Modelling and Simulation of Heterogeneous Embedded Systems with UML

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

Abstract

This paper explores some aspects concerning system-level specification, modelling and simulation of real-time embedded systems in the context of telecommunication applications. By means of case studies, we investigate how UML suits the modelling of industrial scale real-time systems, and how different architectures can be explored by model simulation. In this paper we are mainly interested in the problem of system specification as it appears from the prospect of the whole design process. Our main focus is on how OO methodologies, and in particular UML, support such requirements and how they can be used for the specification and design of complex embedded systems. The discussion is based on a GSM base transceiver station model.

1. Introduction

Current real-time embedded systems have to fulfill more and more complex requirements concerning functionality, timing, power consumption, reliability, cost, etc. Typical application areas for such systems are telecommunications, automotive industry, avionics, or industrial process control. Embedded systems are very often implemented on distributed architectures consisting of several programmable processors and application specific integrated circuits (ASICs) [1].

Due to the complexity of such systems, design environments have to be developed in order to assist the designer throughout the whole design process, starting from the system specification, going through the design phase, until the final implementation. In Figure 1 we show a very simplified view of such a design flow. One of the important tasks performed during the design phase is architecture exploration. Several architectures (which differ 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.

An essential aspect of such an iterative design process is the ability to check the functionality of a certain design alternative and to estimate different parameters characteristic to it like, for example, timing. Although formal verification and static analysis are more and more used in this context, simulation is still the basic technique which allows to get a feedback concerning the degree to which a specification or design alternative fulfils certain requirements [2].

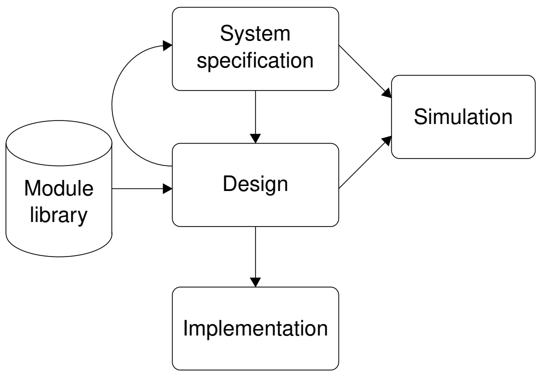


Figure 1. The design cycle of RT systems

An important trend in current design methodologies for embedded systems is towards the reuse of pre-designed components as an alternative to the complete synthesis of the whole system. Such an approach is well known for the software design community but has been only recently considered by hardware designers as a practicable strategy in order to cope with increasing system complexity [3]. According to such a methodology, pre-designed programmable cores and ASICs (called intellectual property (IP) components), stored in a module library (Figure 1), are used as building blocks for new system designs.

In order to support such a complex design process, the system specification has to fulfill several requirements. Some of them are well known and have been much discussed in the literature [4, 5, 8]. Such are the support for concurrency, exception handling, timing, hierarchy, or nondeterminism. In this paper, however, we are mainly interested in the problem of system specification as it appears from the prospect of the whole design process. Thus, requirements like support for IP-based design, separation of control and dataflow, or facilities for system simulation are of particular interest. Our main focus is on how OO methodologies, and in particular UML, support such requirements and how they can be used for the specification and design of complex embedded systems. The discussion is based on the experience gained from the modelling of a large application in the telecommunications area, namely the GSM Base Transceiver Station (BTS). This is a representative example of a large reactive embedded system, and is therefore well suited to illustrate some design problems and possible solutions. Another example, an intelligent traffic lights controller, has been used for a set of experiments concerning the architecture exploration aspects.

In the following section we identify some of the main problems to be discussed. Section 3 describes the basic functionality of a GSM BTS, our main case study. Section 4 presents solutions to the problems identified in section 2, in the particular context of UML. In section 5 we focus on aspects related to architecture exploration. The last section presents our conclusions on modelling and simulation of embedded systems with UML.

