DOI:10.17048/AM.2018.207

Tudor Jebelean
Johannes Kepler University, Linz, Austria
tudor.Jebelean@jku.at

Gabor Kusper
Esterhazy Karoly Egyetem, Eger, Hungary
gkusper@aries.ektf.hu

Anna Medve
University of Pannonia, Veszprem, Hungary
medve@almos.vein.hu

CASE STUDIES IN CERTIFIED SOFTWARE DEVELOPMENT

Introduction

Education of competent professionals constitutes the foundation of the effective and efficient development of software systems, and in general of the complex systems needed in the present and in the future. We approach the improvement of education in software engineering by developing case studies in certified software development. On one hand, this contributes to the education of students, of the academic personnel, and of the industrial practitioners. On the other hand, we aim at producing best practice examples for illustrating the benefits of using formal models and of model based software engineering in concrete industrial environments.

Motivation and Goal

We start from the fact that the university is an ideal environment for the creative combination of education, research, and applications. Each of these activities has a positive inpact on the others. Successful education produces competent professionals for industry, and in the long term also successful scientific researchers. Advanced scientific research is able to increase the efficiency of industry, and it is also improving the education of students because it connects them to the state-of-the-art. The study of industrial applications is very useful (and probably necessary) for an effective education of industrial professionals and for the advancement of scientific research. There are also finer interactions, like the education of students influencing the quality of university educators, rich research producing better researchers, etc.

We adress the situation in software industry, which is notably famous for numerous failures, including massive ones. It is obvious that the security of software systems is a crucial aspect of the emerging information society. We witness various problems in automatic systems which are consequences of incorrect behaviour of software systems. This is mainly caused by the high complexity of these systems and by the absence of practical methods for their verification.

The goal of our work is to contribute to the education all parts involved in education, research, and industry. We contribute to the education of students by exposing them to concrete and successful model-driven development industrial applications. We contribute to the education of academic personnel by training them for the process of solving industrial problems. Finally, we contribute to the education of industrial practitioners by conveying to them the advantages of using formal modelling and model-driven software development techniques for concrete projects.

Framework

Our concrete activities took place in a multiple framework, constituted by: application projects with industrial partners and the Software Competence Center Hagenberg (SCCH), precompetitive projects with industrial partners in conjunction with Master of Science internship of ISI (Internationa School of Informatics, Hagenberg) students, and several cooperations with industrial partners in the frame of a special bilateral (Hungary – Austria) project on the topic. The partners of this project are: the Research Institute for Symbolic Computation (RISC), Johannes Kepler University of Linz, Austria; the Esterhazy Karoly Egyetem, Eger, Hungary; and the Pannonia University in Veszprem, Hungary.

Methodology

The most general concept of our methodology is the model-driven software development, which is illustrated in Fig. 1. In more detail, the real-life problem must be first translated in an abstract problem, which is a mathematical model reflecting the relevant domains. This model is investigated using mathematical techniques and the solution of the problem is developed in the frame of the mathematical model, in form of mathematical algorithms, which are independent of any software technology (like e.g. computer systems, programming languages, software libraries, etc.). As the final step,

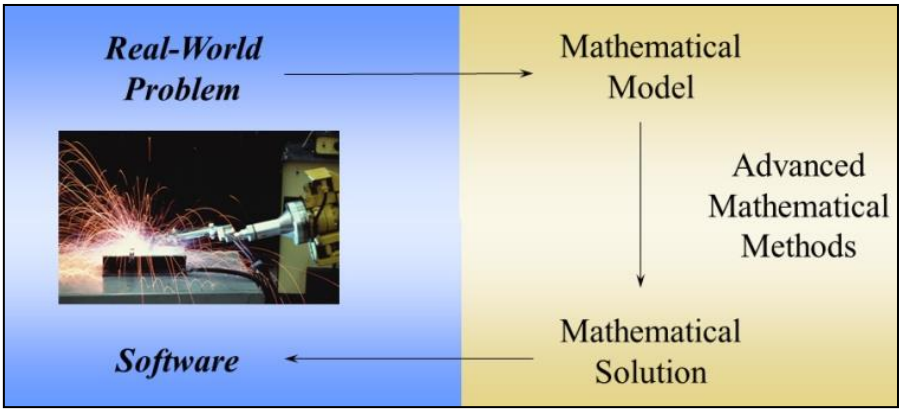

Figure 1. Solving problems by mathematics and software.

the mathematical abstract solution is applied in real life by implementing the appropriate software and distributing it on the appropriate computing environments.

In more detail, our methodology consists in: formal modelling and specification, formal verification by proving, systematic certification, and stepwise refinement.

By formal modelling we mean the construction of the mathematical theories relevant for the problem which has to be solved. Specification is the formal description of the requirements for the needed software.

Formal verification consists in two steps: first the verification conditions are produced, then these are proved. The verification conditions are a set of logical formulae which express the connection from the functionality of the program and the required specification. The proofs should be produced in a format which is easy to understand.

By systematic certification we understand the production of documents which describe the activities of modelling, solving, implementation, testing, verification, stepwise refinement, etc. These documents should reflect in a convincing way the correctness of the system which is realised.

Stepwise refinement applies to complex systems which are designed top-down, starting at coarse description of the main components and refining their design step by step until the final implementation. This refinement works hand in hand with modelling, testing, and verification and is supported by various tools for model-driven software engineering – see [8].

