and grading of students in assignments that are related to technology, particularly in programming.

Since programming is a relatively new subject content in elementary school, there is little research done on assessment at that school level. However, there is some research on assessment at higher levels of education (Daly & Waldron, 2004; Harland, D'Souza, & Hamilton, 2013).

Internationally, programming is often related to computational thinking. Computational thinking (Wing, 2006) can be described as "a problem solving methodology that can be automated and transferred and

applied across subjects" (Barr & Stephenson, 2011, p. 115). The British project, Barefoot Computing (2014), created a model for computational thinking that includes:

- Logical reasoning: predicting and analysing
- Algorithms: making steps and rules
- Decomposition: breaking down into parts
- Abstraction: removing unnecessary detail
- Patterns and generalisation: spotting and using similarities
- Evaluation: making judgements

There are some projects trying to assess computational thinking skills. Bebras (Dagiené & Stupurienė, 2016) is an international initiative that started in Lithuania, with the aim to stimulate informatics (Computer Science, or Computing) and computational thinking in schools. It is constructed as an international contest where the students perform different tests. Logical thinking, pattern recognition and simple programming are tested. Teachers sometimes use Bebras in their assessment of students computational thinking. In a research study evaluating a Code Club, Bebras was used and the result showed that learning at the Club did not have any impact on pupils' computational thinking over and above changes that would have occurred anyway (Straw, Bamford, & Styles, 2017). Initiatives like Bebras and Barefoot focus on computational thinking, but it may be used in technology education as a part of the teaching. DrScratch is a web-based assessment tool using the idea of computational thinking and automatically grading a scratch program according to a special template and grading scale (Moreno-León, Robles, & Román-González, 2016).

In higher education emphasis is put on the skill of writing a program to code. Simon et al. (2012) analyzed introductory programming exams and found that a large percentage of the marks (81%) were awarded for the product and the coding skills of the student and much less for other skills of design, debugging, testing, or explaining and documenting.

It is not enough to know syntax or semantics of a language to be able to create a program. One needs to be able to put the pieces together to create a program, it is a kind of problem solving (Soloway, 1986). "Experts have built up large libraries of stereotypical solutions to problems as well as strategies for coordinating and composing them" (Soloway, 1986, p. 850). To be able to give formative support teachers should also be able to assess the process in the classroom, students tinkering, creating, debugging, persevering, and collaborating (Barefoot, 2014).

The aim of this paper is therefore to examine experienced teachers' individual criteria, explicit, tacit and subjective criteria, when they grade and assess students in technology tasks related to programming. Are they confined to assess the product or are they grading other qualities in the work of students?

Method

In this study teachers representing lower secondary school (n=1), upper secondary school (n=2) and university (n=3) were interviewed. All teachers had been teaching students in assignments related to computer programming. In upper secondary school and at universities, courses in programming have been an important part of the syllabus for a long time. Therefore, in this study we focus on teachers in upper secondary school and university to identify how they assess and grade their students. The results may be useful to technology teachers in lower grades where programming is a new topic.

The participating teachers were asked to bring seven to nine examples of student assignments which they had graded and were familiar with. They were asked to select some good, some poor, and some ordinary examples of assignments. These student examples are from now on referred to as *the elements*.

The first part of the interview was started by representing each element by a small paper card. The participants then made a holistic assessment of the elements using *Comparative judgment* (CJ) a method that has been used to assess student work in e.g. technology and science. CJ produces a ranking of elements with remarkable inter-rater reliability (I. Jones & Alcock, 2014; Pollitt, 2012). The interviewee was presented with two elements/cards at a time and was asked to compare them, "which one is the best?", by making a holistic judgement. These pair wise comparisons resulted in a ranking of the elements. Next, the participants were asked to grade the rank-ordered elements using a 1-9 marking scale. The result was documented and used in the second part of the interview.

The second part of the interview was performed by the Repertory Grid Technique (RGT). The Repertory Grid Technique is well known in the field of psychology and has also been used to analyse how teachers grade and assess in a range of subjects: Design & Technology, Sports, Physics, and Art (Björklund, 2008; Lindström, 2006; Svennberg, Meckbach, & Redelius, 2014). RGT was designed to be used in clinical psychology but has also since the 1960's been very useful in a variety of research areas.

