Proceedings of the 1983
Winter Simulation Conference
S. Roberts, J. Banks, B. Schmeiser (eds.)

Simulation Programming with Ada

Wilhelm F. Bürger
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

This tutorial session deals with features of ADA which are of interest for formulating simulation problems. A general overview of ADA is given with emphasis on packages and tasking. A simulation package for discrete event simulation will be sketched.

1. Introduction

Ada is a new general purpose programming language which was developed for the Department of Defense. It was designed in response to the proliferation of languages currently in use at the Department of Defense. A series of documents refined the requirements for the language design. The final language is based on the 'Steelman' requirements [1]. The Ada language was standardized in 1983, and the defining document is [2]. Ada is a registered trademark of the United States Government, Department of Defense, Under Secretary for Research and Engineering.

Ada is designed to support the development of very large software systems. It is intended to be used for a broad spectrum of applications. The language supports real time processing and concurrent processing which are needed for the programming of embedded systems. Efforts are underway to define and standardize also the Ada environment [3].

We first give an overview of the language facilities, and then we identify the useful features of the language with respect to simulation. These features will be demonstrated in the talk with a simulation package for discrete event simulation [4].

2. Language Overview

In order to achieve its design goals Ada provides the following concepts and features:

- strong typing
- control structures to support structured programming
- overloading
- separate compilation and libraries
- tasking
- exception handling
- generic program units

An Ada program consists of one or more units which can be separately compiled. Units are subprograms, packages, tasks, and generics. A unit consists of a specification part and an implementation part each of which can be compiled separately. This feature allows the creation of independent components which can be used to build a program.

Packages are the major tool for program modularity. They correspond to some extent to Simula classes [5]. Packages are used in three ways: as named collections of declarations, as a facility to group related functions and procedures which share internal data and types, and for encapsulating data types and associated operations. Encapsulation provides information hiding so that other components cannot depend on details of structure and implementation of the encapsulated type. Components therefore can be designed in such a way that they are 'plug-compatible' with other components without the need for any program modifications.

In order to deal with real-time issues and concurrent processing Ada provides the tasking concept. Tasks can be thought of as processes running on their own logical processor in parallel to other tasks. Synchronization of tasks is achieved by the 'rendezvous' mechanism. Task types are provided so that tasks can be created and disposed of in a dynamic way. The task facilities are also used for managing concurrent access to a shared resource.

Ada enforces strong typing. An operation, procedure or function can be only applied to an object which is of the same type for which the operation, procedure or function

is defined. However, generalized algorithms which are applicable to a variety of types can be formulated through the use of generic program units. Further, names and operators can be overloaded with the definition of different algorithms for different types.

The data type definition facilities of Ada are similar to Pascal's type definition facilities. Types can be classified into scalar types, composite types, access types, and private types. The scalar type includes numeric types for fixed point numbers and floating point numbers. These types can be specified, respectively, with fixed or relative error bounds.

Ada does not have language features for input and output but instead provides predefined packages for input and output. The specification part of these packages is defined as part of Ada employing the full generality of the language like subprogram overloading and generic instantiation.

Several books on Ada are now available. Recommended reading is *Programming in Ada* [6] by J.G.P. Barnes, one of the original authors of Ada. The language reference manual [2] is indispensable, however, to appreciate the full language.

3. Ada as a Simulation Language

The basic facilities usually provided by a simulation tool are:

- process / event handling
- list and queue handling
- random numbers and distribution functions
- statistic gathering and reporting of results

Depending on the point of view and the simulation system at hand discrete simulation is formulated either in a 'process-oriented' way, or in an 'event-oriented' way. In the first case a process describes a specific activity of a real system that evolves concurrently with other activities. In the second case events are associated with state changes which schedule other events. Simula [5] is representative of the first case, SIMPAS [7] of the second.

Ada's tasking feature provides the basic facilities for process oriented simulation. By using appropriate tasking idioms complex interactions between processes can be formulated. Tasking can be used also with advantage for the managing of the event queue as a shared resource in an event oriented approach. The packaging facility of Ada allows the extension of Ada to new domains - in the simulation context generic program units provide the simulation primitives.

In a likewise fashion the package concept can be used to define abstract data types which represent 'entities'. Entities describe the objects which are manipulated in a simulation. They usually have attributes which gather information to produce later a simulation report. The list and queue handling functions also can be provided in a generic way for these objects. Furthermore objects which exhibit dynamic behavior can be formulated with access types.

Random number generation and a variety of distribution functions can be supplied by an appropriate library package. The gathering of data, and the scheduling of events can be made numerically quite precise through the fixed and floating point number definition capabilities of Ada. Packages for producing a standard report of a simulation run may be also available from a library. As the user works in the environment of a general purpose language, new output facilities specifically tailored to the problem at hand can be integrated with the available report generation services.

The package facilities of Ada can be used to provide several simulation environments. For example the SIMSET and SIMULATION class facilities of Simula can be made available in Ada in a natural fashion. It should be possible to imbed other simulation environments into Ada in a similar way.

4. Conclusion

The packaging and tasking facilities of Ada provide the necessary features to make Ada a useful simulation language. The package concept allows extensions of Ada to new application domains. For example a library of simulation components could be created from which a model builder could choose the appropriate components to rapidly build new simulation models. Simulation thus would be a readily available tool for a decision making environment. Further the combination of simulation primitives and database components together with other application domain specific packages could become a basis for 'modeling methodologies in the large' [8].

References

[1] *Department of Defense Requirements for High Order Computer Programming Languages - 'STEELMAN'*, Defense Advanced Research Projects Agency, June 1978.

[2] *Ada Programming Language ANSI/MIL-STD-1815A*, United States Department of Defense, January 1983.

[3] *Department of Defense Requirements for Ada Programming Support Environments - 'STONEMAN'*, Defense Advanced Research Projects Agency, February 1980.

[4] Bruno, G., *An Ada Package for Discrete Event Simulation*, Proceedings of the AdaTec Conference on Ada, Arlington, VA., 1982.

[5] Dahl O.J., et al., *Simula 67 Common Base Language*, No. 5-22, Norwegian Computing Center, 1970.

[6] Barnes, J.G.P., *Programming in Ada*, Addison-Wesley, 1982.

[7] Bryant, R.M., *SIMPAS - A Simulation Language Based on Pascal*, 1980 Winter Simulation Conference.

[8] Zeigler, B.P, *Impact of General Systems Orientation: Present and Future*, 1981 Winter Simulation Conference.