Models and Clickers for Teaching Computer Science

Matthias Hauswirth

Matthias.Hauswirth@usi.ch
http://www.inf.usi.ch/faculty/hauswirth
Faculty of Informatics
University of Lugano, Lugano, Switzerland

Abstract: Many courses in a computer science curriculum, from computer architecture over programming languages to operating systems, discuss complex and intricate mechanisms and systems. Engineers who *develop* such mechanisms and systems (e.g. an operating system) use models to deal with their complexity. Instructors who *teach* the concepts behind those mechanisms and systems often implicitly use models to make those concepts more approachable: they present simplified abstractions and draw diagrams to focus on the essential.

In this position paper we propose to make this implicit use of models explicit. We propose to use models as a teaching tool in *all* courses where they are helpful, not just in a course on models or model-driven development. Moreover, we present an infrastructure, Informa, that provides support for integrating models into an interactive classroom.

Keywords: Models for understanding, technology-enhanced learning, clickers

1 Introduction

Maybe the best way to teach models is **not** to *teach* the topic of models. Maybe the best way to teach models is to teach other topics by *using* models as an explanatory device. In this paper we propose to subliminally introduce modeling already in early computer science courses, decoupled from the notion of model-driven software engineering, by creating models when explaining and analyzing newly introduced concepts, and by asking students to create or transform models to demonstrate their understanding of such concepts. In the next section we outline the use of models for teaching one example computer science topic: programming. We believe that the approach applies equally well to other courses, especially courses related to systems topics, such as computer architecture, operating systems, compilers, or databases.

2 Example: Using Models for Teaching Programming

"Programming" is one of the central topics in a computer science curriculum. When teaching students how to program, instructors have to explain the semantics of a given programming language. Besides giving students a multitude of examples and informal descriptions of the language's semantics, textbooks and instructors often also use more formal textual (such as BNF

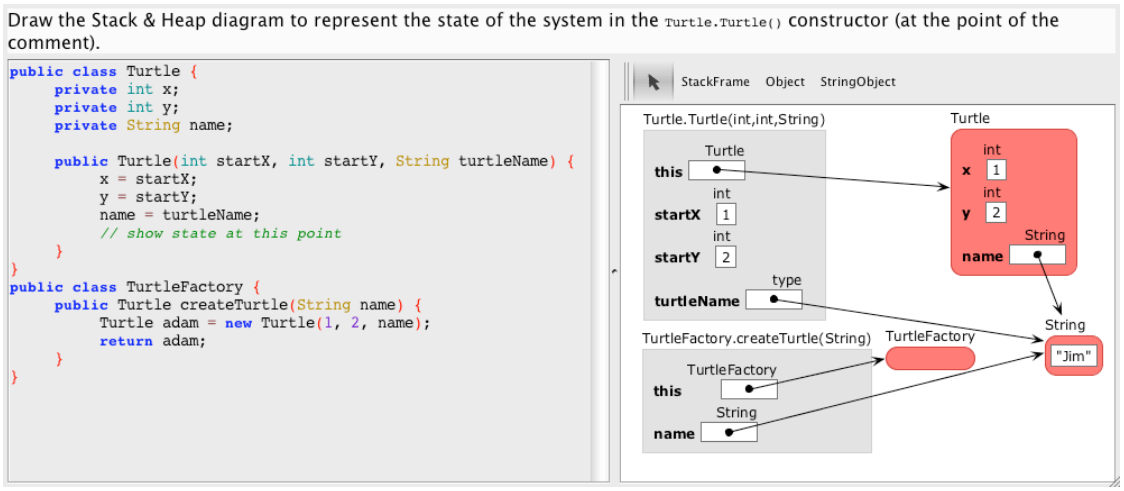

Figure 1: Informa problem requiring a student to model the state of a running program

when discussing syntax) or visual (such as flow charts when discussing control flow) modeling languages.

3 Models in the Classroom with Informa

Brandsteidl et al. [BWH11] have shown the benefit of using innovative teaching approaches for teaching modeling. A particularly innovative approach that they have not discussed, the use of classroom response systems (clickers), has been shown [FM06, RPA04] to be beneficial for teaching a broad range of scientific topics. Clickers are a form of remote controls with a small number of buttons that all students bring to a lecture. The instructor then poses a (multiple-choice) question, and the students submit their answers by clicking the corresponding button on their clickers. The instructor's clicker server collects all responses and provides the instructor (and often also the students, via a classroom projector) with a histogram of the submitted answers. This use of clickers allows an instructor to immediately check whether students understand the topic she just explained, and it allows students to check their understanding already during the lecture.

We propose to use clickers for teaching models, and to do so "subliminally" already in early courses that are not themselves about modeling. To do so, we propose to use Informa [Hau08], a software-based clicker system that is not limited to multiple-choice questions. Informa allows an instructor to post arbitrary types of problems, including problems that require students to write text or to draw graphs. Informa is particularly strong for those kinds of problems that one could call "model transformations", where students demonstrate their understanding by transforming one representation of a concept (e.g., an XML document) into a different representation (e.g., a node-link diagram representing the document's DOM tree). Moreover, Informa's Solve & Evaluate approach [HA09, HA11] enables peer evaluation, where students who already have solved a problem get to evaluate the solutions of their peers.

