Exploring Students' Computational Thinking Skills in Modeling and Simulation Projects: a Pilot Study

Nataša Grgurina
University of Groningen
The Netherlands
n.grgurina@rug.nl

Erik Barendsen
Radboud University, Open University of NL
The Netherlands
e.barendsen@cs.ru.nl

Klaas van Veen
University of Groningen
The Netherlands
klaas.van.veen@rug.nl

Cor Suhre
University of Groningen
The Netherlands
c.j.m.suhre@rug.nl

Bert Zwaneveld
Open University of NL
The Netherlands
g.zwaneveld@uu.nl

ABSTRACT
Computational Thinking (CT) is gaining a lot of attention in education. We explored how to discern the occurrences of CT in the projects of 12th grade high school students in the computer science (CS) course. Within the projects, they constructed models and ran simulations of phenomena from other (STEM) disciplines. We examined which CT aspects occurred in students' activities and how to assess students' CT accomplishments. For this purpose we employed a framework based on CT characterizations by Wing [14, 15], CSTA [4] and Comer et al. [3]. We analyzed students' project documentation, survey results and interviews with individual students. The findings indicate that this framework is suitable for detection of occurrences of CT aspects in students' data. Moreover, our preliminary results suggest that the framework is useful in assessment of the quality of the students' CT performance.

CCS Concepts
• Social and professional topics~Computational thinking

• Computing methodologies~Modeling and simulation

Keywords
Computational thinking; modeling; simulations; computer science education; students' learning.

1. INTRODUCTION
Computational Thinking can be described as a set of mental activities that comprises the decomposition of open-ended problems and the construction and evaluation of models that simulate the nature of these problems in order to be able to provide solutions to those problems. To support the development of CT, teachers need to be familiar with adequate learning activities and specific learning difficulties students encounter while learning CT. They also need adequate assessment instruments to assess the quality of students' CT skills and monitor their development.

In this study we explore the assessment of CT skills, specifically those concerning modeling and simulations. Previous efforts to assess CT skills focus mostly on other CT aspects such as algorithmic thinking or programming. Nevertheless, they suggest that a broad approach utilizing several data sources is needed to comprehensively assess the students' development and achievements concerning CT skills. We focus on the following two questions: which aspects of CT are brought up in CS students' projects and how can we assess the students' CT accomplishments in the collected data?

We report on an ongoing case study on a project-based lesson unit within a regular CS course in the 12th grade of high school where students studied Modeling and Simulations. They programmed models of phenomena from other (STEM) disciplines and employed the scientific method to explore these phenomena through simulations. We gathered data from various sources and analyzed it for both occurrences and quality of CT aspects.

This study is a part of a larger research project on CT. In the first phase of the project, we refined the CSTA definition of CT [6] and explored teachers' PCK [7, 8]. The results of this study will, together with the results of previous studies, serve as input a the later study into a CT assessment instrument.

2. THEORETICAL BACKGROUND

2.1 Computational thinking
In this section we elaborate our understanding of CT. Developing models and running simulation, a core CT concept, is standard practice in scientific research. Our framework is concerned with the specific case of the use of the scientific method when experimenting means programming a model of a phenomenon and running simulations with it. The scientific method consist of the following steps: (1) asking a question, (2) doing background research into the relevant matter, (3) formulating a hypothesis, (4) testing the hypothesis through an experiment, (5) analyzing data and drawing a conclusion and finally (6) reporting the results. In our view, CT includes the steps 2 through 5. According to Wing [14, 15], CT encompasses doing background research and in the abstraction process, "deciding what details we need to highlight and what details we can ignore". Comer et al. [3] consider *theory* which is rooted in mathematics, *abstraction* (modeling) which is "rooted in the experimental scientific method" and *design* – "the

WiPSCE '15, November 09-11, 2015, London, United Kingdom
 ISBN 978-1-4503-3753-3/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2818314.2818325

construction of a system (or device) to solve a given problem" to be the three major paradigms of computing. Using CT in the scientific method therefore identifies step (2) as belonging to CT. Steps (3) and (4) find their interpretations as a chain of steps in the two paradigms *abstraction* and *construction.* The *abstraction* is situated in the context domain where the problem originates while the *design* deals with the CS concepts, see Figure 1.

