# Automation of Real-time Embedded System Design

**Pavel KUCERA**
**Centre for Applied Cybernetics, Brno University of Technology**
**Brno, 61200, Czech Republic**

**and**

**Petr HONZIK**
**Department of Control and Instrumentation, Brno University of Technology**
**Brno, 61200, Czech Republic**

## ABSTRACT

The goal of this paper is to present a tool for automation of designing of real-time embedded system which time-behaviour description is based on formal methods approach. Formal description of the system is based on timed automata diagrams and its temporal logic verification. The purpose of this tool is to automate process of transferring timed automata diagrams into real-time operating system running in the embedded system hardware. This tool significantly simplifies design, implementation and verification of real-time embedded systems because human resources will be concentrated more on the area of specification and verification than implementation issues. This software tool is being developed as an open-source project under the national grant agency support at the Department of Control and Instrumentation, Brno University of Technology.

**Keywords**
Real-time systems, Real-time operating systems, Timed Automata, Temporal Logic, UPPAAL

## 1. INTRODUCTION

The area of formal methods is one of the most dynamic domains of software engineering. There were published initial works about formal methods as a separated branch in the beginning of nineties of 20$^{th}$ century; significant works are mainly [1], [2], [3] and [4].

Since then, formal methods have infiltrated into almost all engineering branches and today it is an interdisciplinary tool facilitating work on design of products and systems from HW to SW parts of applications. Many articles and papers are published every year; they show formal methods as a useful tool at many areas of human activity like aviation, economy, industry, communication, control systems … [5], [6], [7], [8].

Embedded and real-time control systems are very important areas in the everyday life. People and systems itself are surrounded by tons of different embedded system. They usually help us, but in the case of wrong operation they can be a threat to us or other systems. Embedded and real-time control systems have passed through the secular trend in the sphere of design, implementation and verification of the systems. The tradition of control systems in the embedded systems is based on its ability to produce control systems working with highly reliable and safe parameters. To produce such reliable and safe systems via standard (and proved) approach is a painful work requiring a lot of time, skills and human efforts. However, the human work is mainly in the area of implementation (writing routines and programs, designing HW ...) and not in the area of specification and verification.

When considering different approaches of formal design of real-time embedded systems, it is usually problematic to choose an appropriate formal method considering peculiarities of timeliness. The formal approaches used for designing the real-time systems (like UML, ROPES, etc.) concentrate predominantly on theoretical solution of the appropriate interactions of the blocks to provide a real-time behavior in terms of timeliness and synchronism. However, practical experiences show that a fair amount of failures in real-time systems are not only caused by flaws in design but also by underestimation of the efficiency of the particular point-to-point connection. Having designed proper state automata, synchronization points of processes, and employing further measures to diminish compromising of timeliness is crucial if the real-time embedded system is to be transferred to the practice.

## 2. DEVELOPMENT STRATEGY

Project of the real-time embedded system based on formal description requires specific development strategy. This strategy should include control mechanisms based on formal methods used during development of the project as well as suitable final implementation tools. The flowchart of such project's development strategy is shown in Figure 1.

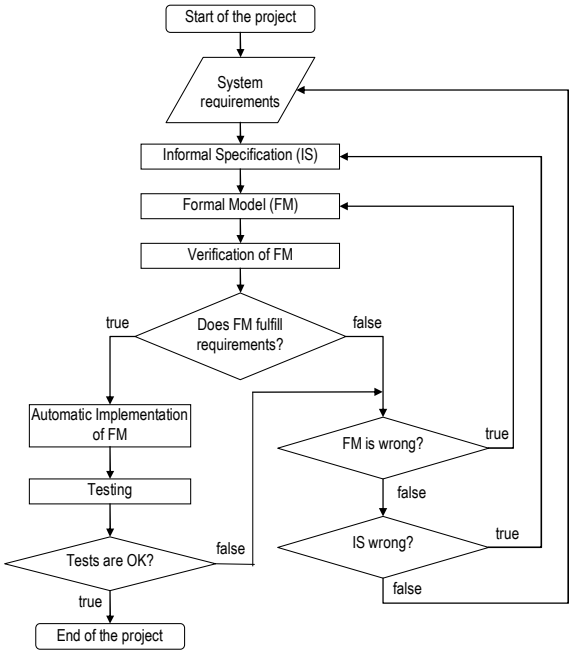


**Figure 1 – Development Strategy**

Flowchart of our development strategy consists of several important steps. Step *Start of the project* represents formal beginning of the project design and it is followed by the step called *System Requirements*. This step is easy to understand by the inquiry: "What we must to do?" The answer to this question is also first specification of the project; inputs from several engineering branches are collected and transformed into uniform utterance. Step *System Requirements* is followed by the step *Informal Specification* (IS) of the system; in this step an informal model of the embedded system task is created. The unexceptional design of the task is finished in this step because an informal model is usually machine interpretation of the design inputs; computer program (indifferently to the target platform) or flowchart are examples of such informal specification. In the case of the formal design, the formal model must be created and verified. Our development strategy solves these issues in the steps *Formal Model* (FM) of the system and *Verification of FM*. When the verification step is finished, first decision replies the question: „*Does FM fulfill the requirements?*" In case that all conditions, specified during collection design inputs, are satisfied, formal model of the project is automatically transferred into the real-time operating system in target HW platform; it is done in the step *Automatic Implementation of FM*. As soon as the implementation is created, final embedded system must be tested.

