Modelling of Real-Time Embedded Systems in an Object-Oriented Design Environment with UML

Razvan Jigorea, Sorin Manolache, Petru Eles, Zebo Peng
Computer and Information Science, Linköping University, Sweden
{g-razji, g-sorma, petel, zebpe}@ida.liu.se

Abstract

This paper explores aspects concerning system-level specification, modelling and simulation of real-time embedded systems. By means of case studies, we investigate how object-oriented methodologies, and in particular UML, support the modelling of industrial scale real-time systems, and how different architectures can be explored by model simulation. We are mainly interested in the problem of system specification as it appears from the prospect of the whole design process. The discussion is illustrated by a large system model from the telecommunications area, the GSM base transceiver station.

1. Introduction

Real-time embedded systems (RTES) have to fulfill increasing complex requirements concerning functionality, timing, power consumption, cost, etc.

Design environments have to be developed in order to assist the designer throughout the design process. Figure 1 shows a simplified view of such a design flow. An important tasks performed during the design phase is architecture exploration. Several architectures (differing in number and kind of processors, interconnection structure, number of ASICs, memory structure, etc.) and alternatives for task partitioning are explored in order to find an efficient solution which also satisfies the imposed requirements. The ability to check the functionality of a certain design alternative and to estimate different parameters is essential. Simulation is the most common technique which allows to get a feedback concerning the degree to which a design alternative fulfils the requirements [1].

An important trend in current design methodologies for embedded systems is towards the reuse of pre-designed components. Such an approach has been considered by hardware designers as a practicable strategy [2]. Pre-designed programmable cores and ASICs (called intellectual property – IP), stored in a module library (Figure 1), are used as building blocks for new system designs.

The system specification has to fulfill several requirements. Some of them, like support for concurrency, timing, hierarchy, nondeterminism, are well known and have been discussed in the literature [3, 4, 7]. In this paper we are interested in the problem of system specification as it appears from the prospect of the whole design process. Some of the following issues have been identified as result of our work related to the modelling and design of RTES in the telecommunications area: data and control flow separation*, g*eneric functional and control units, support for architecture exploration, timeliness verification, multiple abstraction levels, support for reuse.

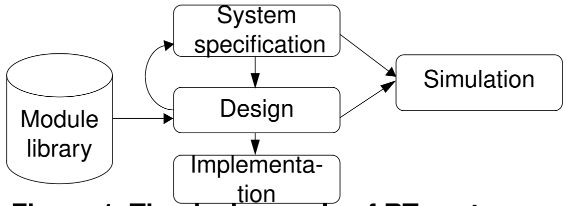

Figure 1. The design cycle of RT systems

Our main focus is on how OO methodologies, and in particular UML, support such requirements. The discussion is based on the experience gained from the modelling of a large application in the telecommunications area, a GSM Base Transceiver Station (BTS). Another example, an intelligent traffic lights controller, has been used for a set of experiments concerning the architecture exploration aspects.

Section 2 describes the functionality of a GSM BTS, our main case study. Section 3 presents solutions to the problems highlighted above, in the context of UML. Section 4 focuses on aspects related to architecture exploration. The last section presents our conclusions.

2. The Base Transceiver Station (BTS)

The BTS is a device in the canonical architecture of the Global System for Mobile Communication (GSM) [6]. Figure 2 shows the place of the BTS in the context of the GSM architecture. The radio interface connects the BTS with the mobile stations (MS). A terrestrial link, the Abis interface, connects the BTS with the BSC. The BSC is connected with the NSS. The BTS comprises radio transmission and reception devices, and all the radio interface signal

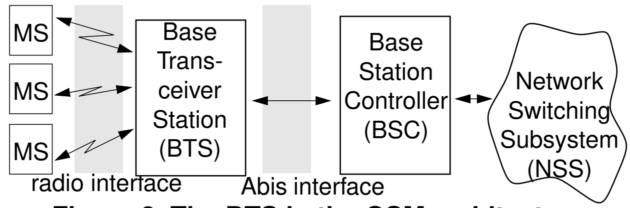

Figure 2. The BTS in the GSM architecture

processing. To counter the high error rate of the radio interface, complex modulation/demodulation algorithms are deployed in the BTS and computation intensive channel encoding/decoding methods are applied. The number of these channel encoding/decoding algorithms is relatively large. In order to adapt more users, a time division multiple access scheme (TDMA) is used. Thus, several communication channels multiplexed in time and frequency are managed by the BTS. The BTS allocates/deallocates and changes the channel mode at the request of the NSS. Signalling protocols between the BTS and the MSs and between the BTS and the infrastructure are used in order to make such a channel management possible.