2. Modelling of Real-Time Embedded Systems

Embedded systems, and in general all real-time (RT) systems, have characteristics that make their design very complex. They usually exhibit a reactive behaviour, meaning that they respond in a well-defined manner to input stimuli. Moreover, the response is time-constrained, therefore it must occur within a more or less specific time window. Safe critical embedded systems must be also robust, showing correct behaviour even in unexpected circumstances.

Concerning the specification and design of RT embedded systems, several particular issues have been identified and discussed in the literature [4, 5, 8]. Such are state-transition oriented behaviour, inherent parallelism, timing aspects, exception handling, environment dependency, and non-functional characteristics (performance, safety, reliability, power consumption etc.)

As mentioned in the previous section, in this paper we are interested in particular aspects of system specification, which are related to the whole design process. Some of these problems have been identified as result of our work related to the modelling and design of RT systems in the telecommunications area. Here are some of the issues which will be further explored in the following sections:

- *Data and control flow separation.* The model should be organized as a set of functional and control layers. A functional layer performs operations on the incoming data, while a control layer coordinates clusters of functional units. The amount of control implemented in a certain layer should not exceed the minimum needed in order to coordinate the operation of the lower layers.
- *Generic functional and control units.* Many functional or control blocks share certain basic characteristics, differing just in some aspects. Identifying such generic units and specifying them as superclasses allows for an additional degree of reusability.
- *Support for architecture exploration.* Both the specification methodology and the related design environment should provide the support for exploration of alternative implementation architectures.
- *Timeliness verification.* The model must be checked not only for functional correctness, but also for temporal correctness. We therefore need time-checking mechanisms embedded with the model.
- *Multiple abstraction levels.* The specification of a RT system often consists of blocks which are described at very different levels of abstraction. In the initial specification some blocks can be treated as black boxes and be specified at a high level of abstraction, while other blocks are specified in more detail. As result of subsequent design steps, some of the blocks are further refined while other are left unchanged. Despite such an unbalance in the abstraction level, a clear separation between specification of interfaces and that of the internal behaviour, allows the whole system to be kept coherent and executable.
- *Support for reuse.* As mentioned earlier, the reuse of predefined components (IP-based design) is of extreme importance in managing the complexity of the design process. Thus, specifications and models shall be built with a high-degree of reusability as an essential goal. At the same time, the specification and design methodology has to support the reuse of existing components.

In the following section we introduce a typical telecommunication application of high complexity. All the problems highlighted above will be discussed along with proposed solutions in the context of an OO design environment based on the UML.

3. The Base Transceiver Station (BTS)

The BTS is one of the devices in the canonical architecture of the Global System for Mobile Communication (GSM), as defined by the GSM standardisation group [7]. 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 Base Station Controller (BSC). The BSC is further connected with the Network Switching Subsystem (NSS). The BTS comprises radio transmission and reception devices, and also all the signal processing specific to the radio interface. The radio interface is characterized by a high bit error rate. To counter this, complex modulation/demodulation algorithms are deployed in the BTS and computation intensive channel