In the case that tests are not successful or FM does not fulfill project requirements, it is necessary to observe what part of the project is wrong. In this case, there exist three possibilities:

I. The formal model is wrong - for instance, deadlock or livelock is observed during simulation/verification. In this case we have to return back into the step *Formal Model* and another appropriate formal description of the system must be created.
II. The informal model is wrong - for instance, the behavior of the system does not correspond with the claims. In this case we have to return back into the step *Informal Specification* where the behavior of the systems is more precisely specified.
III. If both above mentioned problems do not occur, then we have to return back into the step *System Requirements* and new requirements must be found or precisely determined.

In the simplified way, the entire sequence is done in the following four steps:

1. Define system's requirements.
2. Create formal model of the system from the informally specified system's requirements.
3. Simulate the formal model and verify system's requirements.
4. Transfer formal model of the system into the real-time operating system for target HW platform.

## 3. FORMAL MODEL

As was mentioned in the introduction, an appropriate formal method considering peculiarities of timeliness must be chosen. For the purposes of our automation tool, timed automaton and temporal logic have been chosen as an appropriate description method. The main advantage of this formal method is that there exists an integrated tool environment for modeling, simulation and verification of real-time systems called UPPAAL. UPPAAL was developed jointly by BRICS at Aalborg University and the Department of Computer systems at Uppsala University [9]. It is the appropriate tool for system that can be modeled as a collection of non-deterministic processes with finite control structure and real valued clocks, communicating through channels or shared variables. Typical application areas include

real-time controllers, communication protocols in particular, those where timing aspects are critical and embedded systems control.

UPPAAL consists of three main parts: a description language, a simulator and a model-checker.

1. The description language is a nondeterministic guarded command language with simple data types (unbounded integers, arrays, etc.). It serves as a modeling or design language to describe system behavior as networks of automata extended with clock and data variables.
2. The simulator is validation tool that enables examination of possible dynamic executions of a system during early design or modeling stages ant thus provides an inexpensive mean of fault detection prior to verification by the model-checker, which covers the exhaustive dynamic behavior of the system.
3. The model-checker is to check invariant and bounded liveness properties by exploring the symbolic state-space of the system, i.e., reachability analysis in terms of symbolic states represented by constraints.

The theory of timed automaton is well described in [10]. A number of verification tools have been developed for timed systems in the past year. UPPAAL is one of them as is described in [11], [12], or [9]. The other tools are well described in [13].
The goal of UPPAAL has always been to serve as a platform for the tool to provide a flexible architecture that allows experimentation. It should allow orthogonal features to be integrated in an orthogonal manner to evaluate various techniques within a single framework and investigate how influence each other [14].

## 4. AUTOMATION DESIGN TOOL

Automation design tool for real-time embedded system design solves step *Automatic Implementation of FM* in the Figure 1. Formal model of the system is based on timed automata created in UPPAAL. This formal model can be automatically converted into the objects that are easy to implement in real-time operating systems and these objects results in executable code. Structure of such automation tool is shown in Figure 2. Formal model is represented by timed automata diagram(s) created in UPPAAL. Model is stored in XML file. Model Abstraction Layer (MAL) and Real-time Abstraction layer (RAL) are created from this XML file.

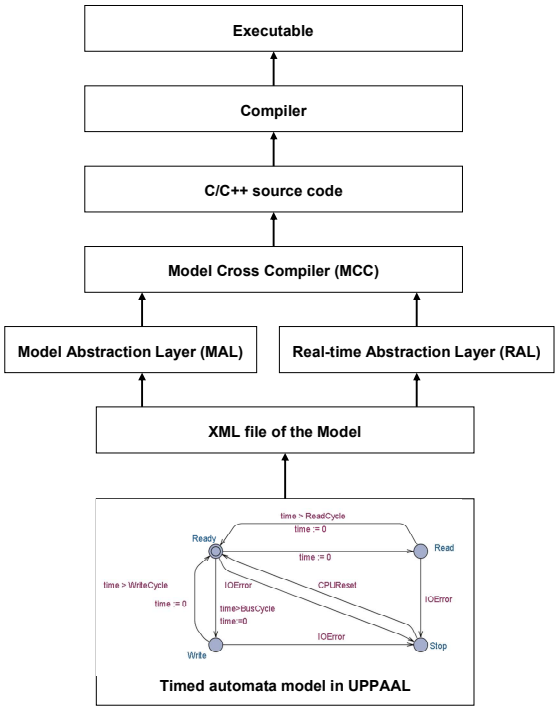


**Figure 2 – Structure of the Automation Design Tool**

The purpose of the MAL is to create an independent interface between timed automata model and real-time entities (processes, threads, synchronizations, IPC) that will be automatically implemented.

