1. Designing real-time systems with UML – the ROPES method

Zbigniew Huzar

Object-oriented approach becomes dominating in contemporary software engineering. After a period of tempestuous development of object-oriented methods for software development, the Unified Modeling Language (UML) has emerged as a de facto standard. The UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system’s blue-prints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. Emerging of the UML gave rise to development of object-oriented methods of software development. Additionally, common UML acceptance enables development of professional tools supporting the process of software development.

The ROPES (Rapid Oriented Process for Embedded Systems) is one the first object-oriented methods of real-time system development, which bases on UML. The chapter gives a general review of the method, and next gives a simple case study illustrating application of the method.

It should be underlined that the UML is still developed. Its present version – UML 1.4 – does not take into account the needs of real-time systems development, but the next expected version – UML 2.0 – should allow for it.

2. Temporal logic in the software specification and verification: Towards real-time systems

Radosław Klimek

Temporal logic belongs to the family of modern and formal logics, as well as it belongs to the group of classical logics. That means that this property does not refer only to its ancient origins but rather to the fact that it does not change the meaning of the well-known logical constants. In addition to this, the modal functors relate to the
truth of logical formulae. Temporal and modal logics originate in some philosophical considerations, grammatical tenses and some linguistic notions but the roots of these logics go back to the ancient times. Now temporal logic is a well-established and well-explored formalism. At the present time the position of temporal logic is quite strong and this is a result of two important factors. Firstly, temporal logic increases the expressive power of the formal languages which are based on it. These languages are created with the intention of making the task of the specification of complex systems much easier and more effective. Subtle interactions among software components is another feature of these systems, i.e. concurrent, reactive and even real-time ones. Secondly, the phase of the basic theoretical research is completed and some approaches to the formal verification are proposed. Advantages and disadvantages of these methods are discussed but one of these methods makes possible a fully automatic verification of realistic and non-trivial systems.

Linear time and branching time are two basic time models which are considered in temporal logic. Temporal logic does not enable to measure a time directly. The correctness of real-time systems depend on timing constraints. Thus, real-time systems require a special treatment in the area of temporal logics and some extensions of syntax of temporal logic are proposed. Temporal logic enables to describe the fundamental system properties, i.e. safety, liveness and fairness. All these issues are discussed in this work.

3. **Information project planning**

*Stanislaw Szejko*

The rapid evolution in information technology during the last years has resulted in more complex systems, of higher quality and better fitted to user needs being required. As software makes a very specific stuff a lot of attention is paid to information systems development, implementation, and software process improvement. Still, a lot of projects either fail or exceed their time and budget limitations, or simply their results do not meet the expectations. Effective management of IT projects and their successful planning can be seen as panacea to the situation. Hence the discussion of the project planning, its aims, objectives and levels across the software development process is preceded with the projects presentation in a wider context of company’s business plans, risks and strategies of their development, as well as project infrastructure aspects. Based on these, a method for step-by-step project planning, starting from business case analysis and going through considering the project risks, choosing its development strategy, task identification and evaluation, and finally the project scheduling is presented. While different methods for project evaluation and scheduling are shortly described the precedence diagrams are used to demonstrate the project activity planning task. Motives for plan iterations, like critical path analysis and allocation of the resources, complete the presentation.