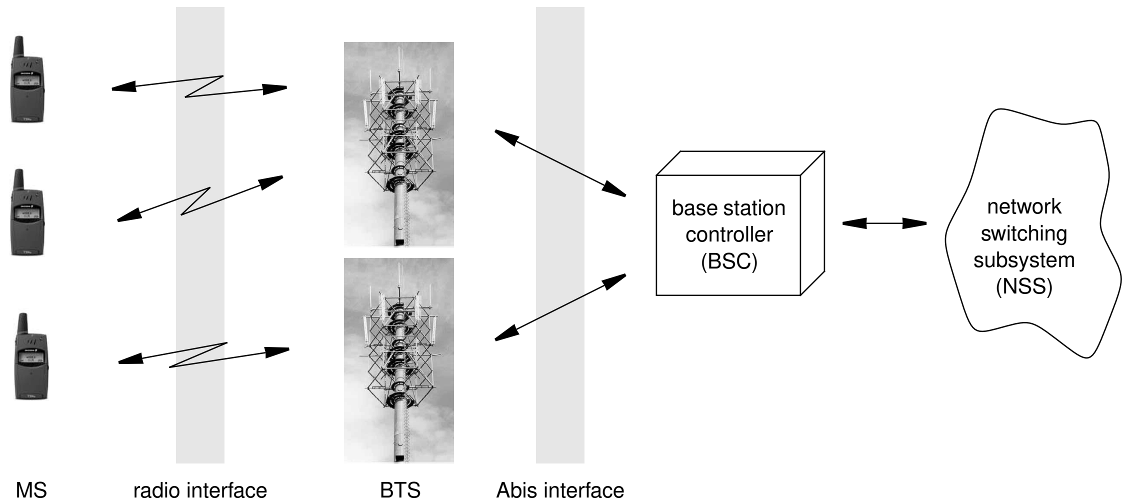


Figure 2. The BTS in the architecture of GSM

encoding/decoding methods are applied. Because of the large range of services GSM offers, the number of these channel encoding/decoding algorithms is also very large. The BTS also performs the so called characterization of the radio interface, i.e. it measures interference level, bit error rates, processes measurements of received power and reports the results to the NSS. In order to adapt more users, a time division multiple access (TDMA) scheme is used. This allows a BTS to manage several communication channels multiplexed both in time and frequency. The BTS allocates/deallocates and changes the channel mode at the request of the NSS. Signalling protocols between the BTS and the mobile stations as well as between the BTS and the infrastructure are used in order to make this channel management possible.

4. The BTS Model

In this section we first present the general structure of the BTS model. Next, solutions to the problems identified in section 2 are discussed. The model underlying specification and design strategy is based on an OO methodology. The discussion here is in the particular context of the UML [5, 10]. General aspects concerning OO methodologies and the basics of the UML have been much discussed in the literature [8, 9, 10], and therefore are not mentioned in this paper.

4.1 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 functional units are related to channel encoding/decoding and to data interleaving/deinterleaving.

The encoding/decoding algorithms are:

- CRC encoding/decoding;
- convolutional encoding/decoding in its various variants (punctured or not);
- tailing;
- fire codes (similar to CRC codes).

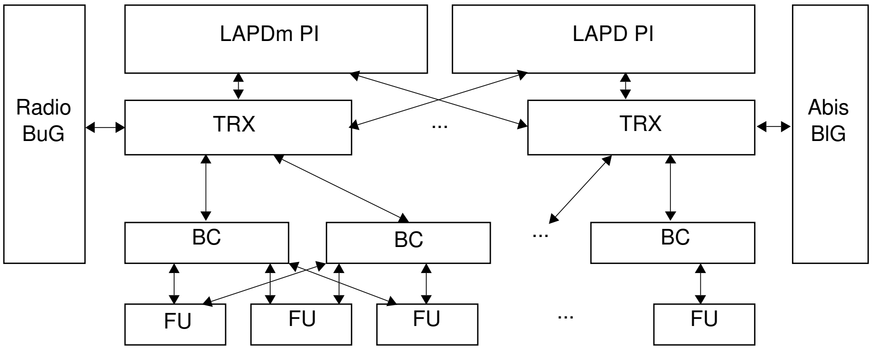
Subsequent interleaving/deinterleaving schemes are applied in order to build radio-bursts from data-blocks and to assemble data-blocks out of the incoming radio-bursts. A functional unit in the model is specialized for a certain encoding/decoding or interleaving/deinterleaving task.

The first layer of control, above the functional layer, consists of the baseband controllers (BC). Their responsibility is to assure a correct sequence of operations (performed by the FUs) for each channel mode. BCs are specialized for the control of channels in a certain channel mode. By channel mode we understand a set of properties that characterize the channel. Such properties are:

- semantics of carried data (speech, data, signalling);
- gross data rate, which classifies channels in full and half rate channels);
- nett data rate;
- whether the channel is dedicated to a particular user at a moment in time or it is a broadcast (common) channel;
- whether the channel is a duplex one or not.

The next level of control consists of the transmitter-receiver (TRX). A TRX control a set of BCs. It emits or receives continuously, but on a single frequency at a given moment. From the signalling point of view, each TRX corresponds to a signalling link. A TRX manages traffic and signalling for eight (due to the TDMA scheme) physical channels.

