Criteria for Evaluating Reuse Support in Education Courses

Ben R. Whittle

Dept. of Computer Science, University of Wales, Aberystwyth,
Aberystwyth
Dyfed, Wales, UK
Tel: (+44) 970 622450
Fax: (+44) 970 622455
Email: brw@aber.ac.uk

Abstract

This position paper discusses the criteria that could be used to examine support techniques for a reuse course based upon traditional lecturing principles. A prototype environment is described that is being developed to support reuse courses at UW Aberystwyth. This development is part of the larger TIPSE teaching environment project.

Keywords: Component Reuse, Component Description, Technology Transfer, Reuse Education.

Workshop Goals: Attendance of the workshop will provide me with the opportunity to ascertain and understand the state of the art in reuse research and practice. I will be able to bring this knowledge to bear on my own research, the projects I am involved in, and report it to the UK through the British Computer Society Special Interest Group on Reuse, who's newsletter I edit. The workshop will give me a chance to explain and obtain feedback on the ideas in my PhD thesis which has recently been submitted.

Working Groups: Design guidelines for reuse, Reuse and OO methods, Tools and environments, Education.

1 Background

The University of Wales, Aberystwyth (UWA) has a long record of reuse research including the Alvey funded project ECLIPSE [1], and the European Esprit funded project DRAGON [2]. Part of this work at UWA included that of Gautier, Ratcliffe and Shah on component description languages [3, 4]. UWA involvement in environments research has been particularly influential in the formation of a teaching environment project, the TIPSE [5, 6]. The work in my PhD thesis combines these two fields, looking at the design of an appropriate a component description language, which I call the Component Interface DEscriptoR (CIDER), and an associated toolset that could be used as a way of introducing students to the ideas of reuse. Rather than being an academics toy, this language, based on the 3C model of the reusable component [7, 8, 9], provides an interesting insight into the use of object-oriented software description for reuse.

In recent years, reuse education has become an important issue. At the 1992 Reuse Education Workshop, Mosemann [10] made the comment that reuse education is:

> one of our major hurd if reuse is to become common in the lifecycle of our systems.

In a similar way to reuse itself, two schools have developed; these can be termed 'education with reuse' and 'education for reuse'. That is, should people be taught reuse as part of the existing modules, or should reuse be taught separately. A lack of agreement on definitions and the turbulen of an evolving discipline only add to this debate. The position presented in this paper stems from ideas formulated during the development of a reuse technique suitable for both education with reuse and education for reuse.

Rather than describe the development of the CIDER language and its environment, a subject that will be covered in other academic publications, this position paper will be used to outline:

- a set of pragmatic guidelines for choosing techniques to be used in reuse education,
- the reasons why I think that a component description language and its associated environment is particularly appropriate for this.

2 Position

2.1 Capturing Reuse Principles for Education

This section discusses the criteria that could be used to examine support techniques for a reuse course based upon traditional lecturing principles. These criteria are summarised at the end of the section.

In an educational environment specialising in the training of software engineers, it is important to stress the underlying principles of programming over and above the practice and idiosyncrasy of one language or method. These principles can be reinforced by practical use and experience of example techniques. The example technique must transparently embody the principles of the research. It would also be an advantage if the technique was widely used in practice and contained state of the art research work[1].

[1]It is unlikely that a language that is widely used will contain recent research.

In industrial use a reuse language that is used as part of a design process or method must have (Krueger [11]) as small a cognitive distance as possible. Cognitive distance is the amount of intellectual effort that a software designer needs to understand a software system so that, for example, it may be taken from one stage of development to the next. Cognitive distance is correlated to the reusability of a component. The idea of understandability can also be applied to the description language itself, that is the language should itself present little problem for the student to understand.

Having said that the principles of reuse are abstract, it is necessary to demonstrate these principles through an exemplary concrete medium, in this case the reuse language. The area chosen to demonstrate a technique should be complete or a conceptually complete subset of a larger model. It should build on existing techniques, for example using the language with which the students are most familiar. Students must be able to see that the technique is of use. To satisfy this latter goal the technique should be appropriate for use within small projects. For example, a student should be able to reuse components or groups of components using the technique and follow this through to the implementation languages with which he is familiar.

Languages are an area that the student/engineer understands as most of their practical experience will involve the use of programming languages that are used to 'describe' or implement algorithms. One could argue that a language based component description technique could be used to represent software for reuse. Indeed the language used could be the same as that used in practical programming exercises, however it is widely agreed that, as Cramer *et al.* state in [12], programming languages are not suitable for component description for reuse.

