CENTRE FOR SCIENCE EDUCATION
SCIENCE AND TECHNOLOGY
AARHUS UNIVERSITY

25/10/13

Model-Based Thinking & Practice

A white-paper exploring a Future Direction of CS Education Research

Michael E. Caspersen & Palle Nowack
Centre for Science Education, Aarhus University
{mec, nowack}@cse.au.dk

1 Introduction

During the last 50 years many attempts have been made to broaden the participation in computer science. In our opinion, one of the less fortunate results of this, was when computer science curricula in many places in Primary and Secondary Education was replaced by curricula enforcing (more or less basic) ICT-skills. The creative producer had become the streamlined user – the active participant had become the passive spectator. This problem has now been widely recognized, and what has been dubbed "Computational Thinking" [1] has had a revival in the CS Education Community. However, 7 years after Jeannette Wing's paper (and 17 years after the initial remark by Seymour Papert[1]), we are still discussing, what Computational Thinking actually means; especially in the classroom.

Computational Thinking appears to focus on CS concepts in relation to a process of problem solving such as: pattern recognition, pattern generation, abstraction (composition, de-composition, generalization, specialization), modeling, algorithm design (sequence, iteration, selection), data analysis, and visualization. This is a very broad selection of fundamental concepts, and it can be specialized and implemented in many different ways.

In this paper we propose one such possible specialization and concretization of Computational Thinking, which we find not only broadly applicable, but also motivating and rewarding for potentially a much larger group of pupils and students, than the one currently targeted by CS Education. Naturally, we don't claim that it is an overarching research question for the entire CS Education field, but we strongly suggest, that it becomes one of the future research directions.

We focus on the teaching of CS for pupils within the K-12 segment. That is, we focus on knowledge and skills, which we find generally useful at the same level as basic reading, writing and arithmetic.

Our initial research question is the following:

- *How can model-based thinking & practice leverage the teaching of CS principles and practices to pupils in schools?*
    - *By clearly demonstrating how CS provides the means for a radically different and rewarding way of thinking, which is applicable in many aspects of human life.*

[1] http://www.papert.org/articles/AnExplorationintheSpaceofMathematicsEducations.html

- *By empowering pupils with new practices which enables them to create radically new products, processes and organizations – unthinkable without CS.*
- *By broadening participation in CS.*
- *By integrating efficiently with other subject areas, without being reduced to basic ICT-skills and application of tools.*

The paper is structured as follows: In Section 2 we argue why model-based thinking & practice is an important field to develop and teach; In Section 3 we explore what to teach, as we try to characterize model-based thinking & practice; and finally Section 4 summarizes our ideas.

2 Why Model-Based Thinking & Practice?

Mental models are part of almost all human endeavors. We form, share, change, evolve and use such models in our private lives, in our lives as citizens in communities, and in our working lives. We use them to understand ourselves as well as the world around us.

Basically we use models in order to be able to:

- Analyze and understand phenomena.
- Design and construct artifacts.

Working with models accomplish this in two different, but complementing, ways:

- By enabling us to abstract away from (in this particular situation) unimportant details, and to emphasize essential properties of the phenomena corresponding concepts we are considering (thus reducing complexity).
- By enabling us to experiment with multiple (sometimes contradicting) conceptualizations of the same phenomenon, which is the basis for an iterative and incremental way of working: stepwise improvement (thus reducing uncertainty)

With the proliferation of computers and the Internet, many of these models have become explicit. They are represented (more or less explicitly) in the systems we use, and the systems (because of these models) govern how we perceive the world, how we think about it, and how we act in it. They enable us to do things unimaginable just a few years ago, but they also limit our understanding and possibilities for action. Sometimes (often?) the system-embedded models directly contradict our own intuitive mental models. This causes confusion, aggravation, inefficiency, and errors.

In addition to the above-mentioned basic benefits of models in general (which are important in their own rights), we use computerized models, in order to make the models:

- Dynamic (e.g. Simulations).
- Visual (often in combination with the above, e.g. animations).
- Interactive.
- Explicit (as opposed to mental, which are hard to share).
- Distributable and shareable (e.g. using cloud-based services).

- Persistent (as opposed to whiteboards and lectures).
- Scalable (i.e. we use the computer to change the level of detail of our models).
- Rule-based (i.e. the computer enforces certain invariants, e.g. physical laws, are maintained, when manipulating the model).

As computer professionals we play a crucial role in the development and use of systems and models. But the impact is much more profound and essential: if you don't understand the models, if you don't play an active part in their creation, you become reduced to a user of computerized models: i.e. an end-user of software applications. As opposed to an active, creative creator of models and systems, and in turns thus contributing with the creation of new types of organizations, processes, and products.

When object-oriented programming and OO analysis & design emerged in the eighties and nineties, two different schools of thought crystalized: some saw objects primarily as a great mechanism for modularizing, reusing and combining software elements, whereas others primarily focused on the fact, that objects and classes could represent phenomena and concepts from a problem domain (banking, electronics, user-interfaces, etc.)[3, 4], thus changing the entire software development process from requirements specification over analysis and design to implementation, making it easier to involve end-users and clients in a dialogue about the systems. Today we clearly acknowledge both schools of thought as very valuable and complementing perspectives on software development. However, whereas the former perspective is extremely useful for computing professionals (e.g. programmers and developers), we believe that in the latter approach, we see a lot of unfulfilled potential, when it comes to teaching basic computing skills.

