REAL-TIME OBJECT-ORIENTED DESIGN AND FORMAL METHODS

Juan Antonio de la Puente
Dept. of Telematics Engineering
School of Telecommunication, Technical University of Madrid
E-mail: jpuente@dit.upm.es

1. Introduction

The increasing complexity of real-time systems call for advanced software engineering methods to be used for the development of this kind of systems. Object-orientation is a powerful approach to managing complexity, which has received widespread attention in the last years. Its use in real-time systems development, however, must be taken carefully, as the special characteristics of these systems, especially concurrency and deterministic time behaviour, may be difficult to deal with, as many intricate features may remain hidden in the object internals, and thus not available when the global behaviour of the system is being considered.

In the rest of this contribution, we propose to combine object-oriented models with formal methods in order to be able to reason about timing behaviour at the system level. The particular approach that we have used has been developed in the framework of the Esprit projects IPTES and IDERS, and the CICYT projects TAP93-0001-CP and TIC96-0614 that have been carried out at DIT/UPM. We will restrict ourselves to design level models, i.e. models that contain information about the structure of the system and the resources required to implement it, but no implementation code or other details.

2. Object-Oriented design methods

Object-oriented design (OOD) is based on a small number of basic concepts: objects, classes, operations and relationships. Some important relationships are inheritance, aggregation, and usage (see e.g. Booch 1994, Rumbaugh *et al.*, 1991). Different methods and notations have been proposed within this paradigm, and a *Unified Modelling Language* has been developed based on several of these methods.

Most of the developments in object-oriented design have been done with little or no provision to real-time requirements. Concurrency is often not considered, and timing requirements are simply non-existent in most OOD methods. Three notable exceptions are Octopus (Awad *et al.*, 1996), ROOM (Selic *et al.* 1994) and HRT-HOOD (Burns & Wellings 1995). The first two are mostly addressed to the design of soft real-time systems, and only HRT-HOOD provides a method for guaranteeing deterministic timing behaviour in hard real-time systems, based on preemptive priority scheduling theory. This is achieved by eliminating inheritance, introducing a small number of class stereotypes (active, passive, protected, cyclic and sporadic), and restricting the synchronization mechanisms for class operations. Aggregation (or inclusion) and usage relationships are also restricted in order to ensure that the timing properties of the final design are analysable.

Some of these restrictions can be softened while still keeping time determinism (de Miguel 1997a), but in general it seems impossible to keep all the flexibility of general-

purpose OOD methods in real-time systems. In the rest of the paper we will limit ourselves to HRT-HOOD as an example of real-time OOD method, with its current restrictions.

Validation of design models can be done in a number of ways. Model animation can be used to gain confidence on the design model, provided it is based on a graphic executable notation. A technique for the animation of structured models was developed in the IPTES (Pulli *et al.* 1993) and IDERS (Rendón *et al.*, 1995) projects, and has been later adapted to HRT-HOOD models (De Miguel 1997a). The Object-Time[1] tool for the ROOM method is another example of a tool supporting this technique.

3. Formal methods and object-oriented design models

The available object-oriented design methods are good at expressing structural properties of software systems, but lack expressive power for functionality and timing properties, which are often described in ordinary text. This precludes model analysis and even graphic animation, due to the lack of a well defined semantics.

An approach we successfully used with structured methods is combining intuitive, easy to use graphic notations with formal methods in order to provide semantics for the former that makes it possible to develop executable models (Rendón *et al.*1995). What we propose here is to use the same approach for OO models. In this way, the good mathematical properties of formal methods can be used in a way that is easier for developers, as they mostly deal only with a graphical notation, which is annotated with specifications of functional and non-functional behaviour.

The basic step is to develop an operational semantics of the graphic notation in terms of the formal notation. As an example, we have done this for HRT-HOOD using High-Level Time Petri Nets (HLTPN) as a formal notation. HLTPN (Felder *et al.* 1993*)* is an extension to Petri Nets in which different aspects of a system, such as concurrent execution, timing, data, and functionality, can be formally described.

We have applied this approach to an extension of HRT-HOOD (de Miguel *et al.* 1997b). Figure 1 shows an example of an HRT-HOOD design model, and figure 2 shows the way in which the implicit behaviour of a cyclic (periodic) object is described in terms of HLTPN. The same is done for other class stereotypes, and for operations of passive, protected, and active classes.

Temporal attributes are modelled by time stamps on tokens and firing intervals on transitions. In the example, the firing interval for the “cycle_deadline” transition is set so that if the transition is still enabled (i.e. there is a token in the “executing” place) after the specified interval has elapsed, then it fires thus putting a token in the “error” place.

[1] Object-Time is a registered trademark of Object Time Ltd.

