Model-Based Thinking and Practice

A Top-down Approach to Computational Thinking

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

ABSTRACT

In this paper, we discuss using models and modeling in a new way to teach basic computing to pupils within the K-12 segment. We argue why we believe understanding and creating models are fundamental skills for all pupils as it can be characterized as the skill that enable us to analyze and understand phenomena as well as design and construct artifacts. We also try to characterize the essence of model-based thinking and practice. 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) could provide a new approach to teaching computing. 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 computing, as it will allow more pupils to become active creators with computing.[1]

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computers and Information Science Education—Computer Science Education

General Terms

Experimentation, Human Factors, Languages, Theory.

Keywords

Models, modeling, teaching, thinking, practice.

1. INTRODUCTION

During the last 50 years many attempts have been made to broaden the participation in computer science. One of the latest and most promising approaches is computational thinking: **“**Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” [9]. Computational thinking involves thinking in terms of recursion, parallel processing, interpretation, generalization, naming schemes, correctness, efficiency, aesthetics, abstraction, decomposition, separation of concerns, representations, models, invariants, modularization, caching, planning, learning, scheduling, and much more [27].

This is a very broad selection of fundamental concepts, and it can be conceptualized and implemented in the classroom in many different ways. In Figure 1, we provide an attempt to characterize IT in education. “Basic ICT skills” and “ICT and learning” are in our opinion not relevant in relation to computational thinking, so in the remainder of the paper we focus on IT as a subject versus IT in subjects (“What to learn” in the illustration). Computational thinking lends itself very much to the use of IT in other subjects, whether in subjects as we know them today (journalism, economics, chemistry, etc.) or as a defining technology for transforming and innovating subjects (e. g. digital journalism or bioinformatics). But it appears that both these (in-subject) applications of computational thinking require a reformulated (more general) IT-subject (in the sense IT-as-a-subject) and not computer science as such. Hence in order to broaden the participation in computing, we advocate broadening the computing subject itself.

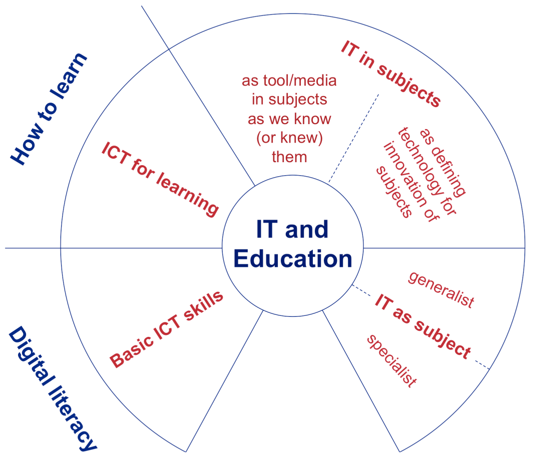


Figure 1. IT and Education.

In this paper we describe one direction in which to search for a new and broader computing subject based on computational thinking. We propose to focus on the use of models and modeling, both in order to benefit from a strong tradition in computing, but also to build bridges to a wide range of other subjects. We focus on the teaching of computing 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 mathematics.

[1] An early version of this paper was submitted and accepted as a white paper for the "Future directions in Computing Education Research Summit" in Orlando, January 2014. The accepted white papers were not published but are available from the summit website.

Koli Calling '14, November 20 - 23 2014, Koli, Finland Copyright 2014
ACM 978-1-4503-3065-7/14/11…$15.00
http://dx.doi.org/10.1145/2674683.2674686

The paper is structured as follows: Section 2 briefly touches upon related work, Section 3 defines our take on modeling, Section 4 argue why this is an important area, and finally Section 5 summarizes the paper.

2. RELATED WORK

Computational thinking has been the topic for many contributions over the last few years.

There has been focus on the elaboration of the concept itself [10, 11, 15], surveys on how to integrate it in curriculum [8, 21], frameworks describing how to implement this integration [20], inter- and multidisciplinary perspectives [1, 12-14, 23], the extremely important and often overlooked aspect of teacher training [2, 28], didactical design principles such as applying a "use-modify-create" framework [17] and constructive alignment for teaching concurrency [6].