The TRXs use the services of the LAPD (Link Access Protocol for the “D” Channel) and LAPDm (Link Access Protocol for the “Dm” Channel) protocol interpreters [6]. These modules receive from the TRX the signalling bursts/blocks and are responsible for assembling the frames, maintenance of the link, and notification of the TRX



FU – functional unit
BC – baseband controller
TRX – transmitter-receiver
PI – protocol interpreter
BuG – burst generator
BlG – block generator

Figure 3. Structure of the GSM BTS model

regarding the received message. The two LAPD blocks represent the third and highest layer of control.

The Radio Burst Generator (BuG) (Figure 3) is the module that models the radio subassembly of the BTS. It generates bursts which are sent to the TRX. The BuG has to be aware of the time division scheme, as demodulation is performed differently for different channels.

The Abis Block Generator (BlG) has a similar role to the Radio BuG. It generates blocks, simulating the time division scheme deployed on the terrestrial link. These blocks are then forwarded to the TRXs.

The BTS model has been specified in UML using the iLogix Rhapsody environment [12].

4.2 Deployment of Control

Functional units can be, in principle, reused over many generations of telecommunication applications and often they can be also included in different designs. Such different designs and successive generations of the same design practically differ in their control aspects (e.g. the protocols) but not in the basic functions performed. However, the same philosophy applies to control units, as well. Protocols are organized on different layers and from one product to another only particular layers are modified (upgraded) while other layers are reused. Therefore, in order to support reusability of functional and control units, a minimal amount of control has to be deployed to any processing unit in the model. By doing this, the possibility of reusing the modelling units increases significantly. Figure 4 shows the UML statechart for one of the functional units 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 baseband controller manages functional units according to the mode of the channel it controls. At the same time, it is completely unaware of the existence of the other channels, and it does not know anything about the time-slots structure of the access scheme (a statechart of a baseband controller is shown in Figure 8).

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 baseband controller (the statechart of the TRX is depicted in Figure 9).

4.3 Separation of Control and Data Flows

Another aspect which improves the reusability of the various elements of the model is the separation of data and

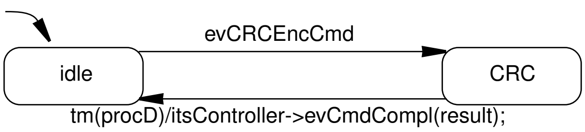


Figure 4. Statechart for CRC processing unit

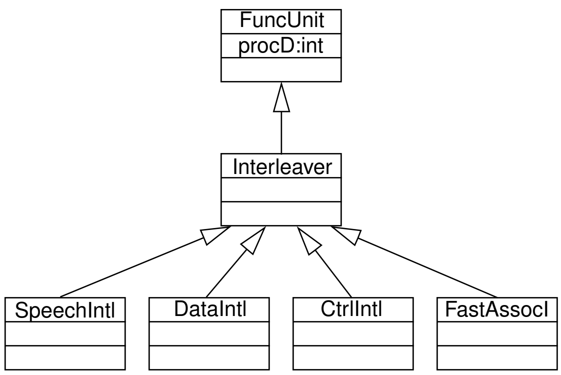


Figure 5. Generic functional units

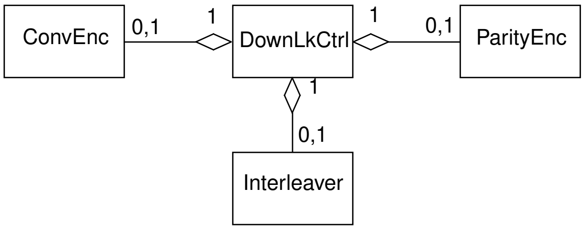


Figure 6. Structure of a downlink controller

control flows. Object-oriented 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. Functional units are operating on the data path, while no control functions are embedded within them. On the other hand, baseband controllers as well as TRXs do not perform any processing on the data flow. They just coordinate lower level entities (other controllers or functional units).

4.4 Uniform Interfaces and Generic Units

Various types of low level control or processing units are treated uniformly by the upper layers. This highly simplifies the modelling process. Such an uniformity is achieved by means of the inheritance relationship.

