



The modular approach of a real-time course

Shmuel Tyszberowicz

Tel-Aviv University and The Academic College Tel Aviv Yaffo

Abstract. This article describes some thoughts concerning the needed syllabus for a “Real-Time Systems” course for computer science students. The ideas described are based on several years of experience in teaching such a course both for graduate and advanced undergraduate (in the last year of studies) students at several academic institutes: Tel-Aviv University, the Open University of Israel, and the Academic College of Tel-Aviv Yaffo.

1 The problem

For over a decade I have been teaching undergraduate and graduate students at several academic institutes various “Reactive and Real-Time Systems” courses. *Reactive systems* are computer systems that continuously react to their physical environment, at a speed determined by the environment. *Real-Time Systems* (RTSs) have reactive behavior. An RTS involves control of one or more physical devices with essential timing requirements. The correctness of an RTS depends both on the time in which computations are performed as well as the logical correctness of the results. Adding the term reactive system to the name is due to the fact that we mainly deal with the reactivity of the system.

The syllabus of the course, its rationale, and the kind of projects the students should do have already been described in [4, 6, 5]. Basically, both courses focus on the development of *provably correct* reactive real-time systems. They cover the full life-cycle of a system development: specification, design, implementation, and verification. The courses combine theoretical issues with practical implementation experience, trying to make things as tangible as possible. The courses have been planned such that no special purpose hardware is needed and all software tools used are freely available from various Internet sites and can be installed quite easily. This makes our course attractive to institutions and instructors for which purchasing and maintaining a special lab is not feasible due to budget, space, or time limitations (as in our case).

The approach we chose was to provide a set of predefined teaching topics, where the instructor selects which topics to teach, and how much time to devote to each of them. As a reference point for the set of topics one can use those suggested by Zalewski [9]. We have carefully studied the lecture topics, and put less emphasis on the knowledge units. Thus, for example, whereas we described analog and digital input/output, we removed topics such as buses, DSP, LAN. This was mainly done due to the fact that computer science students (at least in the above mentioned institutes) do not have the appropriate background. Thus, we actually eliminated the real-time hardware architecture knowledge unit. On the other hand, we added other units such as the synchronous

approach [1]. We also updated the content of others, e.g. the real-time development methodologies knowledge unit. Note that whereas the syllabus offered by Zalewski refers to undergraduate students, the topics can be adjusted to graduate students as well. Actually, we can use identical or at least very similar topics for both the undergraduate and graduate class, but with different content.

During the years I have been teaching those courses, I felt that, much more than in other courses I teach, the success of the real-time course depends on the background of the students. Background in this context refers both to courses already studied and to industry experience, i.e., developing projects that are larger than academic ones. This may have several reasons. Both the content of some courses and the list of mandatory courses is changing from time to time. This fact has directly influenced the content of the real-time course and sometimes also its quality. The academic background of the students may also have a great influence on the content of the course. For example, suppose that the students have already studied threads in Java (or that they have studied both Java and operating systems). If the real-time course requires programming, it is reasonable to emphasize embedded systems, since Java provides good interfaces. Thus it is easier to create interfaces to the embedded hardware, and we can avoid low-level programming. On the other hand, Java is not the right language to deal with hard real-time systems, as it does not have the notion of interrupts and provides no means to specify timing constraints. Moreover, features such as garbage collection or polymorphism, make it difficult to predict the timing, a crucial demand for hard real-time systems. If, on the other hand they know C, hard real time systems can be dealt with.

The mandatory courses can then be used to decide what would be emphasized in the course. Of course, if a minimal number of prerequisites is needed in order to teach other topics, this should be no problem.

Another issue where the real-time systems course differs from other courses is its preparation. For most other running courses, when preparing the materials for a new semester, it seems that the basic topics have not changed, and what might be added are new topics, for example the additions to Java 1.5 as compared to Java 1.4. When looking for new assignments, I almost always end up with a problem that describes a controller of some embedded system, with virtual sensors and actuators. Virtual, since there is no budget to buy even Lego Mindstorm. I always envy teachers who could ask their students to drive satellites, etc (cf. the various proceedings of the Real-Time Educational Workshops), or at least drive a Lego train. And I always asked myself what examples will I use if there would be enough budget.

The tools we have used for our course have also caused a lot of extra work. For example, I am teaching the synchronous language Esterel. The free version from Esterel's website [10] is from 2000. In order to adjust it to changing environments, I have to prepare a file with the needed instructions and the files that are missing in newer environments. A tool that translates Esterel to SMV does work only on the previous Unix version of Esterel. The free academic version of Esterel [11] needs a new license every year.

2 The solution

All the reasons mentioned in Section 1 caused me to take a break from teaching this course. Nevertheless, since I knew that some day I will be asked to teach them again, I tried to see what can be changed, hoping that there is a better, at least easier, way to handle these issues.