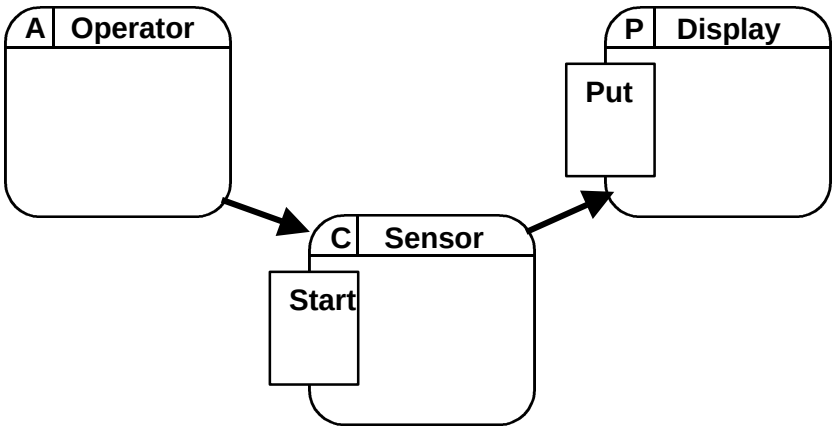

Figure1. Example HRT-HOOD design model.

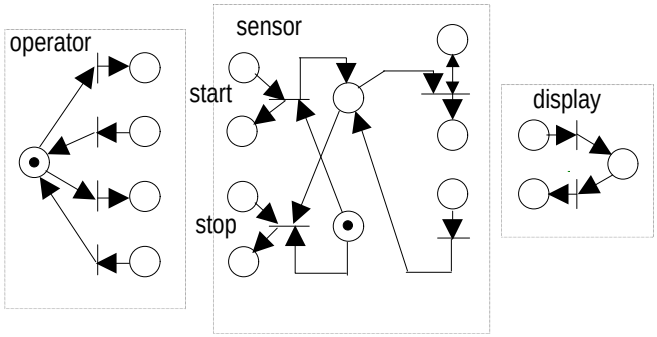

Figure2. Petri-net model of figure 1.

4. Design tool

In order to assess this approach, we have developed a design tool with the following features:

- Graphical editing of HRT-HOOD models
- Automatic generation of HLTPN skeletons from object stereotypes
- Support for time attributes in objects and operations
- Support for data and functional behaviour based on data-flow diagrams and a subset of C.
- Graphic animation of models
- Automatic generation of tables for schedulability analysis

The latter feature enables semi-automatic validation of temporal properties using a schedulability analysis tool.

5. Conclusions

Integrating formal methods and OO graphic notations is a promising technique with a significant list of advantages over conventional approaches:

- Better specification of concurrency, synchronization, and timing properties.
- Executable models enable validation of design models by graphic animation.

- Schedulability analysis can be performed on design models, provided some restrictions are kept.
- Design tools can be built that use formal notations as a kernel, hiding the details from the developer.

Some problems still remaining refer to the interaction between inheritance and concurrency. Another interesting extension is supporting parameterized classes.

Acknowledgements

I would like to acknowledge the work of the following members of the DIT/UPM Real-Time Software Engineering Group: Miguel Ángel de Miguel, Álvaro Rendón, Alejandro Alonso, Juan Carlos Dueñas, and Gonzalo León.

References

Awad, Kuusela and Ziegler. 1996. *Object Oriented Technology for Real-Time Systems.* Prentice-Hall.

Booch,G. 1994. *Object Oriented Design with Applications* (2nd ed.). Addison-Wesley.

Burns, A., and Wellings, A. 1995. *HRT-HOOD: A Structured Design Method for Hard Real-Time Systems.* Elsevier Science.

De Miguel, M.A., M., Dueñas, J.C., Rendón, A., de la Puente, J.A., Alonso, A., and León, G. 1996. Early Validation of Real-Time Systems by Model Execution. *Proceedings of the 13th World Congress of IFAC*. San Francisco, California, USA. Elsevier Science.

De Miguel, M.A. 1997. *Design of Real-Time Systems with Executable Objects*. PhD Thesis, Technical University of Madrid (In Spanish).

De Miguel, M.A., Alonso, A., and de la Puente, J.A. 1997. Object-Oriented Design of Real-Time Systems with Stereotypes. *Proc. 9th Euromicro Workshop on Real-Time Systems*. Toledo, Spain. IEEE CS Press.

Dueñas J.C., Rendón, A., and de Miguel, M.A. 1997. Integrated Validation of Real-Time System Models. *Proceedings of the 9th Euromicro Workshop on Real-Time Systems*. Toledo, Spain. IEEE CS Press.

Felder, M., Ghezzi, C., and Pezzé, M. 1993 High-Level Timed Petri Nets as a Kernel for Executable Specifications. *Real-Time Systems*, vol. 5, no. 2/3.

Pulli, P., Heikkinen, M., and Lintulampi, R. 1993. Graphical Animation as a Form of Prototyping Real-Time Software Systems. *Real-Time Systems*, vol. 5, no. 2/3.

Rendón, A., Dueñas, J.C., de Miguel, M.A. Leskelä, J., de la Puente, J.A., León, G., and Alonso, A. 1995. Animation of Heterogeneous Prototypes of Real-Time Systems. *Proc. 1st. IEEE Int. Conf. on Engineering of Complex Computer Systems - ICECCS'95.*Fort Lauderdale, Florida, USA. IEEE CS Press.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. 1991. *Object-Oriented Modeling and Design*. Prentice-Hall.

Rumbaugh, J., Jacobson, I., and Booch, G. 1999. *The Unified Modeling Language Reference Manual.* Addison-Wesley.

Selic,B., Gullekson,G., and Ward, P.T. 1994. *Real-Time Object-Oriented Modeling*. John Wiley and Sons.