Figure 5 illustrates a class hierarchy for functional units. Particular interleavers are derived from a generic interleaver which again is a specialization of a generic functional unit. For the upper, control layers, every functional unit appears as this generic unit. A similar strategy also apply to control units.

Figure 6 shows the class diagram of a generic downlink baseband controller. The generic controller aggregates generic functional units of the classes Interleaver (see also Figure 5), ConvEnc, and ParityEnc. A specific BC aggregates specialized FUs, depending on the particular channel mode. A data downlink controller, for example, is an instantiation of the generic class DownLkCtrl, and aggregates an instantiation of the class DataConvEnc and of the class DataIntl (see also Figure 5). This class diagram is shown in Figure 7.

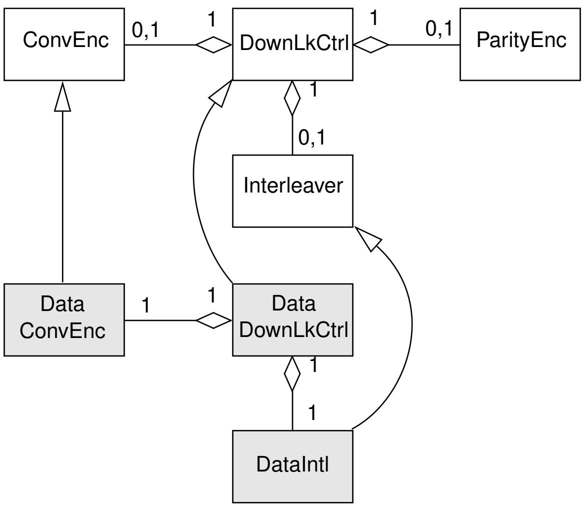


Figure 7. Specialization of a downlink controller

4.5 Timing Aspects

Specification of timing aspects is mandatory in order to verify timeliness and to perform architecture exploration. Modelling of such timing aspects comprises specification of deadlines and execution delays.

Execution delays are modelled as time-triggered state transitions. The parameter which characterizes the execution delay on a certain functional unit is specified as an attribute of that unit. During architecture exploration, this delay depends on the particular processor to which the unit is assigned for execution. This delay has to be estimated [11] and the resulting value is assigned to the unit. The delay attribute is specified in the superclass of all functional units (procD in Figure 5).

The controlling units are in charge of coordinating the functionality of lower lever units. They are not characterized by a processing delay in the sense the functional units are. However, 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. In Figure 8 we illustrate this mechanism for a baseband controller. If the execution delay of the processing chain exceeds the predefined value

