

# A Playful Approach to Formal Models

## A field report on teaching modeling fundamentals at middle school

**Katharina Spies, Bernhard Schätz**  
Technical University Munich  
Department of Computer Science IV  
Boltzmannstr.3, D-85748 Garching (Munich)  
Tel. +49/89 289-17800/17826  
Fax +49/89 289-17307  
spiesk | schaetz@in.tum.de

**Abstract:** Formal methods – besides supplying a set of generally mathematics-oriented formalisms – provide a collection of elementary concepts capturing the essential aspects of (software) systems independent of the chosen formalism. Getting a good intuition of these elementary concepts (e.g., variable, value, state, event) is a prime requisite to education in formal methods. The introduction of Informatics in German middle schools offers an opportunity to introduce these modeling concepts in a non-mathematical fashion to foster their early and intuitive understanding as a basis for further education.

**Keywords:** model-oriented approaches description techniques, modeling, teaching CS, specification, formal methods

## 1 Teaching CS Fundamentals at Middle School

Models have always been the prime tools of computer science. Traditionally, there have been two fractions in computer science dealing with them. On the one hand, software engineers have applied them due to their possibility to extract essential aspects from otherwise too complex systems. Modern modeling techniques cover these points in different flavours, especially with the UML in combination with colloquial descriptions; the fast adoption of these approaches is most likely due to their (assumed) understandability, often achieved at the price of impreciseness and informality. On the other hand, formal methods have traditionally applied models to obtain precise, analyzable and even verifiable descriptions of a system. Approaches like Z, CSP, or ASM address these aspects, together with increasing tool support for analysis; the reluctance to use these approaches is most likely due to a lacking familiarity of typical computer scientist with the underlying base concepts combined with the – nowadays decreasing – lack of convenient notations or editors.

To further acceptance of formal/precise models and make them accessible – in whatever lightweight form – the basic concepts of these models must be commonplace to computer scientists; understanding their advantages must be second nature to them. In the following we show how some basic concepts common to formal models can be taught in middle school, establishing an early and intuitive understanding of these concepts prior to learning a mathematical apparatus to formalize these concepts. We also address illustrating the benefits of using these modeling concepts to 7<sup>th</sup> graders.

To ensure sustainability, basic CS courses taught at schools should provide the pupils with universally valid as well as applicable CS knowledge. Pupils/students must rather acquire knowledge of typical CS models and methods than of specific tools, languages, or notations. Applied tools and software should be only used as means of demonstration; especially in middle school context, they should be used in a playful manner. Teaching typical concepts of a research field enables the children to transfer the acquired concepts and detect their counterparts in possible fields of application.

We present a field report on two half-year elective courses with 13-years old children held by the authors at the Werner-Heisenberg-Gymnasium (see [9] and [11]) at Garching (Munich, Germany). The approach was applied in four different groups of about 15 pupils (both male and female), over a course of two years. Classes were held over a period of half a year with about 12 sessions of 2 hours each. Classes were focused on immediate experience of the pupils, thus with more than 50% of the time spent on practicing the introduced concepts.

Although no formal evaluation and assessment of the effect of the taught classes was performed, some observations seem to be plausible and hold for all the classes performed: The principle of playful discovery of the introduced concepts, immediate feedback loops, and as little as possible up-front teaching proved a fruitful method of knowledge transfer, leading to a maximum of participation by the pupils (seemingly independent of the academic background of their parents) with little variation of the performed classes. Up to our experience, no significant difference could be found concerning speed of learning and depth of understanding between female and male pupils.

The construction of models proved to be a good method to let children experience typical basic concepts of CS. The curriculum was based on fundamental concepts like objects, states, events, or actions. In the following these concepts and their application in the class material is demonstrated. Modeling –combined with basic algorithmic concepts – helps setting a good foundation of CS methodology. Modeling was learned like a child’s play.

The paper is organized as follows: first, we discuss the importance of modeling as core technique of CS and the possible advantages of understanding the basics of modeling in CS. Based on this, we use three examples based on the course material to explain the objective of the curriculum: learning *fundamental modeling concepts*, understanding the impact of using *models as abstraction and generalisation*, and learning to appraise advantage of *working with precise descriptions*. By applying these concepts in the context of object-oriented models, behavioural models, and algorithmic models, the exercises establish the children’s ability to intuitively apply these concepts without requiring the mathematical apparatus to precisely define these concepts.