3. The BTS model

In this section we first present the general structure of the BTS model. Next, solutions to certain specific problems are discussed. The underlying specification and design strategy is based on an OO methodology. The discussion here is in the particular context of UML [4, 8].

The BTS model has been specified and simulated with UML using the iLogix Rhapsody environment [10].

3.1. General description of the model architecture

In Figure 3 we show the structure of the BTS model. Our model has been organized on four layers. The bottom layer consists of purely functional processing units (called functional units — FU). The control tasks to be performed by the BTS are distributed among the other three layers. The responsibilities of the FUs are related to channel encoding/decoding and to data interleaving/deinterleaving. Subsequent interleaving/deinterleaving schemes are applied in order to build radio-bursts from data-blocks and to assemble data-blocks out of the incoming radio-bursts. FUs in the model are specialized for a certain encoding/decoding or

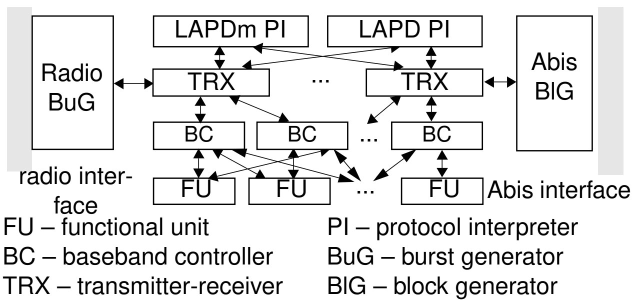

Figure 3. Structure of the GSM BTS model

interleaving/deinterleaving task.

The control layer above the functional layer consists of the baseband controllers (BC). They have to assure a correct operation sequence for each channel mode. BCs are specialized for the control of channels in a certain channel mode.

The next control level consists of the transmitter-receivers (TRX). A TRX controls a set of BCs. Each TRX corresponds to a signalling link. A TRX manages traffic and signalling for eight physical channels.

The TRXs use the services of the third control layer, the LAPD (Link Access Protocol for the "D" Channel) and the LAPDm (LAP for the "Dm" Channel) protocol interpreters [5]. These modules receive from the TRX the signalling data and are responsible for assembling the frames, link maintenance, and notification of the TRX regarding the received message.

The Burst Generator (BuG) models the BTS radio subassembly. It generates bursts to be sent to the TRX. The Abis Block Generator (BlG) has a similar role as the Radio BuG. It generates blocks, simulating the time division scheme deployed on the Abis.

3.2. Deployment of control

FUs can be reused over many generations of telecommunication applications or over different designs. The same philosophy applies to control units. Protocols are organized on different layers and from one product to another only particular layers are modified while other layers are reused. In order to support reusability of FUs and control units, a minimal amount of control has to be deployed to any unit in the model. Thus, the potential of reusing the modelling units increases significantly.

Figure 4 shows the UML statechart for one of the FUs which performs a part of the channel encoding. The unit, a CRC encoder, is unaware of the channel whose processing chain it belongs to. Thus, it could be used as a building block for both speech encoding and for fire codes, as well as for any other design in which CRC encoding is performed. The unit has a very simple interface: after a processing delay (procD) has elapsed from the moment it receives a processing command (evCRCEncCmd) from a controlling entity, it will notify the controlling entity about completion of the processing and offer the transformed data (evCmdCompleted(result)).

The same principle of minimum amount of control deployment was adopted for the controlling units. A BC, for example, manages FUs according to the channel mode of the controlled channel. It is completely unaware of the existence of the other channels (a statechart of a BC is

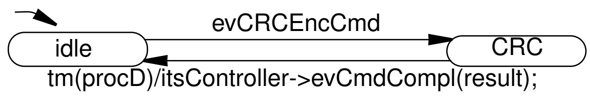

Figure 4. Statechart for CRC processing unit

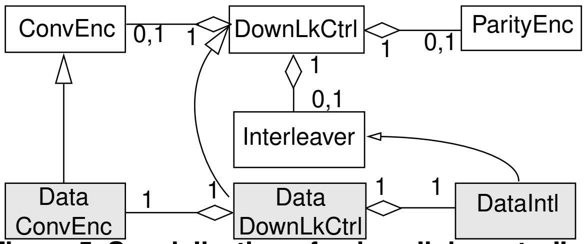

Figure 5. Specialization of a downlink controller

shown in Figure 6).

The TRX, on the other hand, is in charge of the eight channels it manages. It is the only unit which has to be aware of the timing scheme in order to identify a channel correctly and to activate the right BC (Figure 7).

