

# Teaching TECS — A Case Study for MMiSS

Markus Roggenbach

FB 3 — Mathematik und Informatik, Universität Bremen  
roba@informatik.uni-bremen.de

**Abstract.** TECS (Techniques for the development of Correct Software), a two semester course presented at Universität Bremen, gave rise to first experiences with teaching in the context of the MMiSS project. The talk will present sample slides of this course and discuss the benefits of the MMiSS teaching environment for the different participants: the author of the course material, the lecturer, and the students.

The two semester course TECS (Techniques for the development of Correct Software) provides a gentle introduction to formal methods for software development. It deals with sequential as well as with reactive systems, using the algebraic specification language CASL [2] and the process algebra CSP, e.g. [3], respectively. On the tool's side, the theorem prover Isabelle and the model checker FDR play central roles. Besides simple exercises explaining single concepts, the TECS problem sheets also includes more complex tasks like specifying a family game (Nine Men's Morris) in CASL; verifying a simple interpreter within Isabelle/HOL; modelling a file system in CASL at both the requirements and the design level; proving the refinement relation between these two specifications in HOL/CASL.

To present courses like TECS within the MMiSS project, a markup language MMiSSI $\LaTeX$  has been developed, which consists essentially of a collection of  $\LaTeX$  style files. These style files provide semantic annotations for sustainable development, and create an adequate presentation in situations where standard  $\LaTeX$  is not sufficient. For example, a style for lecture slides produces a uniform format together with a consistent, configurable color scheme — see the example slide in Fig. 1 — and a variety of additional features, such as hyperlinks, animations, and interactive invocation of applications. Such features are not available in DVI. Therefore MMiSSI $\LaTeX$  produces PDF. Semantic annotations are implemented as newly defined  $\LaTeX$  commands. This way, authors can write usual  $\LaTeX$  first (e.g. to produce slides for a lecture rather quickly), and add semantic annotations later.

Presenting TECS using the presentational part of MMiSSI $\LaTeX$  has been quite successful:

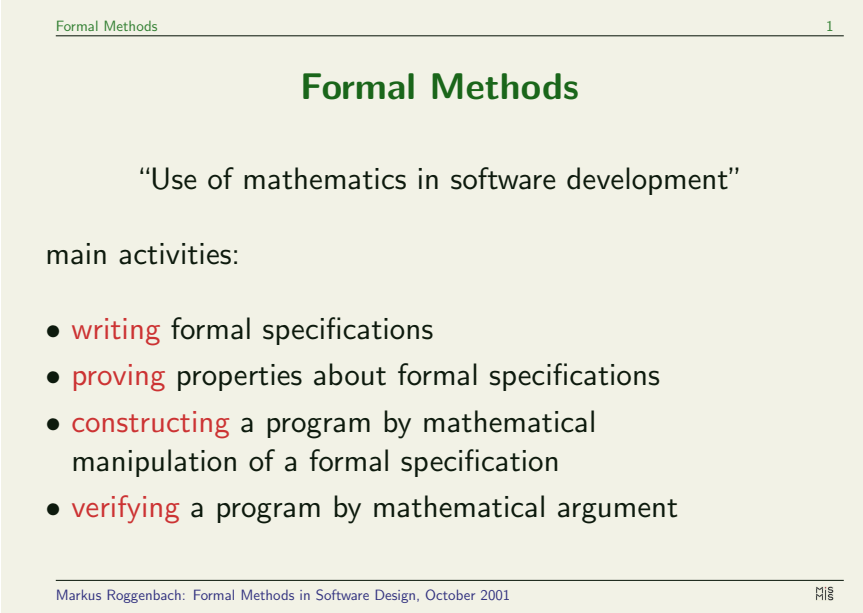
1. For the *author*, the overhead to produce course material within the MMiSSI $\LaTeX$  format is neglectable compared to another presentation system.

2. Besides the usual benefits of a computer based presentation like ‘no slide confusion’, the  $\text{MMiSSLaTeX}$  integration of tool demonstrations in the slides encourages the *teacher* to vivify the lectures by live demonstrations on the computer.
3. The *students* are fond of the
  - the readability
  - the consistent markup, and the
  - download-friendly PDF-filesizeof the slides.

It should be mentioned that these positive results also arise from a cautious usage of computer based presentations: about half of the course material has been taught in ‘classical style’ using a blackboard. A poll among the students of TECS gave the result that this is an optimal mixture.

## References

1. CoFI. The Common Framework Initiative, electronic archives. Notes and documents accessible from <http://www.cofi.info>.
2. CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary, version 1.0.1. Documents/CASL/Summary, in [1], Mar. 2001.
3. A. Roscoe. *The theory and practice of concurrency*. Prentice Hall, 1998.



The slide is titled "Formal Methods" and is numbered "1" in the top right corner. The main heading is "Formal Methods" in green. Below it is the subtitle "‘Use of mathematics in software development’". The text "main activities:" is followed by a bulleted list of four items: "writing formal specifications", "proving properties about formal specifications", "constructing a program by mathematical manipulation of a formal specification", and "verifying a program by mathematical argument". The words "writing", "proving", "constructing", and "verifying" are highlighted in red. At the bottom left, the footer reads "Markus Roggenbach: Formal Methods in Software Design, October 2001". At the bottom right, there is a small logo for "MIS".

**Fig. 1.** A sample slide of TECS