## 2 Models as the Language of Computer Science

On the one hand models form the product and the result of CS typical system development. On the other hand, applying models for the representation and construction of systems and considers the activities, all transactions and the process of system development. By achieving a playful approach to (formal) models, an intuitive understanding of the possibilities of these models can be supplied at an early age:

1. Software development to a large extent is about the use of models. Each step is based on working with explicit or implicit models of the system under development or its environment. Learning to apply models on an intuitive level independently of more or less formal techniques supplies the necessary basics for a ‘toolbox’ of concepts needed in software engineering, which are often experienced to be quite challenging in a formal setting.
2. Even without supplying a formal definition, models are easily understood as having a precise meaning, allowing to assess whether a specification is valid. Furthermore, the formality of the language encourages greater rigor in the system specification. If 7th grade CS education gets across the understanding that exact and precise working leads to better results, an important step is done towards better CS products and processes, see e.g. [2] or [6].
3. Using models trains thinking in structures and hierarchies as well as understanding modularisation and the composition of systems, thus supporting the stepwise description of a complex system starting with a simplified model, going via several enhanced and more detailed models to the final models. By learning that models support a view-based description of a system ([6], [7] and [8]), children can experience the use of (formal) models to support the purpose-oriented definition of correct abstractions and descriptions of a system, allowing

to focus on the necessary aspects rather than an formalism.

Therefore, the central objective of the approach presented here is:

- (1) The development of products and/or applications requires appropriate concepts depending on the intended purpose, covering all details and criteria relevant to the problem. (**Abstraction**)
- (2) Based on such models supply suitable and unambiguous descriptions. (**Specification**)
- (3) Descriptions and presentations created using those concepts support a systematic development process. (**Application**)

### **3 Course material: Some concrete exercises**

The main topic of the course is teaching modeling approaches in general, and especially core concepts like object-orientation, behavioural modeling, and the description of algorithms. Based on simple but not over-simplified exercises children achieve an intuitive approach of thinking in models, using standard tools and techniques to present and describe their own ideas.

Office software suites are generally available at schools equipped with computers, supplying a simple tool set for constructing system descriptions. As a second advantage, children nowadays already generally are experienced with the use of office tools. Other similar programs can also be used for the following examples if they support graphical descriptions.

#### **3.1 Specification by classes and objects**

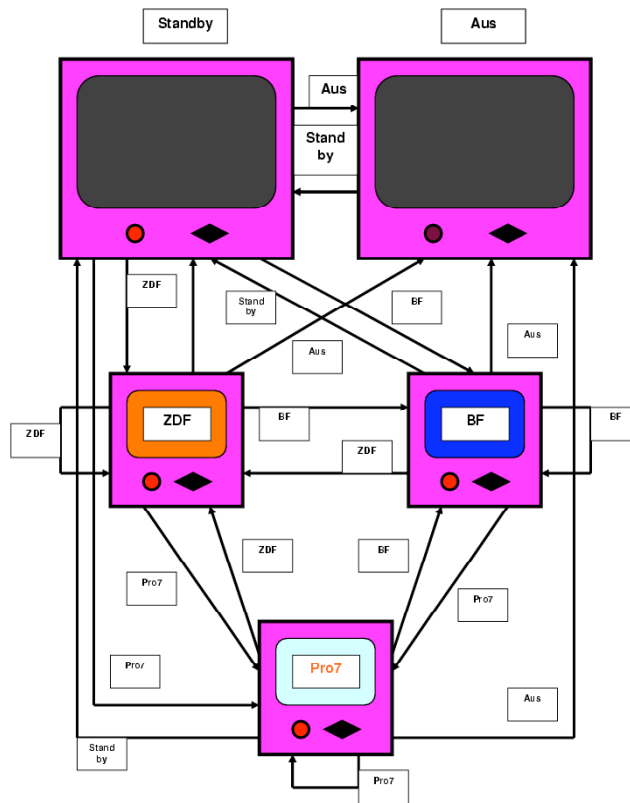
Using the geometric *auto forms* provided by office applications, the concept of specifications (and their models) as abstractions is introduced, as well as the concept of an attribute (or property) of a model. By introducing graphical models composed of geometrical modules (like triangle, rectangle or circle with attributes for each module (like fill or line color, position), the usefulness of specifications to precisely describe a specific model as well as a range of models is demonstrated.

By constructing graphical models (e.g., a TV set or a telephone) from those basic shapes, the pupils intuitively experience the concept of underspecified descriptions – corresponding to those basic shapes – defining classes allowing to generate possible instances – with specific colour, position, etc. – in a stereotypic fashion by fixing the attributes of those instances. By using a textual language representing classes, objects and attributes to describe instances of these basic shapes, and letting the pupils reconstructs a complex model according to a given textual description, the children experience the use of specifications to describe a specific model.

The acquired knowledge of structured/formalized specifications is used in a second step to clarify the benefit of precise and clear descriptions. This is done by playing a group game with switched roles of specifier and implementer. After designing a graphical model and specifying the model using the textual syntax in each group, the specifications are swapped between groups, and group is requested to redesign the picture by using the given specification. As in most cases, the original and the redesigned model do not complete correspond, this makes the pupils realize the impact of precise and clear. Having experienced the discrepancies with their own specification – without any influence of the instructor – the pupils get a good understanding of the consequences of underspecified models.