3.3. Separation of control and data flows

Separation of data and control flows improves the units reusability degree. OO modelling particularly suits this requirement. There are objects which model entities on the data path, and objects which model controlling units. Due to the inherent loose coupling between objects, the separation of control and data flow is easy to achieve. This aspect is well highlighted in Figure 3. FUs are operating on the data path, while no control functions are embedded within them. On the other hand, BCs as well as TRXs do not perform any processing on the data flow. They just coordinate lower level entities.

3.4. Uniform interfaces and generic units

Various types of low level units are treated uniformly by the upper layers. This simplifies the modelling process. The use of inheritance relationships leads to such an uniformity.

Particular interleavers, for instance, are derived from a generic interleavers which again are specializations of generic FUs. For the upper, control layers, every FU appears as the generic unit. A similar strategy applies to control units.

In Figure 5 the unshaded boxes form the class diagram of a generic downlink BC. The generic controller aggregates generic FUs of the classes Interleaver, ConvEnc, and ParityEnc. A specific BC aggregates specialized FUs, depending on the particular channel mode it controls. A data downlink BC for example, is an instantiation of the generic class DownLkCtrl, and aggregates an instantiation of the class DataConvEnc and of the class DataIntl. The corresponding class diagram is represented by the shaded boxes.

3.5. Timing aspects

Modelling of timing aspects comprises specification of deadlines and execution delays.

Execution delays are modelled as time-triggered state transitions (see Figure 4). A parameter of a FU characterizes the execution delay of that FU. During architecture exploration, this delay depends on the particular processor to which

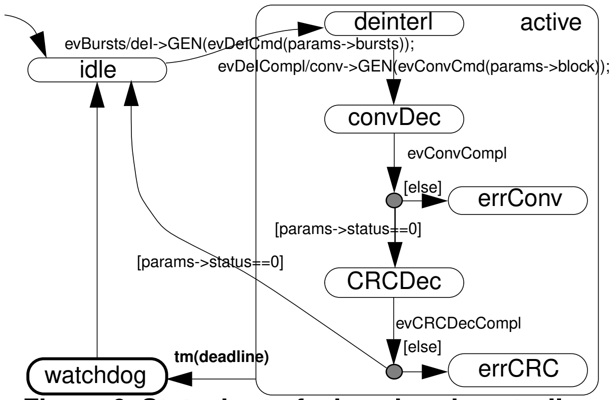

Figure 6. Statechart of a baseband controller

the unit is assigned for execution. This delay has to be estimated [9] and the resulting value is assigned to the unit. The delay attribute is specified in the superclass of all FUs.

Control units coordinate the functionality of lower level units. The control units have to monitor whether certain predefined time intervals have passed and to take certain decision if a deadline has been reached. Deadline is modelled by introducing an additional state, which is entered after a certain time interval has elapsed. Figure 6 illustrates this mechanism for a BC. If the execution delay of the processing chain exceeds the predefined *deadline* value, the watchdog state is entered. The TRX interprets this as an exceptional situation.

3.6. Concurrency issues

Usually, telecommunication devices perform a set of well specified operations on multiple data flows. Identification of this parallelism is important in the analysis phase because it influences the model object structure. In UML, parallelism can be expressed in two ways. First, all the objects are considered to be parallel entities. Synchronization is achieved by means of message exchanges. Second, an entity which has been modelled as a single object can exhibit itself a concurrent behaviour. In this case the statechart corresponding to the object is specified as a set of concurrent components. Such a case is shown in Figure 7, where the statechart of a TRX is depicted. The TRX has to handle uplink and downlink traffic, as well as signalling information addressed to it. Those activities are independent to each other and can be performed concurrently.

3.7. Multiple abstraction levels

Our main focus was set on the specification and design

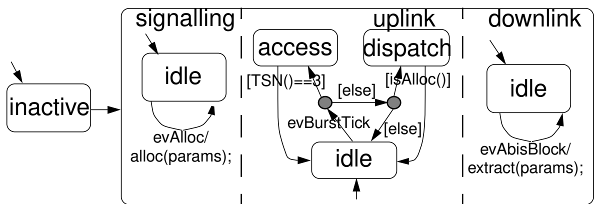

Figure 7. Statechart of a TRX

of the digital components of the BTS. However, in order to perform simulation and architecture exploration, the whole functionality of the BTS had to be modelled. Thus, the radio subassembly was simply modelled as a radio burst generator, at a very high abstraction level. The strong encapsulation, typical to the OO approach, allows for an uniform treatment of entities specified at different abstraction levels. They allow us to concentrate on the refinement of that part of the model we are mainly interested in.

4. Architecture exploration

In order to complete our goal in exploring the way UML suits the RTES design cycle, we imagined a simpler case study, which allows to demonstrate how architecture exploration can be performed. The example we have chosen is an intelligent, adaptive traffic lights controller (TLC).