We see the mathematical logic as the main framework for the model-based software development. This is because mathematical logic offers both the language and the methods for all the activities mentioned above.

The construction of the models and the expression of specifications can be done in the language of mathematics – namely higher-order predicate logic in general, and for specific applications one can use various adapted logics, like first order predicate logic, temporal logic, description logic, etc.

The language of mathematics, as well as various mathematical techniques are appropriate for the investigation of the models – for instance deriving certain properties of the respective notions which are interesting from the point of view of the application at hand. Algorithms are also expressed in this frame, as conditional rewrite rules. Moreover, in certain situations the algorithms can be derived automatically derived from the proofs of specific properties of the model.

When algorithms are produced manually, or when they are expressed in a programming language, then verification can be performed by automated reasoning techniques. Namely, first the verification conditions are produced in a relatively easy way – see [2, 9] – however proving them is in general very difficult. At RISC we develop the automated reasoning system *Theorema* – see [1] – whose aim is to offer computer support for the development of mathematical models, of algorithms, and for proving in natural style (that is, in a style similar to human produced proofs). The system allows to perform all phases of modelling and solving activites in one single frame and in the universal language of predicate logic, and the natural-style proofs can be used as a part of certification documents.

Examples

Currently we have contacts and work is in progress with industrial partners from Austria, Hungary, and Romania. In Austria we work with MIBA (Linz), Software Competence Center (Hagenberg), and RISC Software (Hagenberg). In Hungary we are in contact with: Sightspot Network (Eger), InnovTech (Eger), Sente Soft (Eger), ZF Hungary (Eger), Haliphon (Budapest,), Johnson Electric (Hatvan), Fornax (Veszprem).

In Romania the partners are: Lasting (Timisoara) and Continental (Timisoara).

The following applications have been selected for the case studies:

- matching jobs against candidates – see [6],
- optimization of energy consumption in smart homes – see [7],
- synchronization of actions of mobile phones,
- geometric quality estimation of disk brakes,
- mathematical operations on embedded computers,
- workflow management – see [3],
- testing of automotive hardware components.

Currently the following mathematical models are used for these applications:

- set theory – see [6],
- probability theory – see [7],
- graph theory – see [3],
- analytic geometry,

- integer arithmetic,
- automata theory,
- model checking and SAT solving – see [4, 5].

Within the models we developed some algorithms interesting from the point of view of the respective applications. Some of these algorithms have been developed in cooperation with the industrial partner and starting from the existing implementations, by refining and improving them using the formal model.

The algorithms developed in the frame of the respective models have been simulated using certain available mathematical tools, on this occasion some of the models and the corresponding algorithms have been improved and some errors have been corrected. For certain parts of the implementations we applied the methods described in [2, 8] for systematic formal verification, and using the automated reasoning capabilities of the *Theorema* system developed at RISC - see [1]. The evolution of the models in parallel with the development of the software has been done using some of the principles presented in [8]. Part of the algorithms have been already implemented and tested by the respective partners, while for some other the implementation, the testing and the debugging is in progress.

Part of the work has been materialized in scientific publications as shown above, however part of the concrete applications cannot be made public for reasons of industrial confidentiality.

Conclusions

Industrial practitioners need further education on formal models and model driven software development in order to be able to apply in succesfully in their activity. Academic personnel needs closer contact to industrial projects in order to be able to realize the full potential of mathematical and engineering methods. Education of students towards practical application of formal modelling can increase the use of formal certification and of model-driven software engineering in the future. Formal certification is possible, but requires hard work and fine adaptation to the concrete applications, because of the variety of situations which we find in practial applications. Furthermore, domain-specific natural-style proving methods need to be further developed in order to be able to solve real-life situations

References

[1] B. Buchberger, T. Jebelean, T. Kutsia, A. Maletzky, W. Windsteiger. *Theorema 2.0: Computer-Assisted Natural-Style Mathematics.* JFR 9 (1), pp. 149-185. 2016.

[2] M. Erascu, T. Jebelean. *Automated Certification of a Logic-Based Verification Method for Imperative Loops.* Contributed talk at CiE 2012 - How the World Computes, 2012.

[3] T. Jebelean, A. Medve. *Formalization of Workflows Using Fork-Join Automata.* Technical Report 12-21, RISC, Johannes Kepler University of Linz, Austria. December 2012.

[4] T. Jebelean, G. Kusper. *SAT Solving Experiments in Multi-Domain Logic.* ICAI 2011, pp. 95-105, 2011.

[5] T. Jebelean, G. Kusper. *Multi-Domain Logic as a Tool for Program Verification.* SCSS 2010.

[6] T. Jebelean, N. Popov. *A Set-Based Approach to Qualitative and Quantitative Estimation of Competence.* Technical Report 17-05, RISC, Johannes Kepler University of Linz, Austria, 2017.

[7] T. Jebelean, N. Popov. *Energy Management in Smart Homes by Statistical-Based Prediction.* Technical Report 17-06, RISC, Johannes Kepler University of Linz, Austria, 2017.

[8] A. Medve. *Model-based Framework for Integrated Evolution of Business and IT Changes*. ICSOFT 2012, pp. 255-260, SciTePress, 2012.

[9] N. Popov, T. Jebelean. *Sound and Complete Verification Condition Generator for Functional Recursive Programs.* In: Numerical and Symbolic Scientific Computing: Progress and Prospects, U. Langer and P. Paule (ed.), pp. 219-256. Springer, 2011.