Open access • Book Chapter • DOI:10.1007/978-3-642-88049-0_27

Issues in Object-Oriented Real-Time Language Design — Source link

Mehmet Aksit, Jan Bosch

Institutions: University of Twente

Published on: 01 Jan 1994 - NATO ASI RTC

Topics: Very high-level programming language, Low-level programming language, Inheritance (object-oriented programming), Specification language and Language primitive

Issues in Object-Oriented Real-Time Language Design

Mehmet Akşit and Jan Bosch

Department of Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

To express real-time applications, most real-time languages introduce specific constructs to specify features like *deadlines*, *periodic behavior* and *time intervals*. These constructs, in general, can be seen as annotations to conventional language structures. Object-oriented programming languages have gained popularity in non real-time applications. These languages are highly modular and provide protection through strictly encapsulated abstract data types called *objects*. In addition, *classes* and *inheritance* mechanisms enable programmers to reuse existing software. Recently, there have been some attempts to define real-time object-oriented languages . One motivation for introducing these languages is to reduce the complexity of applications through modularization so that *predictability* and *reliability* of applications can be increased. Secondly, inheritance mechanisms allow reuse of software modules that have well-defined behavior. This may simplify analysis for a particular real-time application. Thirdly, since object-oriented languages are now more frequently applied to software implementations, it would not be practical to adopt different languages for real-time and general parts of an application. However, there are several issues to be addressed in order to fully utilize object-orientation in real-time applications. Firstly, real-time specifications must be reused separately from the 'application code'. This allows the reuse of classes in applications with different real-time behavior. Otherwise, changes made to the application requirements or real-time specifications in the sub-classes may result in excessive redefinitions of super-classes although this seems to be intuitively unnecessary. This we refer to as the *real-time specification anomaly*. Secondly, since a subclass may extend, exclude or replace the real-time specifications of its super-classes, semantics of inheritance must be clearly defined. Thirdly, there must be language mechanisms to modularly specify and reuse alternative implementations. For example, inter-object interactions often result in blocking execution threads. Blocking time can be minimized using dedicated strategies. This requires abstraction of inter-object communications and large scale synchronization among objects as first class objects. In addition, an object may adopt different implementations for its public interface. Lastly, all these language mechanisms must be uniformly integrated in a single consistent framework. We believe that the conventional object-oriented model is far too restricted to fulfil these real-time requirements. Language annotations made to conventional languages may result in real-time specification anomalies and non-uniform language constructs. At the University of Twente, we have been working on new object-oriented language mechanisms using the *composition-filters* approach [1]. Composition filters affect the received and sent

messages to or from an object. By proper configuration of filters, one can specify inheritance, delegation, inter-object communications and "real-time constraints" in a single framework.

References

1. M. Akşit, L. Bergmans, S. Vural, *An object-oriented language-database integration model: The composition filters approach,* ECOOP'92 Conference Proceedings, Springer-Verlag, Vol. LNCS–615, June 1992, 372–395.