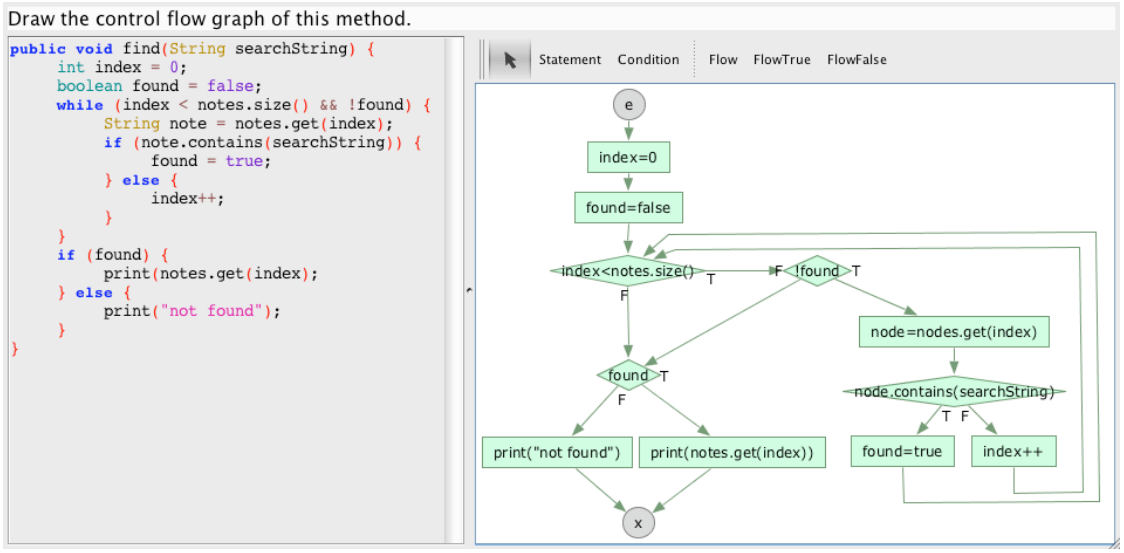

Figure 2: Informa problem requiring a student to model the control flow of a program

Figure 1 shows an example of an Informa modeling problem as it could occur in a programming course. Given the Java program on the left, the students have to model its state (by drawing the diagram on the right), in the form of its call stack frames and its heap objects, at a given point in time in the program's execution. In this problem type (which we call "Stack & Heap"), we do not use UML object diagrams, but we use a visual language introduced in a textbook [BK06] specifically for modeling the state of a Java program in terms of its stack and heap. The modeling language is limited to the aspects that are essential to the concept being taught, and the user interface of Informa allows students to efficiently[1] draw these diagrams.

Figure 2 shows a different kind of problem type (called "Graph") supported by Informa. Here the instructor asked the students to model the given method (left) as a control flow graph (right). The Graph problem type is more general, because it allows the instructor to define simple metamodels (visual languages based on node-link diagrams with different types of nodes and edges). This way, instructors can ask students to draw control flow graphs, to draw simple class diagrams (currently nodes do not support structured contents, so only the class names can be shown), to draw call graphs, data-flow graphs, DOM trees, or any other type of model representable by graphs with unstructured nodes and binary links.

Informa supports many other problem types, such as multiple-choice questions, text highlighting, or text editing. Moreover, it is architected as an extensible framework, allowing developers to implement other kinds of problem types, such as editors for the UML and other modeling languages.

[1] Our students found drawing these diagrams with Informa so efficient that they asked us to provide them with the corresponding editor so they could also use it to draw such diagrams when taking notes during lectures.

4 Conclusions

Modeling is a complex intellectual activity. The subliminal introduction of models already early in the computer science curriculum may turn the creation, analysis, and transformation of models into a natural activity for our students. Instructors can use models to explain arbitrary phenomena and concepts in the entire spectrum of computer science courses, and they can ask students to demonstrate their understanding by creating or transforming models of the concepts under study. To support this kind of formative assessment using models, we propose to use Informa, an extensible classroom clicker system that is freely available[2]. We hope that the pervasive use of models, together with Informa, will improve the teaching not just of models, but also of any other complex concept in computer science.

Bibliography

[BK06] D. J. Barnes, M. Kölling. *Objects First with Java: A Practical Introduction using BlueJ*. Prentice Hall / Pearson Education, 3 edition, 2006.

[BWH11] M. Brandsteidl, K. Wieland, C. Huemer. Novel Communication Channels in Software Modeling Education. In Dingel and Solberg (eds.), *Models in Software Engineering*. Lecture Notes in Computer Science 6627, pp. 40–54. Springer Berlin / Heidelberg, 2011.
http://dx.doi.org/10.1007/978-3-642-21210-9_5

[FM06] C. Fies, J. Marshall. Classroom Response Systems: A Review of the Literature. *Journal of Science Education and Technology* 15(1):101–109, March 2006.
doi:10.1007/s10956-006-0360-1

[HA09] M. Hauswirth, A. Adamoli. Solve & Evaluate with Informa: A Java-based Classroom Response System for Teaching Java. In *Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*. PPPJ '09, pp. 1–10. ACM, New York, NY, USA, 2009.
doi:http://doi.acm.org/10.1145/1596655.1596657
http://doi.acm.org/10.1145/1596655.1596657

[HA11] M. Hauswirth, A. Adamoli. Teaching Java Programming with the Informa Clicker System. *Science of Computer Programming, Special issue: PPPJ 2009/2010, to appear*, 2011.

[Hau08] M. Hauswirth. Informa: An Extensible Framework for Group Response Systems. In *Proceedings of the 4th International Conference on Collaborative Computing (CollaborateCom'08)*. November 2008.

[RPA04] J. Roschelle, W. R. Penuel, L. Abrahamson. Classroom Response and Communication Systems: Research Review and Theory. In *Annual Meeting of the American Educational Research Association*. April 2004.

[2] http://informaclicker.org/