Tools to Facilitate the Reuse Process

Rebecca Joos

Motorola
6501 William Cannon Drive
Austin, Texas 78735
MD: OE112
Tel: (512) 891-3617
Email: beckyj@pets.sps.mot.com

Abstract

Once an organization has implemented a process that embodies software reuse similar to the process discussed in this position paper, tools are necessary to facilitate software development. For several years Motorola (and specifically the author) has been involved with research that is the base for technology that will facilitate the development and use of reusable software components (RSCs) [5, 6].

1 Position

Software developers must design with and for reuse. Rather than trying to find a reusable component that matches a design module at the coding phase of the software life cycle, software developers should look for reusable components at the specification phase and continue the process through the analysis, design, coding, and testing phases. Reusable components can and should be specifications, analysis procedures and data, designs, test programs, etc. All experiences from concepts to algorithms involved with the software development should be captured for future reference i.e., reuse.

The basic premise of the reuse technology is to assist software developers with the more difficult activities of software reuse while automating the mundane and tedious tasks. In order to achieve this goal the reuse toolkit (RTk) will provide:

- intelligent assistance that helps the software developer choose the appropriate RSCs to incorporate into the new product,
- an interface that is easy and quick to use yet enables the developer to provide exact input necessary for the RTk to select appropriate RSCs,
- a simple method for adding new RSCs,
- automation to relieve the software developer of the burden of linking RSCs into the new product, and
- a mechanism for viewing the new product at several different levels of detail.

1.1 Designing for Reuse

Building new products with RSCs assumes that there exists a wealth of components. This is a simple assumption but not a simple task [4].

1.1.1 Reclaiming

Many companies have their valuable product knowledge encoded in software systems that are undecipherable. This knowledge represents the company's competitive advantage and expertise in their product area. If this knowledge was easily gleaned from the software, new products could be developed quickly and profitably. The activity of finding and developing these RSCs is tedious and time consuming.

In response to this realization reverse engineering tools are appearing on the commercial market. Reverse engineering is the process of taking source code and constructing a graphical representation from that code. It is often referred to as design capture since the end product i.e., the graphical representation should resemble the original design.

One of the most difficult problems of reverse engineering is determining "what" the end product is once it is produced. In many cases the end product is pages of arrows and names. Deciphering this overwhelming amount of information is not easy [2, 3, 11].

Reverse engineering tools have evolved towards analyzing programs for porting or cloning these programs. But design recovery tools are aimed at helping the software developer to understand the specific applications and general concepts and designs that comprise classes of applications [1]. It is these general items that become RSCs.

The objective of the design recovery aspect of the RTk is to aid the software developer form a mental model of the software, to record that model for future use, and to make the recovered knowledge easily and readily available for use. This involves taking very detailed information and creating abstract forms of the information that can be applied generically i.e., reused in many products.

Domain analysis (DA) is the first step in creating reusable components [7, 9, 10]. DA captures product expertise that is used to identify common problems and solutions of software systems. The system recognizes domain concepts in code or textual information. Recognition is achieved by pattern matching of names, composition, development, subordinates, and user relationships to other components. Recognized concepts are displayed to the engineer who validates or rejects the accuracy of the recognition which the system automatically incorporates to improves its performance. The system provides the user with methodological guidance primarily for understanding the software and secondly (perhaps more importantly) for re-engineering the software.

1.1.2 Construction

When developing new software components that are potential RSCs the RTk assists the developer in formatting the components for readability, documenting the components for understandability and traceability. Once the component is completed the RTk provides testing mechanisms to assure the accuracy and quality of the component. The component is also categorized for storage in the repository.

1.2 Designing with Reuse

Several questions arise when searching for RSCs to use in the new product:

- Where are the best RSCs?
- What RSCs will be profitable?
- Which RSCs should be developed first?

The reusable component is actually a node with a set of links to the software component's requirements, transformation rule set, analysis, design, interfaces, and test suite nodes. This hypernet of nodes and links comprises all the RSCs and their related items (e.g., tests) plus the information required to use the RSCs. Since different components can share (i.e., reuse) different nodes, the repository becomes a large hypernet of component subnets.

There will be two repositories. One is the master repository where all the proven reusable components are stored. The second repository will be an intermediate holding place for components that are being tested for their correctness, usability, and profitability. Until validated these components must be used with the same precautions as any newly developed software. Once they have qualified as viable reusable components they will be entered into the master repository.

Eventually introducing this technology into a reuse program will automate the development cycle to:

1. The user enters a set of requirements for which the RTk finds a matching RSCs.
2. Based on user input the RSCs will be customized (i.e., transformations will fire) to satisfy the software requirements.
3. RSCs are interfaced to new components.
4. The new product is tested by the automated testing tool.
5. The new components are re-engineered into RSCs for future product development.

1.3 Conclusion

The RTk encompasses design capture, re-engineering, reuse engineering and forward engineering. Although the toolkit is designed to facilitate software reuse, its real purpose is to provide a software design environment that incorporates the best methods and techniques to quickly develope high-quality software.

References

[1] T. Biggerstaff, J. Hoskins, and D. Webster, "DESIRE: A System for Design Recovery," Tech. Rep. STP-081-89, Microelectronics and Computer Technology Corporation, Austin, Texas, 1989.

[2] T. Biggerstaff and M. Lubars, "Recovering and Reusing Software Design - Getting More Mileage From Your Software Assets," Tech. Rep. STP-RU-044-91, Microelectronics and Computer Technology Corporation, Austin, Texas, 1991.

[3] T. Biggerstaff, "Software Reusability," Tech. Rep. MT-119-91, Microelectronics and Computer Technology Corporation, Austin, Texas, 1991.

[4] E. Guerrieri, L. Lashway, and T. Ruegsegger, "An Acquisition Strategy for Populating a Software Reuse Library," Tech. Rep. 302, SofTech, Inc., Waltham, Massachusetts, July 1989.

[5] R. Joos. PhD thesis, Texas A&M University, College Station, Texas, 1989.

[6] R. Joos, "Software Reuse Department," tech. rep., Motorola, August 1990.

[7] M. Lubars, "Domain Analysis for Reuse." Presentation to RTSG, Arlington Heights, IL, May 1990.

[8] R. Prieto-Diaz and P. Freeman, "Classifying Software For Reusability," *IEEE Software*, January 1987.

[9] R. Prieto-Diaz, "Domain Analysis For Reusability," *IEEE Software*, January 1987.

[10] W. Vitaletti and E. Guerrieri, "Domain Analysis within the ISEC RAPID Center," Tech. Rep. 308, SofTech, Inc., Waltham, Massachusetts.

[11] D. Webster, "Domain Modeling and Software Design Information Recovery," Tech. Rep. STP-358-89, Microelectronics and Computer Technology Corporation, Austin, Texas, 1989.