The method was developed by Kelly (1955), starting from his Personal Construct Psychology, a psychological theory that explained why people have different views and attitudes towards actions in the world. People use personal criteria, constructs, to make meaning, and the theory builds on that statement. A construct is defined as both a similarity and a difference. A construct with one pole described as "Tidy" must be described with its contrasting pole “Sloppy”. Kelly designed the method to elicit personal constructs, The Role Construct Repertory Test. That test has then been developed by Kelly and others, and is now referred to as the Repertory Grid Technique (RGT) (Björklund, 2008).

Three cards representing the elements, created at the beginning of the interviews, were randomly chosen and presented to the interviewee with a question: "Does two of these have something in common that separates them from the third?". The goal was to find constructs that verbalised the teacher’s assessment by describing the differences between the constructs symbolised by the cards. For example, a participating teacher could describe two of the constructs as “Well written report” and the third as the opposite, “Poorly written report”. By doing that, a construct was found, and the interviewee was asked to place all the cards (representing elements) on the construct’s marking scale from 1 to 9. The result was documented in a grid. The procedure was repeated several times until no new constructs were elicited. The process usually gave six to ten different constructs or criteria. The gradings were put in a grid together with the result of the holistic assessment. The resulting grid was then statistically analysed using correlation and factor analysis.

Results

The results from one of the secondary school teachers (Gy1) will exemplify how this interview technique works. The teacher’s gradings of the six students (A, B, C, D, E, and F) using each and every criteria are documented in the grid shown in Figure 1. To the left of the grid the correlation value of each criteria compared with the holistic grading is shown. Good skills in coding (0.95) and showing an advanced level of technological understanding (0.89) were strongly correlated with the holistic grading. Behaviors like being enthusiastic (0.92), showing good persistence (0.89) and an ability to learn from others (0.75) also seemed to be important.

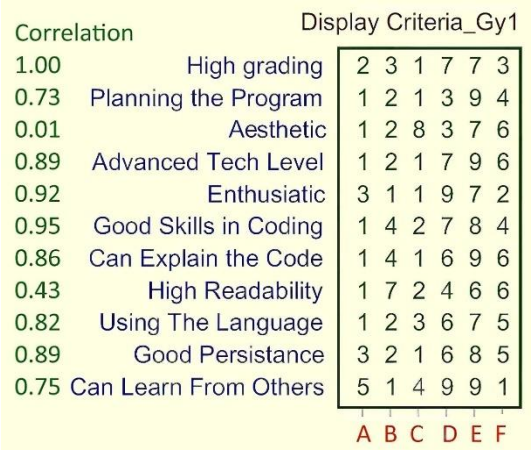

Figure 1. Display of elicited criteria from secondary school teacher (Gy1)

A graphical representation of the factor analysis of the data (Figure 2) shows what seems to matter most to this teacher when assessing her students. Criteria that are closely aligned with the holistic assessment (High grading) could be interpreted as important to this teacher.

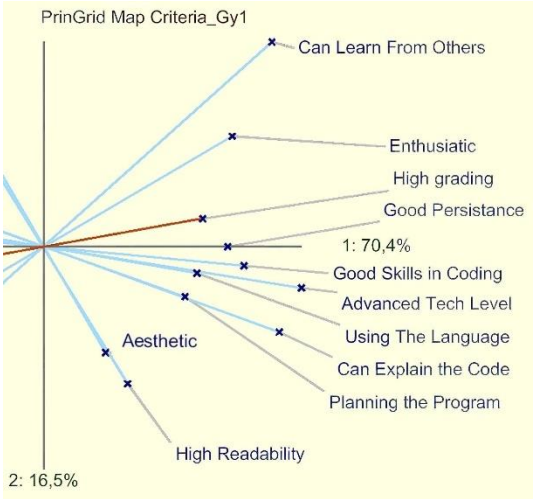

Figure 2. Factor analysis of the data from the same teacher.

As a result of the interviews several different criteria were found. It seemed to be productive to sort the criteria in two groups: product criteria that can be assessed looking at the final product, the program, and process criteria that assess the work in progress. In the following list, all the criteria elicited from the six teachers are sorted in two groups:

Criteria about the product, the program

- The program is complete.
- The function of the program, it fulfils its purpose.
- The student utilizes the various functions of the program language.
- The student can explain the code used in the program.
- The coding is advanced, on a high technical level.
- Readability is high, easy to follow and understand the program.
- Good coding.
- The program is accurate and not sloppy.
- The documentation is good.