The computer is an excellent and unique tool when it comes to using, changing and creating models of phenomena and concepts from the real or the imaginary world (e.g. games). On the other hand, the computer is almost useless when it comes to understanding and formulating the problems to be dealt with, and to attribute sense to the results of the computations. For this we need people with insight into domains and problems. But in order for this to become a truly efficient combination, we need to leverage the basic understanding of computerized models.

3 What is Model-Based Thinking & Practice?

As computer professionals we have all had the experience of working with people from other fields of expertise, and we have all experienced the power of being able to translate their ideas and thoughts about their field of practice into powerful, interactive, and efficient software systems, which provides value for them. As computer professionals, we find it very natural to think in terms of models, when dealing with new and unfamiliar domains. We have experienced how it leads to more informed discussions and actions regarding the field of interest. Instead of hand waving about abstract ideas and thoughts, we can actually discuss explicit representations of these ideas and thoughts:

- Is this a "good" and/or "correct" model of the situation? (Often leading to a clarification about qualities of the situation in the domain as opposed to qualities of the model itself)

- How can we improve the situation/model?
- How about different models explaining different aspects of the complex situation (dynamics, statics, structure, values, etc.)? (Often leading to a realization, that typically multiple perspectives (and corresponding models) are often called for to reduce complexity)
- What happens if we do this and that to the model (and thus the situation)?

Being able to understand (some) models, explain them, design them, implement them in software, modify them, and evaluate them is a very powerful competence. It is this competence that we think should be a basic competence at the same level of importance as reading, writing, and arithmetic.

This must be not be confused with the elaborate use of rigid models in e.g. software engineering, e.g. UML diagrams in high-ceremony development processes etc. In our understanding this level of modeling is not very relevant for teaching computing in e.g. K-12 education. Neither must is be confused with the (rather simple) use of computing as a mere tool for investigating an area, e.g. using an application, where somebody else has already designed the model (and the domain of possible models) and buried it underneath functionality and user-interface features.

In our understanding the code itself is a model (explicitly of a program execution, and implicitly of a problem and an application domain)[2], but we have yet to develop abstractions, modeling techniques, languages and tools suitable for teaching the essential modeling skills to a broader collection of pupils and students.

We believe that all pupils should be better at:

- Understanding computer-based models.
- Formulating problems, which can be transformed into a model, which can be represented in and manipulated by a computer.
- Manipulate (change, evolve, interact with) computer-based models.
- Create computer-based models.

This must be understood in the light of the ever-moving border between what the computer can do, and what humans must do. One of the major problems with the current focus on ready-made ICT skills in primary and secondary education is that it is based on a fixed static picture of technology – and not the longer-lasting principles of computation, systems thinking, development, and modeling.

However, it is obvious, that this is not going to happen with the current state-of-the-art elements of computer science (tools, languages, models and processes) as they have been developed and evolved for a quite different purpose. Nevertheless, we see a clear line of progression with respect to emphasizing and focusing on the relation between mental models and computerized models in many tools for teaching, e.g. in the works of Michael Kölling and others with the BlueJ[2] and GreenFoot[3] environments. This is being further pursued in the development of a notional machine for introductory programming [5].

[2] http://www.bluej.org

A particular fascinating and promising aspect of a model-focused approach to computing is the many ways it can be integrated with other subject areas. The heavy use of models (and computerized models) is evident in Science subjects (physics, chemistry, biology, etc.) and the Social Sciences, but also in the Liberal Arts, models are abundance, e.g. in relation to music, text analysis, etc. [6].

4 Conclusion

Model-based thinking is part of many human endeavors - especially in relation to education. Computerized models are powerful tools for creating new organizations, processes, and products, because computers and software directly support model-based thinking by making models explicit, tangible and interactive. However, the current level of maturity of computing clutters the understanding of this with extra/incidental (not inherent domain-related) complexity. The current fix is to hide the complexity under layers of functionality and user-interfaces, which creates a huge gap between the people who create with computing, and the people who consumes with computing.

We propose that a strong focus on the relation between mental models (of real or imaginary systems) and computerized models (embedded in computer-based systems) will open up a new field in CS education, which will result in new tools (languages, systems), models, and processes for teaching CS. This approach should clarify and make explicit the role of models in computing in connection with other subject areas. We believe that such an approach would strongly broaden the participation in CS, as it will allow more pupils to become active creators with computing.

5 References

[1] Wing, J. (2006): Computational Thinking, *Communications of the ACM*, 49(3):33-35.

[2] Madsen, O.L., Møller-Petersen, B., and Nygaard, K., "Object-Oriented Programming in the BETA Programming Language", Addison-Wesley/ACM Press, 1993.

[3] Bennedsen, J. and Caspersen, M.E.: "Programming in Context — A Model-First Approach to CS1", *Proceedings of the thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE 2004, Norfolk, Virginia, USA, 2004, pp. 477-481.

[4] Bennedsen, J.B. and Caspersen, M.E.: "Model-Driven Programming", *Reflections on the Teaching of Programming*, LNCS 4821, Springer-Verlag, 2008, pp. 116-129.

[5] Berry, M. and Kölling, M.: “The design and implementation of a notional machine for teaching introductory programming”. To appear in the Proceedings of The 8th Workshop in Primary and Secondary Computing Education (WIPSCE 2013), Aarhus, Denmark, 2013.

[6] Guzdial, M.: “Exploring hypotheses about media computation”, Proceedings of the ninth annual international ACM conference on International computing education research (ICER 2013), USA, 2013.

[3] http://www.greenfoot.org