The computing community has extensive and sophisticated experience with the use of models and modeling. It is this strength we propose to capitalize from by defining a new approach called model-based thinking and practice. Examples include software development methodologies, modeling languages, domain-specific languages, model-driven architectures, object-oriented programming, process calculus, data descriptions, algorithm descriptions, program visualizations, and many others.

The modeling aspect has also been the focus of the computing education community: within the field of teaching object-oriented programming, there has been work from a design-by-contract perspective [5], from a learning-theoretical perspective [7], from a constructive alignment and assessment perspective [4] and a conceptual modeling perspective [3]. Furthermore agent-based modeling [22, 26] has elaborated substantially on Seymour Papert's original vision of computational thinking [19].

Common to these approaches is that they all deal with models and modeling, but they do it in different ways, at different abstraction levels, with different terminology, different conceptual frameworks, different methodologies, and different tools. The approaches have been tailored and fine-tuned to do their very specific jobs.

We believe that there would be value in developing a more general and more abstract approach to working with models and modeling. Especially for pupils and students without specialists needs and skills in computing. We believe that this could make learning many different aspects of computing more efficient as well as making the pupils use of computing in other subject areas more informed, creative and efficient.

3. WHAT IS MODEL-BASED THINKING & PRACTICE?

Building on the work of Kristen Nygaard and others [16], we define modeling as a relation between a referent system and a model system (Figure 2). The referent system is a part of the (real or imaginary) world that we choose to consider as a system from a certain perspective. The referent system contains phenomena and concepts. The model system represents phenomena and concepts from the referent systems as "realized phenomena" and "realized concept" representations. In summary, modeling is the activity of building a model system based on a referent system.

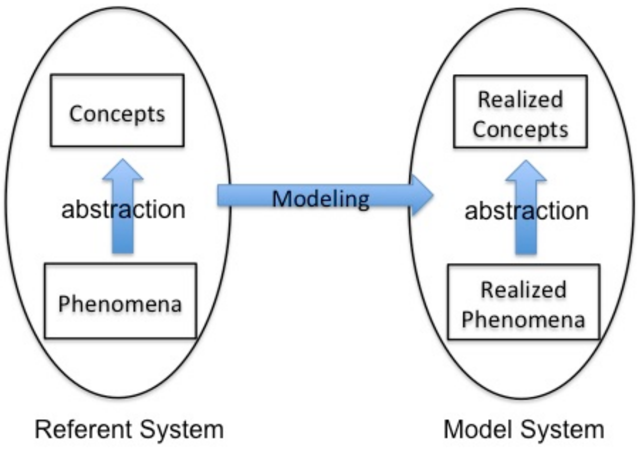
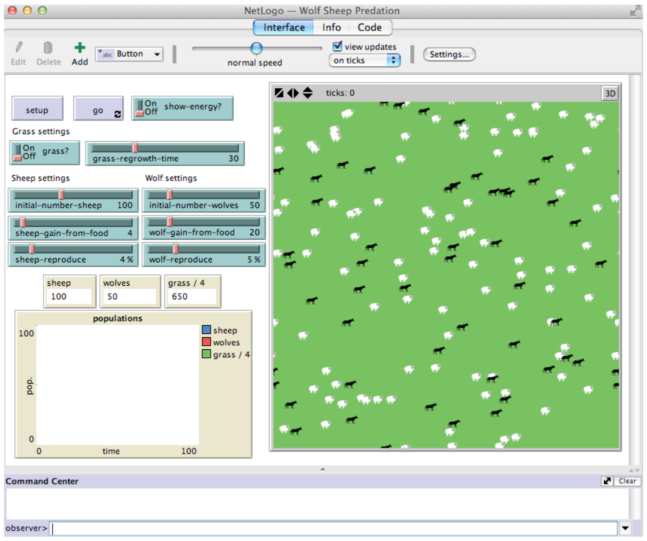


Figure 2. Modeling (Adapted from [16])

The conceptual framework is based on human concept formation processes and abstraction, and it can be used to describe all the approaches mentioned in the previous section. E.g. considering object-oriented programming, the model system consists of objects and classes, and a description of the referent system is called a domain model. Deploying model-based thinking and practice would train pupils in identifying phenomena and concepts, relate them via concept formation processes (exemplification, classification, aggregation, decomposition, generalization and specialization), finding proper representations of these, attributing meaning to models, and having discussions about the referent system in terms of the model system. Hence they would benefit from learning both about computing as well as about the particular referent system in question (typically from a different subject area).

