REUSABLE SOFTWARE COMPONENTS

Trudy Levine
Fairleigh Dickinson University
Teaneck, NJ 07666
levine@fdu.edu
http://alpha.fdu.edu/~levine/reuse_course/columns

There has been a continued interest in software reuse in the software engineering community, as evidenced by web services and Microsoft .NET, and by the continued work in product-line engineering. New users to software reuse should be made aware of the work that has been done by the Ada community and of the particular appropriateness of Ada for reuse. We therefore review in this column the issues involved in reuse with the Ada programming language. For those who are interested in investigating these topics in depth, we enclose a brief list of helpful sources. We will post this column on our web site, and gladly append any additional references that our readers may supply.

"Reusability [is the] capability of a software component to be used again or used repeatedly in applications other than the one for which it was originally developed. In order to be effectively used, the component, may have to be adapted to the requirements of the new application." [1] In this column, we have repeatedly defined a component as any software asset, including subsystems, documentation, testing suites, etc. Adapting for reuse includes maintenance (modifications for different times), portability (modifications for different systems), and generality (modifications for different applications). [8] Characteristics of systems that support reuse include mechanisms for high levels of abstraction, quality, modularity, cohesion, encapsulation and inheritance, as well as low coupling and machine independence. All of these traits are, of course, desirable in software engineering in general, and were goals in the original design of Ada. (Ada83 supported static inheritance only, but Ada95 includes constructs for dynamic inheritance.)

It has been well documented that Ada programming assists in the reuse process [see, for example, 3,4,5,6,7,9,10,11], with its object oriented capabilities, package construct, generics, entity reusability, private types, overloading, consistent interface structure, strong typing for dimensional analysis, tagged record types, overloading, exception handling, and software repository management. Although all of the above constructs are important for the reusability of components, Ada's object oriented capabilities strongly assist in the implementation of reusable subsystems. We particularly stress that careful use of Ada assists in the development of high quality components and subsystems for reuse. Software becomes more valuable with increased verbatim reuse, because of the additional return on the original development investment. Reliability and confidence also increase, as components are repeatedly used (and thus tested) in various system contexts.

Greg Bowen has found that "both Ada's strong typing and specification constructs address some of the practical aspects of reuse that are missing in many other languages. 1) All possible uses of a reuse asset are unknown at development, but these constructs

allow the compiler-enforced definition of the usage profile and help to assure that assets are used in the manner they were intended. 2) Testing all possibilities would be impractical, and often impossible, but Ada's strong typing reduces whole classes of errors and facilitates development of practical test plans. 3) As the number of uses of an asset increases, the importance of controlling interface changes, and related side-effects to software (or users) dependent on those interfaces, also increases. However, Ada's consistency checking prevents uncontrolled changes and unpredictable side-effects, which would reduce (re)usability. 4) Interface change issues are compounded with "just-in-time" reuse, where assets need to be developed concurrent with the software using those assets, because interface specifications tend to be less stable. Ada (coupled with peer review and baseline practices), enables concurrent development through the separation of design from implementation, backed-up by compiler enforcement of the (latest) agreed upon specification."

Ada is particularly helpful in teaching reuse [2]. In addition to the above features mentioned, Ada's orthogonal structure and comparative simplicity make it ideal for courses such as Data Structures or Software Engineering. Ada83 and Ada95 provide the encapsulation necessary for implementing abstract data types. Ada's object-oriented capabilities support the transparent use of objects of these types.

Some of the high quality, domain-specific, Ada libraries that were built with Ada included COSMIC and AdaNet, sponsored by the National Aeronautics and Space Administration, CARDS and the RAASP program, sponsored by the Air Force, and the STARS Program, sponsored by DARPA. Of importance, as well, was the work by the Reuse Interoperability Group, for the establishment of a software library.

This column has been dedicated to a new generation of programmers. We hope that it will be helpful in referencing the extensive work that has been done in reuse with Ada.

References:

1) Bardin, B.M. and Thompson, C. J., *Composable Ada Software Components and the Re-Export Paradigm*, Ada Letters, Vol. VIII (1), Jan/Feb. 1988, pp.58-79.

2) Battaglia, D., Burke, A, and Beidler, J., *An Ada Reuse Support Systems for Windows 95/NT* Ada Letters Vol. XVIII (1) Jan/Feb 1998,pp. 86-91.

3) Booth, E.W. and Stark, M. E. *Using Ada to Maximize Verbatim Software Reuse*, M. Stark and E. Booth, *Proceedings of Tri-Ada 1989*, October 1989, available at http://sel.gsfc.nasa.gov/website/documents/sec08.htm#8.23

4) Booth, E.W. and Stark, M.E. *Designing Configurable Software: COMPASS Implementation Concepts*, Proceedings of Tri-Ada *1991*, October 1991 available at http://sel.gsfc.nasa.gov/website/documents/sec08.htm#8.35

5) Bowen, G. M., *An Organized, Devoted, Project-Wide Reuse Effort*, (presented at ACM *TRIAda'91* San Jose, California, October, 1991) *Ada Letters*, Vol. XII, No. 1, Jan/Feb 1992, pp. 43-52.

6) Condon, S, Hendrick, R., Stark, M. and Steger, W. *The Generalized Support Software (GSS) Domain Engineering Process: An Object-Oriented Implementation and Reuse Success at Goddard Space Flight Center*, available at http://sel.gsfc.nasa.gov/website/documents/sec08.htm#8.42

7) Levine. G. Ada in the University: Data Structures with Ada, Proceedings of the Fourth Annual Ada Software Engineering Education and Training Symposium, June 1989.

8) Levine, G. *A Model for Software Reuse*, OOPSLA, San Diego, CA Oct. 1996, available at http://alpha.fdu.edu/~levine/reuse_course/model.txt

9) *Impact of Ada and Object-Oriented Design in the Flight Dynamics Division at NASA/GSFC*, SEL-95-001, March 1995, available at http://sel.gsfc.nasa.gov/website/documents/online-doc.htm.

10) Sanden, Bo, Software *Systems Construction with Examples in Ada*, Prentice Hall, Inc. 1994.

11) Smith, M. A. *Object-Oriented Software in Ada 95*, International Thomson Computer Press, 1996.