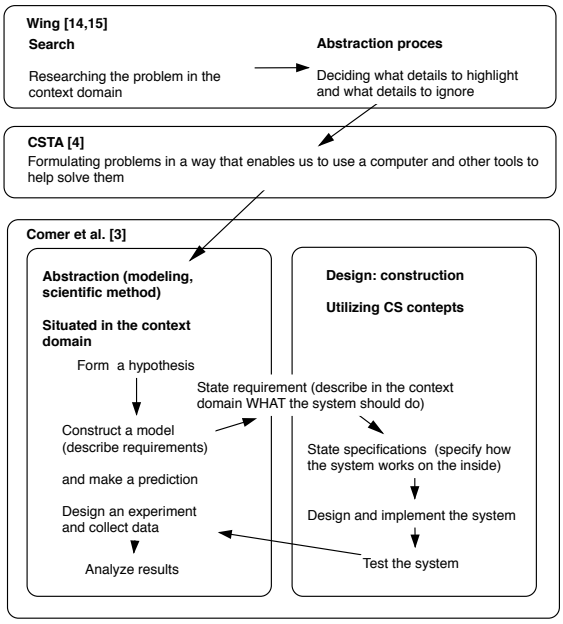

Figure 1. Theoretical framework

2.2 Assessment

A number of assessment instruments for CT (CS) have already been proposed to satisfy a range of needs and suit the particular circumstances. In this section we give a short overview of some of these efforts, ranging from tests with open and closed questions through examination of programming projects to interviews with students and teachers.

In order to "create a profile of students' abilities in computational thinking" and "to determine whether there may be a link between computational thinking performance and success within the introductory computer science course" at their university, Gouws, Bradshaw and Wentworth [5] designed a test containing closed questions from the Computer Olympiad 'Talent Search'. The questions were categorized into the six distinct classifications of CT they recognized and focused on students' abilities in the lower three levels of Bloom's revised taxonomy.

Tests with programming tasks and/or open questions were designed by Werner, Denner, Campe and Kawamoto [13] and Meerbaum-Salant, Armoni and Ben-Ari [11] who were interested in assessing CT performance in middle school students "in order to develop and strengthen efforts to engage K-12 students in CT" [13] and "to investigate if Scratch can be used to teach concepts of computer science." [11] Both these assessment instruments were developed with relevant CT categories in mind. While Werner et al. use Linn's three-link chain metaphor [10] to describe the different cognitive demands in their assessment, Meerbaum-Salant et al. felt no existing taxonomy suited their needs and decided to combine the Revised Bloom Taxonomy and the SOLO taxonomy. Furthermore, they observed students during their work and interviewed their teachers.

A different approach to CT assessment was employed by Grover [9] who interviewed children in order to "to measure elements and dimensions of computational thinking verbally expressed" before and after a week long workshop. To analyze the interviews, "a coding scheme was developed to refine dimensions of CT into taxonomic categories which represented different types of ideas in realm of computing".

Similarly to Werner et al. and Meerbaum-Salant et al, Brennan and Resnick "are interested in the ways that design-based learning activities […] support the development of computational thinking in young people" [2] and explore three approaches to assessment of the development of CT of the children engaged in such activities. They use a suitable CT framework to analyze (1) online project portfolios containing Scratch projects made by young people, (2) artifact-based interviews where Scratchers talk about their projects and, finally, (3) design scenarios where children were asked to comment on and change existing Scratch projects. They discuss strengths and limitations of each of these approaches extensively and subsequently advocate a comprehensive approach to assessment. They list six suggestions for assessing computational thinking via programming and in conclusion they express their hope "that others will take these suggestions, as well as our three example approaches, and remix them to create new forms of assessment."

We agree that a comprehensive approach combining analysis of several data sources is a promising approach for our exploration of the quality of students' work on their projects and we acted accordingly in this study.

3. METHODOLOGY

The data was collected by the first author within the regular CS course she teaches. For the 12th grade students this was the last part of their three-year CS course. During a six-weeks period they studied Modeling and Simulations with NetLogo. The first three weeks were dedicated to studying the textbook material. During the rest of the period, the fourteen students comprising this class worked in seven groups on a practical assignment where they applied the scientific method to investigate a phenomenon of their choosing by making a model in NetLogo and exploring it through running simulations. Where necessary, students were assisted in formulating their hypotheses or research questions. The entire process was strictly planned and contained milestones when the students turned in the required project documentation and kept logbooks.

At the end of the period, each group presented its model to the rest of the class and the students were encouraged to discuss their models, results, design choices, programming issues and other relevant questions. A few days later, twelve students (comprising six groups) turned in their final reports and NetLogo programs. After receiving their grades, they were asked to fill in an online questionnaire individually (twelve students did) and invited to be interviewed. Five semi-structured interviews were conducted with individual students. The students were requested to describe their projects and they were asked if they could design a new NetLogo model on the fly (ie. draw a sketch of the interface on paper and describe the model in terms of agents and interactions). Finally, they were asked what they learned during their work on the projects. The interviews were recorded and transcribed verbatim.