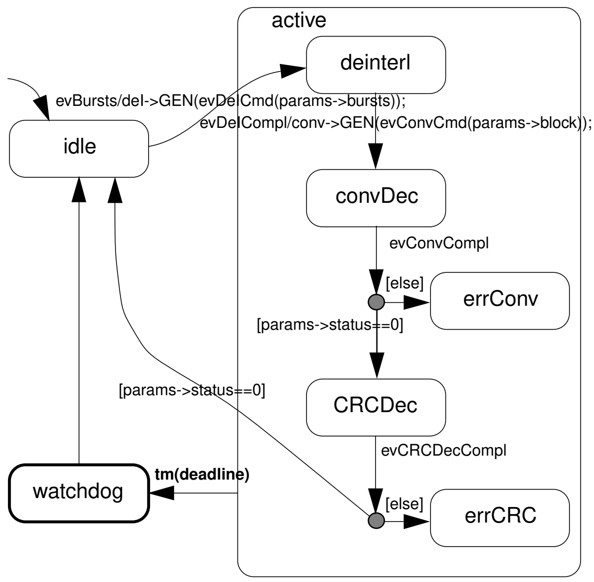


Figure 8. Statechart of a baseband controller

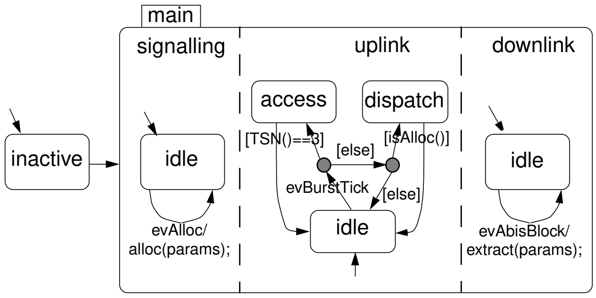


Figure 9. Statechart of a TRX

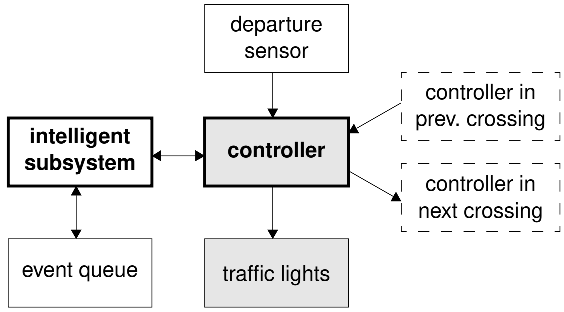


Figure 10. Intelligent traffic lights controller

of *deadline*, the watchdog state is entered. Of course, this is an exceptional situation.

4.6 Concurrency Issues

Usually, telecommunication devices, and the BTS makes no exception, perform a set of well specified operations on multiple data flows. Identification of this parallelism is important in the analysis phase because it influences dramatically the object structure of the model. In the UML, parallelism can be expressed in two ways. First, all the objects are considered to be parallel machines. Synchronization is easily 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 orthogonal (parallel) components. Such a case is shown in Figure 9, where the statechart of a TRX is depicted. The TRX has to handle both uplink and downlink traffic, as well as signalling information addressed to it. Those activities are independent to each other and can be performed concurrently.

4.7 Multiple Abstraction Levels

Our main focus was set on the specification and design 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 object-oriented approach, allows for an uniform treatment of entities specified at different abstraction levels. They permit us to concentrate on the refinement of that part of the model we are mainly interested in.

5. Architecture Exploration

In order to complete our goal in exploring the way UML suits the real-time systems design cycle, we imagined a simpler case study, which allows to easily demonstrate how architecture exploration can be performed. The example we have chosen is an intelligent, adaptive traffic lights controller (see Figure 10).

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, heading to the next crossing on the main road. This 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 will adjust the synchronization and traffic lights timing in order to have an optimal "green wave" on the main road, which adapts to the current traffic condition.

In Figure 10, the shaded blocks represent the core of the traffic lights controller. We considered that the controller is physically mapped on a microcontroller and its corresponding software, and the intelligent subsystem (which performs statistical analysis and adjusts the timing accordingly) is mapped on a processor. This mapping is illustrated by the UML deployment diagram in Figure 11.

A high-level object model diagram of the intelligent traffic lights controller is presented in Figure 12. The model was conceived in a way which allows simulation