In summary, the technique used to encourage reuse should:

- transparently embody the principles of reuse,
- ideally be widely used in practice and contain state of the art research work,
- not be at the implementation level,
- be easy to understand in itself,
- present a complete, workable system,
- relate to the students' other work, and thus demonstrate the applicability of reuse.

2.2 A Component Description Language & Environment as an Example of Reuse

In this section I describe a prototype environment that is being developed to support reuse courses at UW Aberystwyth. This development is part of the larger TIPSE teaching environment project.

The approach to reuse education proposed in this position statement is a traditional lecture course backed up with practicals. I envisage a lecture course covering the key principles of reuse at different levels of the lifecycle, together with an explanation of the tools and techniques that can be used to achieve this. This course would be applicable to advanced undergraduates, postgraduates and could be collapsed into a one week intensive course for industry.

The component description environment will be used as the practical element of the course. This gives the course participant the chance to use some of the tools and techniques, learning their use first hand. The most important part of the environment is support for a design level language for component description. Using the design level ensures that the student does not immediately think of the reusable component as source code, but as a representation of an idea, and as a relationship between a set of other ideas. Around this central reusable component representation and it's associated editor, there are reuse tools allowing the reuser to browse, select and edit reusable components in textual and graphical form. Finally there are a series of source code language editors and translation programs allowing a mapping too and from the design level description.

Going back to the summary in the section above, we can see how the component description and environment I have just described will provide a support for reuse education.

- **transparently embody the principles of reuse.** The principles of reuse are perhaps a matter of philosophical debate. However the language I have briefly introduced implements the 3C model of the reusable component, and the environment contains tools which display the key ideas of reuse, cataloging, browsing, selecting, and configuring components.
- **ideally be widely used in practice and contain state of the art research work.** The environment is extensible and can therefore contain both those tools and techniques widely used in practice together with recent research.
- **not be at the implementation level.** The component language is at the design level.
- **be easy to understand in itself.** The component description language has been designed to be orthogonal, with a relatively small set of constructs. Each language mechanism and construct has a well defined syntax and distinct semantic purpose.
- **present a complete, workable system.** The combination of the component description language with the environment tools gives a complete solution. The small scale of the toolset insulates the student from the learning curve of larger industrial strength environments. The environment has a similar look and feel to the support environment that the students will have used on other courses.
- **relate to the students' other work, and thus demonstrate the applicability of reuse.** The provision of the mapping programs enable a student to transfer the component descriptions into implementation language scripts, in our case C++ and Ada. This translation enables the student to relate the design level reuse work to source code in languages that they know and understand. The ability to produce more than one language is an advantage, clearly demonstrating that reuse is not language specific, and enabling students to see the relationship between semantically similar mechanisms in different languages.

3 Comparison

In the sections above I have presented my ideas for the evaluation of reuse course support material, and suggested one approach to this problem, a component description language with a support environment. In this section I will briefly discuss some of the other approaches to reuse education. A direct technical comparison is not possible as the approaches differ in the method of teaching. This comparison serves to highlight the differences between the approaches, and the reason for my position.

The failure to adopt a widespread practice of software reuse must in part be attributed to the educators. The education of tomorrows software engineers must tread a fine line between educating for the tools and skills of today, and enlightening and explaining the techniques of tomorrow. Despite the large body of research into reuse and the large number of projects in this area (see [13, section 1.3] for example), there is only evidence of a few entire course modules devoted to reuse. For example, Gray [14] talks about a practical course, using Ada for reuse based software engineering. However the majority of courses are for advanced students, such as the seminar module run by Bill Frakes at the Software Engineering Guild. This is similar to the module at the Norwegian Institute of Technology in Trondheim [15], resulting from their involvement in the REBOOT project. Another is a nine lecture course module at the University of Durham, that concerns Software Design and Reuse [2]. Other courses mention reuse, in passing, but with little practice of the techniques. Whilst it is certain that the methods and models necessary to teach reuse provide a wealth of available and acceptable material, see [16] for example, reuse course modules do not exist. This is due, in part, to the lack of industrial acceptance of reuse, but must surely be short sighted in educational terms.

The position taken here is that a traditional approach with lecture courses will be necessary in some institutions and for industry, where the aim of the course is to relay information rather than encourage debate. This lecture based approach will not be sufficient without a supporting base of practical examples of reuse. These practical examples can be offered as part of an integarted teaching environment, reducing the students learning curve and encouraging reuse as a standard part of the programming experience.