We performed a qualitative analysis of the project documentation, survey results and interviews. We used the categories from our theoretical framework (see Figure 1.) as coding categories and looked for quality indicators for the students' accomplishments.

4. RESULTS

Here we present a limited overview with some examples of our results. The categories from our framework are printed in italics.

There were six successful projects that all contained all the steps described in our theoretical framework. The group that did not finish the project failed to make the transition from *abstraction* (*stating requirements*) to *stating specifications* and *designing and implementing* their model.

As to the quality of the steps, one group reported making an unrealistic model due to overly simplified *abstraction* of their phenomenon, so they were not able to draw the desired conclusions. Another group reported not being able to implement all the desired behavior of their model because they could not come up with the necessary algorithms (*specifications*).

In this preliminary report we discuss an exemplary pair of students, Sue and Cate (not their real names), in more detail by describing the data obtained from their project documentation, survey and interviews.

4.1 Project documentation

After looking around for a suitable phenomenon to model with NetLogo, Sue and Cate decided to explore the spreading of the Ebola disease and investigate what effect would an Ebola medicine have on it. They began with *researching the problem in the context domain* and in the process of the *abstraction* they wrote:

- *"Virus: does not spread through the air but through contact with an Ebola patient (sex, blood), slaughtering and eating of a sick animal, non-sterile needles*
- *Mainly in Western Africa, incubation about 21 days, 9 out of 10 people die"*

Then they translated the relevant aspects into *requirements* describing what the system should do: "*In our model we are going to work with a new medicine that can cure 50% of the infected people. A person can be either ill or healthy but cannot become immune. Ill persons can pass on the disease to healthy persons."*

In the *specifications* they wrote: "*In our model there is only one [breed of] turtles and it stands for people. These turtles can have various properties, such as being ill or healthy. They can be influenced by external factors such as medicine and their life span."* Notice that the distinction between requirements (describing what the system should do in terms of context domain) and specifications (describing how the system should work in programming terms) is blurred.

They went on to *design and implement the system*:

```
to rip
        if age > lifespan
         [die]
        if  (no medicine after incubation period)
        [die]
end
...
determine the incubation period
                (if no medicine on time ? rip)
        slider
```

Notice the incremental development of the program – there is some pseudo-code (in italics) that is written using the terminology of the context domain.

They finished the program, *tested* it (i.e. debugged) and went on to *design experiments and collect data* (i.e. run simulations). Their *analysis* revealed: "*We expected that the new medicine would decrease the spreading of Ebola. It turned out that the medicine worked rather quickly, but that the rate of infectiousness was of influence as well."*

4.2 Survey

Sue commented on the practical assignment: "*... it was quite difficult to figure out what could or could not be modeled ... "*

And after finishing the practical assignment: "*I found it really interesting that you could change different things and could observe immediately what effect that had. Of course, that goes with modeling, but I still enjoyed doing it very much."*

When asked what did she learn about the phenomenon, she replied: "*There was a vaccine with efficiency of 1 to 100 that was sufficient to keep the population alive [she means: prevent extinction] and a vaccine with efficiency of 30 or 40 to 100 was really efficient."*

And further she learned: "*... that you need to have a very clear picture of what you want to make and how you want to make it, concerning characteristics and such. And also that it is important to know what is possible. ... How to put all the pieces together (observer part as well as turtles and such)."*

Cate learned: *"It is a good means to predict/research hypotheses. A good aid for research. I take chemical reactions as an example. You can make it and thus see (visualize) what happens."*

4.3 Interviews

During the interviews the students were asked to design a new NetLogo model. Both students followed Chemistry class and they both chose to model polymers. Cate drew a few small circles and narrated: "*Suppose this is all water and they all have a rim around them so that if this one reacts with that one [she points to the little circles in her drawing], then one of them sticks to it, then you already got something [she draws four little circles sticking to each other] and you let it go on like this, I don't know how could you do it, but in Chemistry you can always let things go faster or direct them, so that you can let it happen faster that all monomers stick together as a polymer."*

Sue described her model of polymers in similar terms, talking about *A's* and *B's* and describing the factors influencing the reactions in great detail. When asked about the potential benefits of making a model for a known chemical reaction, she explained: "*So, that is why it would be interesting to consider for example the industry which needs a particular polymer, how would these polymers react with each other and how they stick together, so that you can see in your model, hey, there is a bigger chance that this gets created rather than that, and we better use this for the reaction - or something like that."*

The interviewer noticed that Cate integrated her programming skills into her thinking as a chemist and remarked to her that she only spoke in terms of monomers, catalysts and circumstances favorable for their interactions (ie. chemical reactions) but never mentioned programming, code, procedures or other CS concepts. Cate replied: "*But code is basically another language for what is written here. This is Dutch and you can convert that into code."*

5. CONCLUSION AND DISCUSSION

We observed that all students who turned in their assignments succeeded in programming their code. When talking about their projects, they persistently used the language of the context domain. While not all the projects were of the same outstanding quality as Sue and Cate's, they all demonstrated that the students were capable of finding their own problems in other disciplines, making a model, running simulations and deepening their knowledge and understanding of the phenomenon they modeled.

