Object-Oriented Programming and Protected Objects in Ada 95

Andy J. Wellings¹, Bob Johnson², Bo Sanden², Joerg Kienzle³, Thomas Wolf³, and Stephen Michell⁴

¹ Department of Computer Science, University of York, UK
² Department of Computer Science, Colorado Technical University, USA
³ Software Engineering Laboratory
Swiss Federal Institute of Technology in Lausanne, Switzerland
⁴ Maurya Software, Ontario, Canada

Abstract. Integrating concurrent and object-oriented programming has been an active research topic since the late 1980s. The majority of approaches have taken a sequential object-oriented language and made it concurrent. A few approaches have taken a concurrent language and made it object-oriented. The most important of this latter class is the Ada 95 language which is an extension to the object-based concurrent programming language Ada 83. Arguably, Ada 95 does not fully integrate its models of concurrency and object-oriented programming. This paper discusses ways in which protected objects can be made more extensible.

Keywords: concurrent object-oriented programming, inheritance anomaly, Ada 95.

1 Introduction

Arguably, Ada 95 does not fully integrate its models of concurrent and object-oriented programming [8,2]. For example, neither tasks nor protected objects are extensible. When Ada 95 was designed, the extensions to Ada 83 for object-oriented programming were, for the most part, considered separate to extensions to the concurrency model. Some consideration was given to abandoning protected types and instead using Java-like synchronised methods in their place, however, there was no public debate of this issue. Similarly, there was no public debate on the issues associated with allowing protected types or tasks to be extended, although a Language Study Note (LSN-1030) was produced which raised some of the issues discussed in this paper. The purpose of this paper is to discuss ways in which the Ada 95 concurrency model can be better integrated with object-oriented programming.

* This paper extends and unifies the approaches described in [3] and [6].
2 Concurrent Object-Oriented Programming

Integrating concurrent and object-oriented programming has been an active research topic since the late 1980s. The majority of approaches have taken a sequential object-oriented language and made it concurrent. A few approaches have taken a concurrent language and made it object-oriented. The most important of this latter class is the Ada 95 language which is an extension to the object-based concurrent programming language Ada 83.

In general, there are two main issues for concurrent object-oriented programming:

- the relationship between concurrent activities and objects – here the distinction is often between the concept of an active object, and where concurrent execution is created by the use of asynchronous method calls,
- the way in which concurrent activities communicate and synchronise (and yet avoid the so-called inheritance anomaly [4]).

3 The Ada 95 Programming Language

The Ada 83 language allowed programs to be constructed from several basic building blocks: packages, subprograms, and tasks. Of these, only tasks were considered to be types and integrated with the typing model of the language. Ada 95 extends the facilities of Ada 83 in areas of the language where weaknesses were perceived. Two of the main innovations were the introduction of data-oriented communication and synchronization through protected types and the introduction of object-oriented programming facilities through tagged types.

Although task types and protected types are fully integrated into the typing model of Ada 95, it is not possible to create a tagged protected type or a tagged task type. The designers shied away from this possibility partly because they felt that fully integrating object-oriented programming and concurrency was not a well-understood topic and, therefore, not suitable for an ISO standard professional programming language. Also, there were inevitable concerns that the scope of potential language changes being proposed was too large for the Ada community to accept. In spite of this, there is some level of integration between tagged types and tasks and protected objects. Tagged types after all are just part of the typing mechanism and therefore can be used by protected types and tasks types in the same way as other types. Indeed paradigms for their use have been developed (see [2] chapter 13). However, these approaches cannot get around the basic limitation that protected types and task types cannot be extended.

4 Making Concurrent Programming in Ada 95 more Object-Oriented

There are the following classes of basic types in Ada: scalar types - such as integer types, enumeration types, real types, etc, structured types - such as record types