One of the key arguements that many people raise against reuse education is that the reuse domain is not yet mature or well enough defined to support this. I submit that at the level of detailed design and component reuse, which has been developing for over 25 years, the domain is mature enough to support a the development of a course in the fundamentals of reuse.

References

[1] M. F. Bott, ed., *Eclipse — An Integrated Project Support Environment.* Peter Pereginus, Stevenage, England, 1989.

[2] A. DiMaio, C. Cardigno, R. Bayan, C. Destombes, and C. Atkinson, “DRAGOON — An Ada based object-oriented language for concurrent, real time, distributed systems,” in *Proceedings of Ada Europe Conference, Madrid*, 1989.

[3] M. Ratcliffe, C. Wang, R. Gautier, and B. Whittle, “Dora — a structure oriented environment generator,” *IEE BCS Software Engineering Journal*, vol. 7, May 1992.

[4] I. Shah, *Abstract Specifications & Programming Language Bindings.* PhD thesis, University of Wales, Aberystwyth, 1990.

[5] M. Ratcliffe, M. Bott, T. Stotter-Brooks, and B. Whittle, “The TIPSE: An IPSE for teaching,” *BCS/IEE Software Engineering Journal*, vol. 7, September 1992.

[6] M. Ratcliffe, B. R. Whittle, M. Bott, and T. Stotter-Brooks, “The TIPSE: An educational support environment,” in *The 11th Annual National Conference on Ada Technology, Williamsburg, VA*, pp. 97–111, March 15-18th 1993.

[2] This description of Boldyreff's course is taken from a news message in a summary to a request for reuse textbooks posted by news@cybernet.cse.fau.edu on 12th February 1993

[7] W. Tracz, "The impact of domain analysis on software reuse," in *First International Workshop on Software Reusability, Dortmund, Germany, Universität Dortmund SWT memo Nr.57*, pp. 180–186, 1991.

[8] W. Tracz, *Formal Specification of Parameterized Programs in LILEANNA*. PhD thesis, Stanford University, to appear, 4th draft 1992.

[9] L. Latour, T. Wheeler, and B. Frakes, "Descriptive and prescriptive aspects of the 3Cs model: Seta1 working group summary," in *Proceedings of the Third Annual Workshop, Methods and tools for Reuse, CASE Centre Technical Report number 9014, Syracuse University*, June 1990.

[10] L. Mosemann, "The reuse challenge: Education," in *Proceedings of the Reuse Education Workshop, 23-24th September 1992*, pp. 12–17, September 1992.

[11] C. Krueger, "Software reuse," *ACM Computing Surveys*, vol. 24, June 1992.

[12] J. Cramer, W. Fey, M. Goedicke, and M. Grobe-Rhode, "Towards a formally based component description language — a foundation for reuse.," *Structured Programming*, vol. 12, pp. 91–110, 1991.

[13] J. Hooper and R. Chester, *Software Reuse: Guidelines and Methods*. Plenum Press, 1991.

[14] J. Gray, "Teaching the second computer science course in a Reuse based setting," in *The 11th Annual National Conference on Ada Technology, Williamsburg, VA*, pp. 38–45, March 15-18th 1993.

[15] G. Sindre, E.-A. Karlsson, and T. S. lhane, "Software reuse in an educational perspective," in *Proceedings of the 6th International Conference on Software Engineering Education, San Diego, 5–7 Oct.*, Springer Verlag, 1992.

[16] F. VanScoy, "Software reuse in computer science courses, *working group summary," in* Proceedings of the Reuse Education Workshop, 23-24th September 1992*, pp. 21–29, September 1992.*

[17] R. Prieto-Diaz, W. Schäfer, J. Cramer, and S. Wolf, eds., First International Workshop on Software Reusability, Dortmund, Germany, Universität Dortmund SWT memo Nr.57*, June 3-5th 1991.*

[18] C. Lillie, "Proceedings of the reuse education workshop, 23-24th september 1992," tech. rep., West Virginia University, January 27th 1993.

4 Biography

Ben Whittle graduated in Agricultural Economics from UW Aberystwyth in 1989. He subsequently completed a masters in Computer Science and was invited to proceed to research for a PhD in Software Engineering. Foremost among Mr Whittle's research interests are component reuse and reuse education, these subject provide the basis of his recently submitted PhD thesis. He is also interested in teaching support, and has been active in the development of the Aberystwyth teaching environment, the TIPSE. Mr Whittle is a member of the British Computer Society Reuse special interest group committee and the editor of the group newsletter.