Our first research question was, which aspects of CT were brought up in these projects. In successful projects we observed that all of the aspects in the framework were present. In the case of the unsuccessful project, we were able to pinpoint what went wrong in terms of steps in the framework.

Regarding our second research question about assessment, we find that combining students' project documentation, survey results and interviews based on our framework allowed us to distinguish various degrees of quality of students' accomplishments.

We believe our framework is suitable to explore CT aspects related to "formulating problems in a way that enables us to use a computer and other tools to help solve them" [4] and as such it is a valuable contribution to explore occurrences of CT in work of advanced CS students. Brennan and Resnick also suggest a comprehensive approach to assessment which combines analysis of several data sources [2] and we intend to explore this avenue of research further. In subsequent research we are going to explore the efficiency quality of individual steps through the analysis of screen and voice recordings of students during their work on the projects.

Several students told us that through work on this project, they learned about the phenomena they modeled. This is in line with the findings reported by Blikstein and Wilensky who found that modeling with NetLogo contributed to students' understanding of the phenomena in material science education [1]. We agree with Taub et al. who believe "that the discipline of CS has the potential to positively affect learning of other scientific disciplines" [12]. We see that these CS students seem to be able to utilize their CS/CT knowledge and skills to advance their learning in other (STEM) disciplines.

The results of this research will contribute to the development of (1) suitable learning activities both within the CS courses as elsewhere and (2) knowledge about monitoring and assessment of CT. They will contribute to the development of the CS curriculum in secondary education in the Netherlands, CS teacher training and CS education in general.

6. ACKNOWLEDGEMENTS

This work is supported by the The Netherlands Organisation for Scientific Research grant nr. 023.002.138.

7. REFERENCES

[1] Blikstein, P. and Wilensky, U. An Atom is Known by the Company it Keeps: Content, Representation and Pedagogy within the Epistemic Revolution of the Complexity Sciences. (2009).

[2] Brennan, K. and Resnick, M. New Frameworks for Studying and Assessing the Development of Computational Thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, 2012.

[3] Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., Young, P. R. and Denning, P. J. Computing as a Discipline. Commununocations of ACM, 32, 1 (1989), 9-23.

[4] CSTA Computational Thinking Task Force. Operational Definition of Computational Thinking for K–12 Education. 2013, 10/16 (2011).

[5] Gouws, L., Bradshaw, K. and Wentworth, P. First Year Student Performance in a Test for Computational Thinking. In Anonymous *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference.* (East London, South Africa,). ACM, New York, NY, USA, 2013, 271-277.

[6] Grgurina, N., Barendsen, E., Zwaneveld, B., van de Grift, W. and Stoker, I. Computational Thinking Skills in Dutch Secondary Education. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education.* ACM, 2013, 31-32.

[7] Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K. and Stoker, I. Computational Thinking Skills in Dutch Secondary Education: Exploring Pedagogical Content Knowledge. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research.* ACM, 2014, 173-174.

[8] Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K. and Stoker, I. Computational Thinking Skills in Dutch Secondary Education: Exploring Teacher's Perspective. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education.* ACM, 2014, 124-125.

[9] Grover, S. Robotics and Engineering for Middle and High School Students to Develop Computational Thinking. In *annual meeting of the American Educational Research Association, New Orleans, LA*, 2011.

[10] Linn, M. C. The Cognitive Consequences of Programming Instruction in Classrooms. Educational Researcher, 14, 5 (1985), 14-29.

[11] Meerbaum-Salant, O., Armoni, M. and Ben-Ari, M. Learning Computer Science Concepts with Scratch. Computer Science Education, 23, 3 (2013), 239-264.

[12] Taub, R., Armoni, M. and Ben-Ari, M. M. Abstraction as a Bridging Concept Between Computer Science and Physics. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education.* ACM, 2014, 16-19.

[13] Werner, L., Denner, J., Campe, S. and Kawamoto, D. C. The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* (Raleigh, North Carolina, USA,). ACM, New York, NY, USA, 2012, 215-220.

[14] Wing, J. M. Computational Thinking and Thinking about Computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366, 1881 (2008), 3717.

[15] Wing, J. M. Computational Thinking. Communications of ACM, 49, 3 (2006), 33-35.