The controller was designed for a typical two road crossing (one main road, crossed by a secondary road), including pedestrian sideways. It is connected with the controllers in the previous and next crossing on the main road. Beyond the crossing, on the main road, there is a departure sensor, used for detecting if a car left the current crossing. It is needed in order to perform statistical analysis on the time needed for a car to get from one crossing to another. According to the statistical data, the intelligent subsystem (IS) will adjust the traffic lights timing in order to have an optimal "green wave" on the main road.

We considered that the controller is physically mapped on a microcontroller, and the IS is mapped on a processor. Figure 8 presents the object model diagram of the system.

The model was conceived in a way which allows simulation with or without the IS. We assumed that the average waiting time for a car is the performance parameter of interest. Thus, we explored how different processors and communication lines affect this performance parameter. The processing efficiency (PE) is a parameter of the processor which runs the IS, and the communication efficiency (CE) reflects the performance of the communication lines connecting our controller to the two controllers in the adjacent crossings. We define PE (CE) as the ratio between the execution time (communication delay) of the best considered processor (communication link) and the processor (link) under consideration.

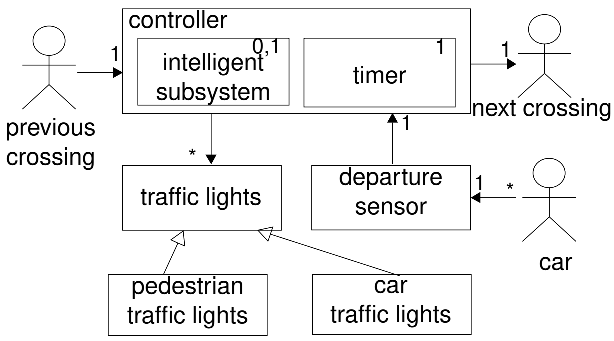

Figure 8. Object diagram of adaptive TLC

Table 1. Simulation results

Intelligent subsystem	Average waiting time for a car (seconds)			
	PE=0.9 CE=0.9	PE=0.9 CE=0.6	PE=0.6 CE=0.9	PE=0.6 CE=0.6
No	11.56	11.56	11.56	11.56
Yes	1.66	5.82	9.09	11.31

The experimental results are presented in Table 1. We run the same scenario, first for the TLC without the IS, and next for the TLC with IS, departure sensor and communication lines. For both settings we explored several values for PE and CE. As it can be seen from Table 1, the IS produces a significant increase in performance. The system quality is degraded for lower performances of the processor and/or communication line. Thus, starting from the performance required for a particular setting, the most cost-efficient architecture (processor and communication infrastructure), which still fulfils the requirements, can be selected.

5. Conclusions

We discussed several issues concerning the modelling of complex RTES using an OO methodology with the UML.

Using the particular example of a GSM BTS we showed how the UML based methodology allows the proper layering of a model, the separation of control and data flows, as well as the definition and usage of generic units. The goals are to facilitate complexity management during the modelling phase. IP-based design and architecture exploration are supported.

Using a smaller example, which allows for a more detailed discussion, we also presented an example of architecture exploration and gave some experimental results.

References

[1] J. Axelsson, *"Holistic OO Modelling of Distributed Automotive RT Control Applications"*, Proc. 2nd IEEE Intl. Symp. on OO RT Distributed Computing, 1999, pp. 85-92.
[2] M. Keating, P. Bricaud, *"Reuse Methodology Manual for System-on-a-Chip Designs"*, Kluwer Academic Publishers, 1998.
[3] A. Sarkar, R. Waxman, J. Cohoon, *"Specification-Modelling Methodologies for Reactive-System Design"*, in "High-Level System Modelling: Specification Languages", eds. J. M. Bergé et al., Kluwer Academic Publishers, 1995, pp. 1-34.
[4] B. Douglass, *"Doing Hard Time"*, Addison-Wesley, 1999.
[5] M. Mouly, M. Pautet, *"The GSM System for Mobile Communication"*, Palaiseau, 1992.
[6] *"GSM 03.02"*, ETSI, http://www.etsi.org.
[7] B. Selic, G. Gullekson, P. Ward, *"Real-Time Object-Oriented Modeling"*, John Wiley & Sons, 1994.
[8] G. Booch, *"Object Oriented Design With Applications"*, The Benjamin/Cummings Publishing, 1991.
[9] J. Gong, D. Gajski, S. Bakshi, *"Software Estimation Using A Generic Processor Model"*, Proc. of the Eur. Design and Test Conf., 1995, pp. 498-502.
[10] *"Rhapsody Reference Guide, Release 2.1"*, iLogix, 1999.