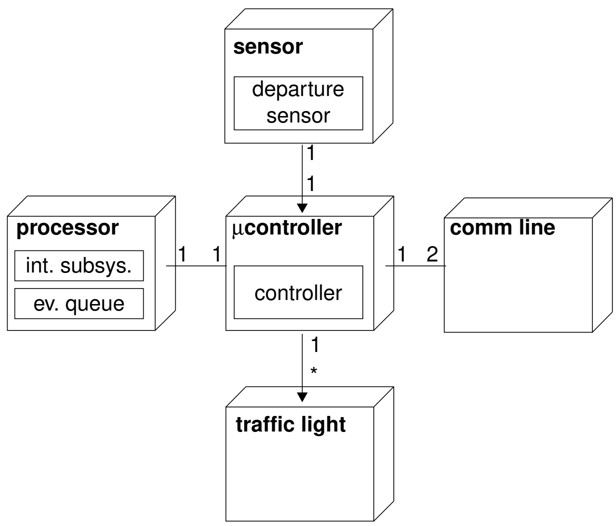


Figure 11. Deployment diagram

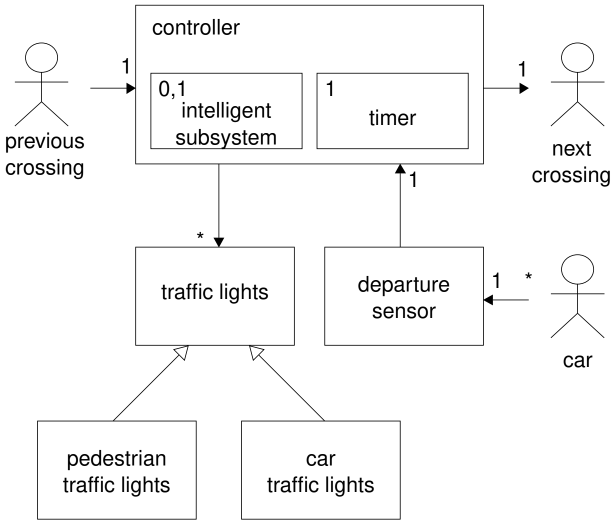


Figure 12. Object model diagram

with or without the intelligent subsystem.

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 intelligent subsystem, 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 and CE as follows:

$$PE = \frac{v_{min}}{v}$$

$$CE = \frac{\delta_{min}}{\delta}$$

where v is the execution time for the intelligent subsystem on the currently considered processor, δ is the communication delay for an instance of data transfer on the currently considered infrastructure for the communication lines, v_{min} is the value of v corresponding to the fastest processor in the module library, and δ_{min} is the value of δ corresponding to the fastest communication line in the module library.

The experimental results are presented in Table 1. We run the same scenario (i.e., input vectors), first for the traffic lights controller without the intelligent subsystem, and second for the traffic lights controller with intelligent subsystem, departure sensor and communication lines. For the case without an intelligent subsystem, the controller works by granting access to the tracks according to a round

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

Table 1. Measurements

robin strategy. For both settings we explored several values for PE and CE. As it can be seen from Table 1, the intelligent subsystem brings a significant increase in performance. The performance is degraded when PE and CE are decreased.

If a certain requirement on the average waiting time is given, such an experiment allows to select the most cost-effective architecture which still fulfils the imposed constraint.

6. Conclusions

We discussed several issues concerning the modelling of complex embedded real-time applications using an OO methodology with the UML.

We have concentrated on particular issues which are characteristic to the modelling of large telecommunication applications. 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 combination of generic units. The main objective is to facilitate complexity management during the modelling phase. At the same time, reusability (IP-based design) and architecture exploration are supported.

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

References

[1] R. Ernst, *"Codesign of Embedded Systems: Status and Trends"*, IEEE Design & Test of Computers, April-June, 1998, pp. 45-54

[2] J. Axelsson, *"Holistic Object-Oriented Modelling of Distributed Automotive Real-Time Control Applications"*, Proceedings of the 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 1999, pp. 85-92

[3] M. Keating, P. Bricaud, *"Reuse Methodology Manual for System-on-a-Chip Designs"*, Kluwer Academics Publishers, 1998

[4] A. Sarkar, R. Waxman, J. Cohoon, *"Specification-Modelling Methodologies for Reactive-System Design"*, in "High-Level System Modelling: Specification Languages", Kluwer Academics Publishers, 1995, pp. 1-34

[5] B. Douglass, *"Doing Hard Time"*, Addison-Wesley, 1999

[6] M. Mouly, M. Pautet, *"The GSM System for Mobile Communication"*, Palaiseau, 1992

[7] *"GSM 03.02"*, European Telecommunications Standards Institute (ETSI), http://www.etsi.org

[8] B. Selic, G. Gullekson, P. Ward, *"Real-Time Object-Oriented Modeling"*, John Wiley & Sons, 1994

[9] G. Booch, *"Object Oriented Design With Applications"*, The Benjamin/Cummings Publishing, 1991

[10] G. Booch, J. Rumbaugh, I. Jacobson, *"The Unified Modeling Language User Guide"*, Addison-Wesley, 1999

[11] J. Gong, D. Gajski, S. Bakshi, *"Software Estimation Using A Generic Processor Model"*, Proceedings of the European Design and Test Conference, 1995

[12] http://www.ilogix.com