As an example, consider the wolf-sheep simulation [24] in NetLogo [25], see Figure 3. Here, turtles and patches (in the model system) represent the wolves, sheep, and grass (from the referent system). Breeds models concept specialization, variables model concept properties etc. By letting the pupils interact with such a model system and modifying it, they learn about both the model system itself (agent-based computing) and the referent system (predator-prey eco systems).


NetLogo — Wolf Sheep Predation

grass-regrowth-time 30
initial-number-sheep 100
sheep-gain-from-food 4
sheep-reproduce 4 %
initial-number-wolves 50
wolf-gain-from-food 20
wolf-reproduce 5 %
sheep 100
wolves 50
grass / 4 650
ticks: 0

Figure 3. Example model in NetLogo [24, 25]

Model-based thinking and practice has the characteristic property that we work from the outside in. We discuss systems and models before we discuss the elements they are comprised of. As opposed to first defining bits, bytes, data, procedures, programs, etc. in a bottom-up fashion before getting to applications that are useful and meaningful to the pupils.

4. WHY MODEL-BASED THINKING & PRACTICE?

We distinguish between two reasons for developing and applying model-based thinking and practice. Firstly, it is broadly applicable. Secondly it invites for integration with other subject areas. In the following we expand on both reasons.

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 (inspired by [18]):

- By enabling us to abstract away from (in this particular situation) unimportant details, and to emphasize essential properties of the phenomena and the 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. 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 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 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)?

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.

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, i.e. we need to teach model-based thinking and practice in general.

5. SUMMARY

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 consume with computing.

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.

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) could provide a new approach to teaching computing. 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 computing, as it will allow more pupils to become active creators with computing.

6. ACKNOWLEDGMENTS

We would like to thank Bent Bruun Kristensen for interesting discussions and insightful contributions to many of the elements of this paper, as well as Lasse Højgaard and the anonymous reviewers for their helpful comments.

7. REFERENCES

[1] Amoussou, G.-A. et al. 2010. Interdisciplinary computing education for the challenges of the future. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 556-557. DOI=http://doi.acm.org/10.1145/1734263.1734449.

[2] Barr, V. and Stephenson, C. 2011. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *ACM Inroads* 2, 1 (February 2011), 48-54. DOI=http://doi.acm.org/10.1145/1929887.1929905

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

[4] Bennedsen, J. and Caspersen, M.E. 2007. Assessing Process and Product: A Practical Lab Exam for an Introductory Programming Course. *ITALICS*, Innovation in Teaching and Learning in Information and Computer Sciences, Vol. 6 (4), Special Issue on Innovative Methods of Teaching Programming, 2007, pp. 183-202.

[5] Bennedsen, J. and Caspersen, M.E. 2004. Teaching object-oriented programming-Towards teaching a systematic programming process. *Proceedings of the Eighth Workshop on Pedagogies and Tools for the Teaching and Learning of Object-Oriented Concepts*, 18th European Conference on Object-Oriented Programming (ECOOP 2004), 14-18 June, 2004, Oslo, Norway.

[6] Brabrand, C. 2008. Constructive Alignment for Teaching Model-Based Design for Concurrency. In *Transactions on Petri Nets and Other Models of Concurrency I*, Kurt Jensen, Wil M. Aalst, and Jonathan Billington (Eds.). Lecture Notes In Computer Science, Vol. 5100. Springer-Verlag, Berlin, Heidelberg 1-18. DOI=http://dx.doi.org/10.1007/978-3-540-89287-8_1

[7] Caspersen, M.E. and Bennedsen, J. 2007. Instructional Design of a Programming Course: A Learning Theoretic Approach. *Proceedings of the 3rd International Computing Education Research Workshop*, ICER 2007, Atlanta, Georgia, USA, September 2007, pp. 111-122.