Criteria analysing the process and how the student is working

- Endurance, the student struggles and doesn't give up.
- The student pushes the work forward and moves on.
- The student identifies the problem and makes a plan to find an effective way of solving it.
- The student has a trial and error approach, tinkering.
- The student is creative.
- The student can learn from other people's code.
- Good team worker.
- The student shows enthusiasm, motivation.

Assessing the product is a widespread way of grading students, but, the process criteria seem to be more useful for formative purposes. Based on the results of product and process criteria, we found strong similarities with the work of Lars Lindström (2006) who designed an instrument to assess creativity in Art education. In programming there are creative elements, just as in technology education and in several other school subjects. Art is a subject where creativity is central, and previous assessment research from that field is of interest when assessment in programming is explored. In a large study in Art education about creative development, criteria or descriptions of general abilities used by teachers to assess creative work were examined (Lindström, Ulriksson, & Elsner, 1998). Using a literature review and interviews, a list of important factors/criteria for the development of creativity was formulated, and criteria were divided in three groups:

Product criteria:

1. Visibility of the intention/ Goal fulfillment
2. Color, form and composition/ Visual qualities
3. Craftsmanship/ Technical skill

Process criteria:

4. Investigative work/ Persistence in the pursuit
5. Inventiveness/ Imagination and risk-taking
6. Ability to use models/ being able to learn from others
7. Capacity for self-assessment/ knowing one's strengths/weaknesses

Others:

8. Overall judgment

The four process criteria found by Lindström (2006) to be important in the development of creativity and problem-solving skills could be explained by a psychological model, the dual system model (Björklund, 2009), in the following way:

- Ability to use models: The student is building a library, partly explicit, partly implicit of solutions to specific problems, a base of patterns that later can be used in a pattern recognition process. This is one part of the concept of computational thinking.
- Investigative work: This is the exploratory phase when the student is trying to find a perspective, a viewpoint from which stored patterns may match the problem.
- Inventiveness: A friendly, forgiving, context is important because the database of learned examples residing in system 1 is sensitive to stress access to implicit memories feasible.
- Capacity for self-assessment, knowing one's strengths/weaknesses: A process where patterns are assessed and given a somatic marker and stored in implicit memories.

Lindström's criteria have been transferred and used in technology education contexts, e.g. (Björklund, 2009). By using our results from the interviews, we adopted and transformed Lindström's instrument to programming:

Product criteria:

1. Visibility of the function/ Goal fulfillment
2. Use of programming language, knowledge of syntax
3. Program quality, coding skill, and documentation

Process criteria:

4. Investigative work/ Persistence in the pursuit
5. Inventiveness/ Imagination and risk-taking
6. Ability to use models/ being able to learn from others
7. Capacity for self-assessment/ knowing one's strengths/weaknesses

Discussion

The results showed two types of criteria used by the teachers; product criteria assessing the program and process criteria assessing the ongoing process and the students "studio habits of mind". This somewhat contradicts results from Simon (2012) that examiners emphasised product criteria when they assessed programming skills. We compared the criteria we found with Lindström's instrument measuring the development of creativity. Our preliminary results make us believe that some sort of process criteria could be used (and indeed seemed to be used by teachers) to assess students struggling with programming tasks.

The rubrics in every proposed criterion could be used in a formative way to lead the student forward in their development of programming skills. A tentative description of the process criteria could be useful in programming education and some examples of rubrics follow below.

Proposed criteria to assess the process of programming

Investigative work/ Persistence in the pursuit

The student tackles and completes the work with stubbornness and patience. An ability to develop and stick to a theme or problem over time.

1. The student gives up easily, doesn't complete any ideas and only does what the teacher asks for.
2. The student shows some patience, tries out her/his own solutions and approaches but doesn't develop them further.
3. The student puts in a great effort, approaches themes and problems from several perspectives and develops the program through a sequence of drafts, sketches and try outs. (Trial and error)

Inventiveness/ Imagination and risk-taking

To dare to take risks and try out own solutions and ideas.

1. The student does not formulate problems on her/his own and doesn't experiment with sensors, actuators or tries different algorithms to reach the goal.
2. The student sometimes formulates problems of her/his own or slightly reformulates some of the problems stated by the teacher. Shows tendencies to experiment and try out her/his own ideas.
3. The student often states her/his own problems or reformulates the problems stated by the teacher. The student moves on, makes experiments and is willing to take risks. The student often finds unexpected solutions.

Ability to use models/ Being able to learn from others

To search for models and find relations between models and their own work.