The purpose of the RAL is to create a unified structure implementing automata and real-time entities into general real-time operating system.

The Model Cross Compiler (MCC) is an interface for MAL and RAL transferring formal model of the system from the description tool into C/C++ source code. This source code can be included into the corresponding compiler and executable code for the target HW platform can be created. While MAL and RAL are independent on the target HW and target operating system, structure of the MCC strictly depends on the selected real-time operating system and programming language (C or C++). Different operating system uses common basic objects/entities (processes, threads, IPC, synchronization primitives, timers …) with different interfaces – that's why different MCC must be created for corresponding operating systems. For the purposes of this automation design tool, RTX (Real-time extension of Windows) has been chosen as a target platform and C++ as a programming language.

## 5. CONCLUSION

Suggested tool for automation of real-time embedded system design brings several advantages into the area of system control design.

Routine implementation work of the software engineer will be replaced by an automatic implementation; it significantly increases safety and reliability of the control systems, while its verification demand will be lower.

Verification of the system will be based on its formal specification; it brings possibility to verify project during specification process and final verification can be automated or semi-automated - without human intervention.

Total design time will be reduced, while safety and reliability parameters will be preserved or improved with the lower costs.

## 6. ACKNOWLEDGEMENT

This work has been supported in part by Grant Agency of the Czech Republic GA1890030 (Implementation of timed automata into real-time operating systems), Ministry of Education, Youth and Sports of the Czech Republic Research Intent MSM0021630529 (Intelligent systems in automation), and Project 1M0567 (Centre for applied cybernetics).

## 7. REFERENCES

[1] J.M. Wing, “A Specifier's Introduction to Formal Methods”, **IEEE Computer**, Vol. 23, Issue 9, September 1990, pp. 8-23.

[2] J.P. Bowen and M.G. Hinchey, “Ten Commandments of Formal Methods”, **IEEE Computer**, Vol. 28, Issue 4, April 1995, pp. 56-63.

[3] H. Saiedian (ed.), “An Invitation to Formal Methods”**, IEEE Computer**, Vol. 29, Issue 4, April 1996, pp. 16-30.

[4] P. Larsen, J. Fitzgerald, T. Brookes, “Lessons Learned from Applying Formal Specification in Industry”, **IEEE Software**, Vol. 13, Issue 3, May 1996, pp. 48-56.

[5] O. Owe, G. Schneider, “Formal languages and analysis of contract-oriented software”, **Journal of Logic and Algebraic Programming**, Volume 78, Issue 5, The 1st Workshop on Formal Languages and Analysis of Contract-Oriented Software (FLACOS'07), May-June 2009, Pages 291-292.

[6] S. Di Cairano, A. Bemporad, J. Julvez, “Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics”, **Automatica**, Volume 45, Issue 5, May 2009, Pages 1243-1251.

[7] T. Herpel, K. S. Hielscher, U. Klehmet, R. German, “Stochastic and deterministic performance evaluation of automotive CAN communication”, **Computer Networks**, Volume 53, Issue 8, Performance Modeling of Computer Networks: Special Issue in Memory of Dr. Gunter Bolch, 11 June 2009, Pages 1171-1185.

[8] R. Gumzej, M. Colnaric, W. A. Halang, “Safety shell for specification-PEARL oriented UML real-time projects“, **Computer Languages, Systems & Structures**, Volume 35, Issue 3, October 2009, Pages 277-292.

[9] K.G. Larsen, P. Pettersson and W. Yi, ”UPPAAL in a Nutshell”. **Interantional Journal on Software Tools for Technology Transfer**, Vol. 1, Number 1-2, 1998, pp. 134-152.

[10] R. Alur and D.L. Dill, ”A theory of timed automata”, **Theoretical Computer Science**, Vol. 126, 1994, pp. 183-235.

[11] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson and W. Yi, “Uppaal - a Tool Suite for Automatic Verification of Real Time Systems”, **Proceedings of Workshop on Verification and Control of Hybrid Systems III**, number 1066 in Lecture Notes in Computer Science, Springer Verlag, October 1995.

[12] T. Hune, K.G. Larsen and P. Pettersson, ”Guided Synthesis of Control Programs Using UPPAAL”, **Proceedings of the IEEE ICDCS International Workshop on Distributed Systems Verification and Validation**, April 2000, pp. 15-22.

[13] S. Yovine, “A verification tool for real time systems” **International Jornal on Software Tools for Technology**, Vol 1, October 1997, pp. 123-133.

[14] A. David, G. Behrmann, K.G. Larsen and W. Yi, “A tool Architecture for the next generation of UPPAAL”, **Technical report**, Uppsala university, Sweden, February 2003.