[8] Caspersen, M.E. and Nowack, P. 2013. Computational Thinking and Practice — A Generic Approach to Computing in Danish High Schools. In *Proceedings of the 15th Australasian Computing Education Conference*, ACE 2013, Adelaide, South Australia, Australia, January 2013, pp. 137-143.

[9] Wing, J. M. 2010. Computational Thinking: What and Why? http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf. Accessed: 2014-10-03.

[10] Cutts, Q. et al. 2011. Computing As the 4th "R": A General Education Approach to Computing Education. In *Proceedings of the seventh international workshop on Computing education research* (ICER '11). ACM, New York, NY, USA, 133-138. DOI=http://doi.acm.org/10.1145/2016911.2016938

[11] Fletcher, G.H.L. and Lu, J.J. 2009. Education: Human Computing Skills: Rethinking the K-12 Experience. *Commun. ACM* 52, 2 (February 2009), 23-25. DOI=http://doi.acm.org/10.1145/1461928.1461938

[12] Guzdial, M. 2008. Education: Paving the Way for Computational Thinking. Commun. ACM. 51, 8 (Aug. 2008), 25–27. DOI=http://doi.acm.org/10.1145/1378704.1378713

[13] Hambrusch, S. et al. 2009. A Multidisciplinary Approach Towards Computational Thinking for Science Majors. *SIGCSE Bull.* 41, 1 (March 2009), 183-187. DOI=http://doi.acm.org/10.1145/1539024.1508931

[14] Heines, J.M. et al. 2008. Interdisciplinary Approaches to Revitalizing Undergraduate Computing Education. J. Comput. Sci. Coll. 23, 5 (May 2008), 68–72.

[15] Hu, C. 2011. Computational Thinking: What It Might Mean and What We Might Do About It. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (ITiCSE '11). ACM, New York, NY, USA, 223-227. DOI=http://doi.acm.org/10.1145/1999747.1999811

[16] Kristensen, B.B. et al. 2007. The When, Why and Why Not of the BETA Programming Language. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages* (HOPL III). ACM, New York, NY, USA, 10-1-10-57. DOI=http://doi.acm.org/10.1145/1238844.1238854.

[17] Lee, I. et al. 2011. Computational Thinking for Youth in Practice. ACM Inroads. 2, 1 (Feb. 2011), 32–37. DOI=http://doi.acm.org/10.1145/1929887.1929902

[18] Mathiassen, L. and Stage, J. 1992. The principle of limited reduction in software design. Information Technology & People, Vol. 6 Iss 2/3 pp. 171 - 185. DOI:http://dx.doi.org/10.1108/EUM0000000003550

[19] Papert, S. 1996. An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning. 1, 1 (Jan. 1996), 95–123.

[20] Perkovic, L. et al. 2010. A Framework for Computational Thinking Across the Curriculum. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (ITiCSE '10). ACM, New York, NY, USA, 123-127. DOI=http://doi.acm.org/10.1145/1822090.1822126

[21] Qualls, J.A. and Sherrell, L.B. 2010. Why Computational Thinking Should Be Integrated into the Curriculum. J. Comput. Sci. Coll. 25, 5 (May 2010), 66–71.

[22] Repenning, A. 2012. Programming goes back to school. Commun. ACM. 55, 5 (May 2012), 38–40.

[23] Soh, L.-K. et al. 2009. Renaissance Computing: An Initiative for Promoting Student Participation in Computing. *SIGCSE Bull.* 41, 1 (March 2009), 59-63. DOI=http://doi.acm.org/10.1145/1539024.1508885

[24] Wilensky, U. 1997. NetLogo Wolf Sheep Predation model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[25] Wilensky, U. NetLogo. 1999. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[26] Wilensky, U. et al. 2014. Fostering computational literacy in science classrooms. Commun. ACM. 57, 8 (Aug. 2014), 24–28.

[27] Wing, J.M. 2006. Computational Thinking. Commun. ACM. 49, 3 (Mar. 2006), 33–35.

[28] Yadav, A. et al. 2011. Introducing Computational Thinking in Education Courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 465-470. DOI=10.1145/1953163.1953297 http://doi.acm.org/10.1145/1953163.1953297