1. The student doesn't show interest in solutions made by others and can't take advantage of them even when the teacher helps to find them.
2. The student makes some active efforts to find models for her/his own work, shows some interest in other people's solutions, but she/he confines them self to copying them.
3. The student is active to find different kinds of examples of programs and uses them in a diverse, independent and well-integrated way in their own work.

Capacity for self-assessment/ knowing one's strengths/weaknesses

A capacity for self-assessment is not innate, it is something that students can develop and refine.
A student with a high ability to evaluate their own work can point out works or parts of works that are successful or that require continued work.

1. The student can't identify strong and weak aspects in their own work, neither distinguish between good or bad programs. The student has no opinions of the class mate's programming work.
2. The student can, with some assistance, identify merits and shortcomings in their own work and also distinguish between good or bad programs. The student begins to produce qualified judgements of peer's work.
3. The student sees merits and shortcomings in their own work and can identify examples of programs and work that shows their own development. The student can motivate and explain why the result turned out in a specific way. The student can give nuanced reviews of classmates' programs and contribute with constructive feedback.

Further research

A web based survey using a Q-sort method will be sent to a number of teachers to find more robust data on what teachers do use when they are assessing programming skills.

References

Barefoot, C. (2014). Computational thinking [web page], from https://www.barefootcomputing.org/

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Inroads, 2*(1), 48-54.

Björklund, L. (2008). The Repertory Grid Technique: Making Tacit Knowledge Explicit: Assessing Creative work and Problem solving skills. In H. Middleton (Ed.), *Researching Technology Education: Methods and techniques*. Netherlands,: Sense Publishers.

Björklund, L. (2009). *The forming and assessment of creative skills, from a neurocognitive point of view.* Paper presented at the Patt-22, Delft, the Netherlands.

Dagiené, V., & Stupurienė, G. (2016). Bebras - a Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education, 15*(1), 25-44.

Jones, A., Buntting, C., & de Vries, M. (2013). The developing field of technology education: a review to look forward. [Article]. *International Journal of Technology & Design Education, 23*(2), 191-212. doi: 10.1007/s10798-011-9174-4

Jones, I., & Alcock, L. (2014). Peer assessment without assessment criteria. *Studies in Higher Education, 39*(10), 1774-1787.

Kelly, G. A. (1955). *The psychology of personal constructs Vol 1 A theory of personality Vol 2 Clinical diagnosis and psychotherapy*. Oxford: W. W. Norton.

Lindström, L. (2006). Creativity:What Is It? Can You Assess It? Can It Be Taught? *iJADE, 25*(1), 53-66.

Lindström, L., Ulriksson, L., & Elsner, C. (1998). *Portföljvärdering av elevers skapande i bild: utvärdering av skolan 1998 avseende läroplanernas mål (US98)*: Skolverket.

Ministry_of_Education. (2018). *Technology in the New Zealand Curriculum*. Retrieved from http://nzcurriculum.tki.org.nz/The-New-Zealand-Curriculum/Technology.

Moreno-León, J., Robles, G., & Román-González, M. (2016). Examining the Relationship between Socialization and Improved Software Development Skills in the Scratch Code Learning Environment. *Journal of Universal Computer Science, 22*(12), 1533-1557.

Pollitt, A. (2012). Comparative judgement for assessment. *International Journal of Technology and Design Education, 22*, 157-170.

Simon, Chinn, D., de Raadt, M., Philpott, A., Sheard, J., Laakso, M.-J., et al. (2012). *Introductory programming: examining the exams.* Paper presented at the Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123.

Skolverket. (2017). *Läroplan för grundskolan, förskoleklassen och fritidshemmet 2011 (Reviderad 2017)*. Stockholm: Skolverket.

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850-858.

Straw, S., Bamford, S., & Styles, B. (2017). *Randomised Controlled Trial and Process Evaluation of Code Clubs*: National Foundation for Educational Research.

Svennberg, L., Meckbach, J., & Redelius, K. (2014). Exploring PE teachers' 'gut feelings': An attempt to verbalise and discuss teachers internalised grading criteria. *European Physical Education Review, 20*(2), 199-214.

Wing, J. M. (2006). Computational Thinking. *Communication of the ACM, 49*(3), 33-35.

Charlotta Nordlöf, Linköping university (Sweden), is a PhD student in Technology education. Her main research interest is technology teachers — their attitudes to technology education and their views on technology knowledge. Her background is technology teaching at upper secondary school.

Lars Björklund, Linköping university (Sweden), is a PhD and a senior lecturer in Science and Technology education. His main research interest is assessment and development of expertise.