It became clear to me that in order for the courses to be successful, and in order to make the preparation process smoother, each of the courses should be built in a way that will easily enable it to be adjusted to the class. It is not enough to have two separate courses, one for undergraduate students and the other for graduate ones; several other aspects need to be considered.

Creating a syllabus for a given class should be based on pre-defined, existing, teaching modules. This can be seen as a multi-dimensional matrix, where each cell contains the relevant teaching module. The first dimension, which also is the most important one, contains the various topics that should be taught.

Other possible dimensions could be the following (in no particular order):

- Who are the students: computer science or engineering students?
There is definitely a difference in the background of the students, and it might be natural to have a more theoretical course for CS students, and a more practical one for engineering students.
- Graduate or undergraduate students?
We can choose the very same topics for both classes, but both the level of formality and thus the difficulty will vary. For example, if the instructor decides that he would like to teach analysis of real-time systems, the following modules will be available:
 - The non formal approach, e.g. Hatley and Pirbahi [3] or Ward and Mellor [8] for undergraduate students.
 - The semi formal approach, e.g., Statecharts [2] as a drawing tool, using only an informal semantics, for undergraduate students and with coverage of the semantics for the graduate students.)
 - The formal approach, e.g., temporal logic [7], for graduate students.
- Theoretical or practical course?
On the one extreme there are real time courses which only teach scheduling algorithms or formal methods for specification and verification of such systems. In a practical course the theory has to be implemented as well.
Subtopics here could for example be which kind of scheduling algorithms to teach: EDF, rate monotonic, both?
- What dominating aspects of a system would we like to emphasize?
Each system is dominated by a specific aspect, which influence the choice of the appropriate language.
 - Data flow dominated systems, such as a heat-control system or active noise reduction system. The preferred language would be Matlab/Simulink.
 - Hard-real-time systems (i.e., systems where the most important issue is timing), such as emergency braking system (in spite of the fact, that there is also substantial data-flow in such a system). The preferred languages are synchronous one, e.g., Esterel, Lustre, Signal [1] or synERJY.

- Embedded systems, for example smart phones. Here the preferred language would be Java.
- What languages do the students already know?
Whereas in the previous item the kind of the system determines the chosen language, here the idea is to choose the kind of systems based on the existing knowledge, as discussed above.
- Budget.
This definitely may influence the course nature. With no budget, the course can be a theoretical one, or use free software. However, almost no large real life project can be implemented.
- Class size.
In a larger class we can have large groups of students working on each project, for example.

Each cell in this multi dimensional matrix contains the appropriate study module. This matrix can also serve as a kind of road map, where the required knowledge for each module is provided. Each cell should also contain information regarding the time needed to teach the module. Hence, while preparing the course, the instructor will know if he or she can teach in the planned order and how much time it should take. This approach also enables cases where the instructors decide to choose only one topic. Thus, one of my colleagues is teaching a course which deals only with scheduling algorithms for hard real-time systems, whereas another is mainly teaching how to formally specify system properties and how to verify these properties using model checkers.

This modular approach makes it easy to add new consideration and hence new study items. I envision instructors adding more dimensions and the relevant modules for each dimension. First, the list of topics should be updated, then the factors that influence the content of each topic should be treated. Last, and maybe most important, modules should be built. The best would be something like a wiki based site, with all the relevant materials. This will include theory, papers, and presentations.

References

1. Nicolas Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic, 1993.
2. David Harel and Michal Politi. *Modeling Reactive Systems with Statecharts: The Statechart Approach*. McGraw-Hill, 1998.
3. Derek Hatley and Imtiaz Pirbhai. *Strategies for Real-Time System Specification*. Dorset House Publishing, 1987.
4. Gilad Koren and Shmuel Tyszberowicz. Graduate course: Reactive and real-time systems. In *RTEW '97: Proceedings of the Second IEEE Real-Time Systems Education Workshop, Canada*, pages 96–103, USA, 1997. IEEE Computer Society.
5. Tal Lev-Ami and Shmuel S. Tyszberowicz. Reactive and real-time systems course: How to get the most out of it. *Real-Time Systems*, 25(2-3):231–253, 2003.
6. Ran Lotenberg and Shmuel Tyszberowicz. Student projects in reactive and real-time systems course. In *RTEW '98: Proceedings of the Third IEEE Real-Time Systems Education Workshop, Poland*, pages 57–62, USA, 1998. IEEE Computer Society.

7. Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York, 1992.
8. Paul Ward and Stephen Mellor. *Strategies for Real-Time System Specification*. Prentice Hall, 1985.
9. Janusz Zalewski. Real-time systems. undergraduated course syllabus, 1993.
10. Esterel home page. <http://www-sop.inria.fr/esterel.org/>.
11. Esterel Technologies. the Esterel academic program. <http://www.esterel-technologies.com/technology/academic-program/>.