#### **3.2 Automata to describe components behavior**

All kind of graphical models designed until this point give a good background to discuss several main aspects in modeling and specification like detecting and construction an abstraction of an actual



**Figure 1: A “Formal” Model of Behavior**

system, the impact of precise descriptions, the use of given tools and modules, the structuring of systems, and the implications of using standard (CASE) tools. But the presented models focus primarily on static aspects and properties of the systems, like the arrangement of the components. This is certainly an important aspect in CS modeling because of the necessity to describe and specify e.g. the architecture of a system.

Obviously, however, the dynamic aspects like the behaviour are not covered by these means and must be modeled, described and specified otherwise. The limitations of the techniques introduced so far in the curriculum concerning the description of additional aspects a system can be experienced when discussing the functionality of a system specified only in terms of its (static) interface. The pupils are able to design the layout of the mobile phone with *auto forms* supplied by the office suite. But the question if this picture models a mobile phone or only the receiver of a conventional telephone can't be answered. The children learn that the picture is enough to represent a telephone but not enough to model the main characteristics of a mobile phone that is in fact given by its behavior.

To deepen the dichotomy of structural and behavioural description of a system, a second exercise is given by modeling a TV with 3 main stations, a standby and the out functionality, as shown in Figure 1. At first children are able to model the layout of the TV. Using this picture, the next step is designing five different TV to show its five states. Only some instructions allow the design of an automaton with five different TV pictures as states, connected via arrows, if there it's possible to reach that state from the other. The knowledge of the remote control and the appropriate switches the design with labelled edges is simple.

### 3.3 Algorithmic thinking

After working with the above topics, children are able to use objects and classes to model architectural



The presented course was taught in two half-year elective courses in the 7<sup>th</sup> grade, without 15 pupils (both male and female, ages 12 to 13) per course. The experiences have shown that children in this age group easily understand the basic concepts presented in the sections above. With little training by smaller examples, they generally manage to apply the introduced techniques (e.g., specification by automaton-like notations) to new problems (e.g., the TV-control) within a session. Furthermore, by supplying simulation for the generated “specifications”, they quickly embrace the possibility to use those concepts using a non-code oriented modeling paradigm. Finally, by exchanging “specifications” between different teams, children have a first-hand experience of the need to precise specifications during the validation of second-hand models.

While in this field-study office tools have been used as the means to convey the concepts, special open-source tools like LTSA [12] may even prove more useful if applied in the direction of putting the intuition above the formalization. In future classes, the use of such tools will be investigated more closely. Currently, additionally some simple tools are developed supporting specific aspects like the specification and simulation of simple programs (e.g., controlling a mouse like in Section 3.3).

Currently, in German curricula for middle schools computer science education is optional and therefore there is little tendency to install state-wide class material. Therefore, the class material presented here – including special adaptations of the applied tools – is only updated and extended on a on-demand basis when holding those optional classes.

## References

- [1] R. Baumann. *Wann sind zwei Objekte gleich?* LOG IN 124. pp 43-52. 2003
- [2] M. Broy. *Unifying Engineering Models and Theories of Distributed Software Systems*. Working Material Summer School Marktoberdorf. August 2002.
- [3] M. Broy, P. Hubwieser. *Ein neuer Ansatz in der Schulinformatik*. LOG IN 17. Heft 3/4. pp 42-44. 1997
- [4] W. Hartmann, J. Nievergelt. *Informatik und Bildung zwischen Wandel und Beständigkeit*. Informatik Spektrum 25. Heft 6. pp 465-476. Dezember 2002
- [5] P. Hubwieser. *Modellierung in der Schulinformatik*. LOG IN 19. Heft 1. pp 24-29. 1999
- [6] B. Schätz, K. Spies. *Modellierung im Software-Engineering; Grundlagen und Anwendung*. Lecture Notes. Lecture in Summer 2001. Technische Universität München. 2001.
- [7] B. Schätz, K. Spies. *Model-based Software Engineering*. ICSSEA 2002.
- [8] B. Schätz, A. Pretschner, F. Huber., J. Phillips. *Model based Development*. Technical Report. TUMI-0402. Technische Universität München. 2002.
- [9] K. Spies. *Informatik Kinder-leicht?! 1 Jahr Wahlkurs Informatik in der 7ten Klasse*. Jahresbericht Schuljahr 2002/2003. Werner-Heisenberg-Gymnasium Garching. Juli 2003.
- [10] A. Schwill *Computer Education Based on Fundamental Ideas*.
- [11] [www.whg-garching.de](http://www.whg-garching.de) Homepage Werner-Heisenberg-Gymnasium. Garching bei München.
- [12] J. Magee. J. Kramer. *Concurrency: State Models & Java Programs*. Wiley, 1999.